الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي والبحث العلمي

UNIVERSITÉ BADJI MOKHTAR–ANNABA
BADJI MOKHTAR ANNABA– UNIVERSITY

جامعة باجي مختار – عنابـــة

Faculty of Technology
Department of Computer Science

Academic Year 2023/2024

T H E S I S

A thesis submitted in partial fulfillment of the requirement for the degree of Doctor

Formal verification in the interactive and intelligent systems

Field: Computer science

Option: Embedded computing

By:

Ferdaous BENROUBA

Thesis committee members

Committee President	Pr. Salim Ghanemi	Badji Mokhtar Annaba-University
Supervisor	Pr. Rachid Boudour	Badji Mokhtar Annaba-University
Advisor	Dr. Khetatba Mourad	Badji Mokhtar Annaba-University
Advisor	Dr. Bentrad Sassi	Chadli Bendjedid El Taref-University

Acknowledgment

I would like to express my heartfelt gratitude to God, who has been my source of strength and guidance throughout my academic journey. Without His blessings and support, I would not have been able to achieve this milestone.

I would also like to extend my sincere appreciation to my thesis director, **Pr Rachid Boudour**, for his invaluable guidance, insightful feedback, and unwavering support. His expertise and encouragement have been instrumental in shaping my research and academic development.

To my family, my mother, and my siblings, I am grateful for your unconditional love, constant encouragement, and unwavering support. Your faith in me and belief in my abilities have been a driving force behind my success.

Finally, I would like to thank my close friends for their unwavering support and encouragement, both academically and personally. Your presence in my life has made this journey all the more meaningful and enjoyable.

Thank you all for being a part of this incredible journey, and for helping me achieve my academic goals.

Abstract

Computer modeling and simulation of human reasoning, decisions and behavior is an important goal of Artificial Intelligence. Much progress has been made in this direction since the emergence of this research problem and several researchers have proposed works using modal logic to represent human reasoning. One of the best known is the BDI model (for Beliefs, Desire, Intentions) proposed by Bratman in the 1980s. In this model, the designer represents the beliefs of individuals and the system builds plans from the goals.

In this thesis, there is interest in the use of this reasoning model in the context of modeling and automatic verification of systems. Modeling and verification of systems consists in describing the operation of a system, using logical formulas that can be interpreted by a computer, in order to verify certain properties. This model checking technique has given good results for several years in critical systems where the slightest error can have serious consequences. However, these realized tools do not take into account human factors such as beliefs, emotions or personality. However, these can be the source of serious failures in the systems in which the human being is part. One of the most famous examples is the Mont Saint-Odile crash where an erroneous belief about the state of the system led to the death of 97 people despite two very well-trained pilots and an aircraft in perfect working order. Our objective is to propose a verification model that integrates these human factors to better prevent this type of failure.

Keywords: BDI, Formal verification, model checking, EBDI, NUSMV

Résumé

La modélisation et la simulation informatique des raisonnements, des décisions et des comportements des humains est un objectif important de l'Intelligence Artificielle. Beaucoup de progrès ont été accomplis dans cette direction depuis l'apparition de ce problème de recherche et plusieurs chercheurs ont proposé des travaux utilisant la logique modale pour représenter le raisonnement humain. L'un des plus connus est le modèle BDI (pour Beliefs, Desire, Intentions) proposé par Bratman dans les années 1980. Dans ce modèle, le concepteur représente les croyances des individus et le système construit des plans à partir des buts.

Dans cette thèse, il y a intérêt à l'utilisation de ce modèle de raisonnement dans le contexte de la modélisation et de vérification automatique de systèmes. La modélisation et la vérification de systèmes consiste à décrire le fonctionnement d'un système, à l'aide de formules logiques interprétables par un ordinateur, pour en vérifier certaines propriétés. Cette technique de model checking a donné de bons résultats depuis quelques années dans des systèmes critiques où la moindre erreur peut avoir des conséquences graves. Cependant, ces outils réalisés ne prennent pas en compte les facteurs humains comme les croyances, les émotions ou la personnalité. Pourtant, ceux-ci peuvent être la source de graves défaillances dans les systèmes où l'être humain en fait partie. L'un des exemples les plus célèbres est le crash du Mont Saint-Odile où une croyance erronée sur l'état du système a conduit au décès de 97 personnes malgré deux pilotes très bien formés et un appareil en parfait état de marche. Notre objectif est de proposer un modèle de vérification qui intègre ces facteurs humains pour mieux prévenir ce type de défaillances.

Pour atteindre cet objectif, nous avons proposé un nouvel agent BDI émotionnel basé sur la théorie OCC. Il s'ensuit la démonstration de la puissance, la robustesse et l'efficacité de l'architecture proposée en vérifiant le système interactif et intelligent. Enfin, l'efficacité de l'approche suggérée est expérimentée voire validée sur diverses études de cas, y compris le système de maintenance des aéronefs et un scénario d'enchères.

Mots- clés : BDI, Vérification formelle, model checking, EBDI**,** NUSMV.

الملخص

تعد نمذجة الكمبيوتر ومحاكاة التفكير البشري والقرارات والسلوك هدفًا مهمًا للذكاء الاصطناعي. تم إحراز تقدم كبير في هذا الاتجاه منذ ظهور مشكلة البحث هذه واقترح العديد من الباحثين أعمالًا باستخدام المنطق النمطي لتمثيل التفكيرالبشري. أحد أشهر هذه النماذج هو النموذج مرن (للمعتقدات، الرغبة و النوايا) الذي اقترحه براتمان في الثمانينيات.في هذا النموذج، يمثل المصمم معتقدات الأفراد، و يبني النظام خططا من الأهداف

في هذه الأطروحة ، هناك اهتمام باستخدام نموذج التفكير هذا في سياق النمذجة والتحقق التلقائي للأنظمة. تتكون نمذجة الأنظمة والتحقق منها في وصف تشغيل النظام ، باستخدام الصيغ المنطقية التي يمكن تفسيرها بواسطة الكمبيوتر ، من أجل التحقق من خصائص معينة. أعطت تقنية فحص النموذج هذه نتائج جيدة لعدة سنوات في الأنظمة الحرجة حيث يمكن أن يكون لأدنى خطأ عواقب وخيمننة. ومع ذلك ، فإن هذه الأدوات المحققة لا تأخذ في الاعتبار العوامل البشرية مثل المعتقدات أو العواطف أو الشخصية. ومع ذلك ، يمكن أن تكون هذه مصدر إخفاقات خطيرة في الأنظمة التي يكون الإنسان جزءًا منها. أحد أشهر الأمثلة على ذلك هو حادث تحطم مونت سانت أوديل حيث أدى الاعتقاد الخاطئ بحالة النظام إلى مقتل 97 شخصًا على الرغم من وجود طيارين مدربين جيدًا وطائرة تعمل بشكل مثالي. هدفنا هو اقتراح نموذج تحقق يدمج هذه العوامل البشرية لمنع هذا النوع من الفشل بشكل أفضل.

لتحقيق هذا الهدف ، اقترحنا وكيل BDI عاطفي جديدًا يعتمد على نظرية OCC.يتبع عرض قوة ومتانة وكفاءة العمارة المقترحة من خلال التحقق من النظام التفاعلي والذكي. أخيرًا ، يتم اختبار فعالية النهج المقترح أو حتى التحقق من صحته في دراسات الحالة المختلفة ، بما في ذلك نظام صيانة الطائرات وسيناريو المزاد.

الكلمات الرئيسية:BDI، التحقق الرسمي ، فحص النموذج، EBDI، NUSMV

Table of Contents

PART III CONTRIBUTIONS

List of Figures

List of Tables

List of abbreviations

BDI	Belief-Desire-Intention
OCC	Ortony, Clore, Collins
CUDD	Colorado University Decision Diagram
NUSMV	New York University Symbolic Model Verifier
BDD	Binary Decion Diagram
MTBDD	Multi Terminal Binary Decion Diagram
BDIE	Belief-Desire-Intention-Emotion
EBDI	Emotional Belief-Desire-Intention
AI	Artificiel Intelligence
MINISAT	Miniature SAT Solver
TL	Temporel Logic
LTL	Linear Temporel Logic
CTL	Computation-Tree Logic
IMS	Information Management System
NetLogo	Network Logo
IDS	Intrusion Detection System
NLP	Natural Language Processing
FOL	First Order Logic
UML	Unified Modeling Language

PART I: INTRODUCTION

Chapter 1

General introduction

Summary

Chapter 1 : General introduction

1.1 Field of Research

In recent years, work on developing interactive and intelligent systems has flourished in a variety of contexts, from self-driving automobiles to chatbots for medical diagnosis. These have the potential to be extremely beneficial to society, but they also carry the possibility of introducing new risks or amplifying already present ones. Making ensuring that the interactive and intelligent systems failures can be thoroughly investigated and understood so that the safety of these systems can be successfully managed and enhanced is one of the most important risk management problems as these developing technologies become more prevalent. Making sure that the system will not make any mistake is one of the most important risk management strategies, as these emergent technologies spread. Contrary to interactive systems, intelligent system verification has been the focus of in-depth research and studies, leading to important breakthroughs. However, researchers did not devote enough time to interactive system verification. One particular reason is the key cause of its lack of research, the absence of a tool that can compute the human parameters. As it is well known, human is influenced by many things such as religion, beliefs, personality, phycology …etc., the main purpose of this PhD thesis is to conceive and implement a verification tool for the interactive and intelligent systems, we have considered one particular human parameter which is emotion more particularly Fear.

The process of conceding the tool went through three main steps:

1. Modeling the agent

For modeling and reasoning the intelligent system, we have extended the BDI architecture with one particular emotion (FEAR). We then built our methodology on the OCC model's emotional extension of the BDI model. The BDI logic is chosen because it enables a variety of symbolic, stochastic, and sub-symbolic AI techniques, as well as rapid development, context-sensitive and robust behavior, intelligibility and verifiability due to its expressive and intuitive character.

2. Choosing the verification system

The reason behind taking the model checking as a verification system is that it proved its effectiveness and efficiency in verifying systems, besides it is totally automatic, quick, and frequently returns a result in a couple of minutes. In addition, even if the system has not been fully stated, it can be used to check partial specifications and provide important information regarding accuracy.

3. Formally verifying the agent

After accurately applying the OCC theory to model the intelligent agent using BDI logic, now it is time to evaluate the system. To do so, we have employed the NuSMV verification tool. NuSMV was chosen because it was designed to be dependable, flexible, and open so that it could be used in technology transfer initiatives and as a research tool in numerous domains.

In this dissertation, we present a cautious agent that acts with extreme caution, perceiving any potential danger as a source of fear and treating all unpleasant incidents with equal concern. This agent could be used to create a program for cognitive agents that calculates various forms of fear emotions during its operation, including unpleasantness and discomfort.

1.2 Research question

The following research questions are the focus of this thesis:

1. What verification systems are currently well used to verify the interactive and intelligent systems?

2. How may the suggested architecture improve the verification of the intelligent and interactive agents?

3. What distinguishes the proposed work from other systems that presently exist in terms of system verification?

1.3 Thesis motivation

The primary motivations behind the research conducted for this thesis were:

- To conceive and to implement a new model for the interactive and intelligent systems putting in consideration human factors such as emotions (most particaly the emotion of fear).
- To improve the verification of the interactive and intelligent systems by incorporating new emotional BDI agent which is based on the OCC theory.
- Assuring safety and preventing accidents by conduction an aircraft maintenance scenario experiment ensuring flight only proceeds if no critical issues are detected.

1.3 Thesis structure

Six chapters, divided into three sections, make up this manuscript. A general conclusion and future works are included after that. Part I, which is composed of chapter 1, provides a general overview of the thesis and introduces the field study and analysis that will be used throughout the rest of the manuscript, while part II, which is made up of three chapters, discusses the state of the art. In chapter 3, we discussed artificial intelligence and reasoning after introducing the fundamental concepts of interactive and intelligent systems, formal verification, and sentiment analysis in chapter 2. The formal BDI logic and the temporal logic are discussed in chapter 4. Finally, yet importantly, part III, which is where the contributions are located, we introduced our first contribution, which is an implementation of a fearful BDI agent based on the OCC theory, followed by formal verification of the system, in Chapter 5. The second contribution, which is a filtering mechanism used on social media content, is described in Chapter 6, in order to eliminate any material that the end user would perceive to be emotionally damaging.

- ***Chapter 01***: In this chapter, we covered the thesis's research area and introduced some of the challenges associated with verifying intelligent and interactive systems. After that, the motivation for conducting this research was explained along with a detailed explanation of what is being studied. Finally, an overview of the completed project has been outlined.
- ***Chapter 02***: In discussing formal verification, the concept and architecture of intelligent systems, and eventually, introducing the analysis of emotions. We discussed how these concepts are related to each other as well as what problems they can solve.
- ***Chapter 03***: In this chapter, we discussed the fundamental ideas and concepts of artificial intelligence. We defined it, discussed its methods, classified it by type (e.g., rule-based systems vs. knowledge-based systems), and looked at some programming languages used to create AI programs. We also explored how humans think using reasoning skills and clarified the notion of human expertise in AI matters
- ***Chapter 04***: In this chapter, we discussed the basics of computer logic. In the first section, we covered BDI (Beliefs, desires, intentions) logic. The second section focused on temporal logic, which is a more advanced form of logical reasoning that allows for proofs to be constructed in a much more formal way. We introduced several formula interpretations and proof systems in both sections.

- ***Chapter 05***: This chapter discusses the contribution of the thesis, which is a model that combines belief-desire-intention (BDI) logic and temporal logic to improve decision-making in emergency scenarios. In section one; it covers the theory behind this model and its benefits. Second section explores how to create this model using different tools and programming languages. The modelling process then proceeds by specifying multiple scenarios based on real world data. Finally, simulations are conducted to test the results of this model.
- ***Chapter 06***: In this chapter, we covered the second aspect of our research, which proposes a filtering mechanism looks for social media content that may be emotionally harmful for certain users, and how it is designed to remove any such content. We explained that this process is essential in order to protect the mental health of all users, while ensuring that potentially damaging or inflammatory material does not reach them.
- ***Chapter 7***: This document ends with a summary that describes the global work briefly and offers some perspectives on how it should be approached as well as what future works are possible.

PART II: STATE OF THE ART

Chapter 2

A Comprehensive Overview of Verified Human Parameters in Interactive and Intelligent Systems

Summary

Chapter 2 : A Comprehensive Overview of Verified Human Parameters in Interactive and Intelligent Systems

2.1 Introduction

According to recent studies, most of the accidents involving intelligent technologies, which we use on a daily basis, are due to the human error, even though the majority of these systems were tested before being released to the market.

Therefore, the problem is no more to validate an intelligent system but rather, to create an intelligent system where the human operator is present.

In this chapter, we will introduce the interactive and intelligent systems, then, we will provide a global overview about the system verification, where its necessity comes from, its types and objectives, finally, we will go into details about one particular parameter of the human parameters, which is emotions.

2.2 Interactive and intelligent systems

Interactive and intelligent systems are a class of computational systems that are designed to interact with humans and exhibit intelligent behavior. This section provides an overview of the fundamental concepts and technologies that enable the development of these systems. The main objective is to understand the characteristics that define the behavior of intelligent systems and how this behavior can be generated through the integration of different processes and components.

2.2.1 Definition and Applications of Interactive and Intelligent Systems

Interactive and Intelligent Systems are computer systems that interact with their environment, users, and other systems. They have the ability to perceive, reason, act, and interact in order to achieve their goals. Interactive and Intelligent Systems have a wide range of applications, including robotics, gaming, simulation, expert systems, and human-computer interaction. The above definition is summarized visually in Figure 2.1.

2.2.2 Functional components of Interactive and Intelligent Systems

2.2.2.1 Perception

Perception is the process by which an Interactive and Intelligent System acquires information from its environment. This information can come from various sources, including sensors, cameras, microphones, and other data sources. Perception is an essential component of Interactive and Intelligent Systems because it allows the system to understand its environment and the needs of its users.

2.2.2.2 Reasoning

Reasoning is the process by which an Interactive and Intelligent System processes the information it has acquired through perception. This information can be used to make decisions, solve problems, and plan actions. Reasoning can be based on various algorithms, including rule-based systems, decision trees, and artificial neural networks.

2.2.2.3 Action

Action is the process by which an Interactive and Intelligent System performs a task or executes a plan. This can involve physical actions, such as moving a robot or controlling a car, or more abstract actions, such as generating a response or displaying information. Action is the result of the reasoning process and the means by which an Interactive and Intelligent System achieves its goals.

2.2.2.4 Interaction

Interaction is the process by which an Interactive and Intelligent System communicates with its environment, users, and other systems. This can include receiving and processing user inputs, providing feedback and outputs, and exchanging information with other systems. Interaction is an essential component of Interactive and Intelligent Systems because it allows the system to communicate its goals and intentions, receive feedback and guidance, and coordinate with other systems.

2.2.3 The Role of Interactivity in Intelligent Systems

Interactivity is an essential characteristic of intelligent systems. It refers to the capability of these systems to interact with their environment and with humans. This interaction can take different forms, including visual, auditory, and haptic feedback, as well as input from sensors, cameras, and microphones. Interactivity enables intelligent systems to respond to changes in the environment and to provide users with real-time feedback and control over the behavior of the system.

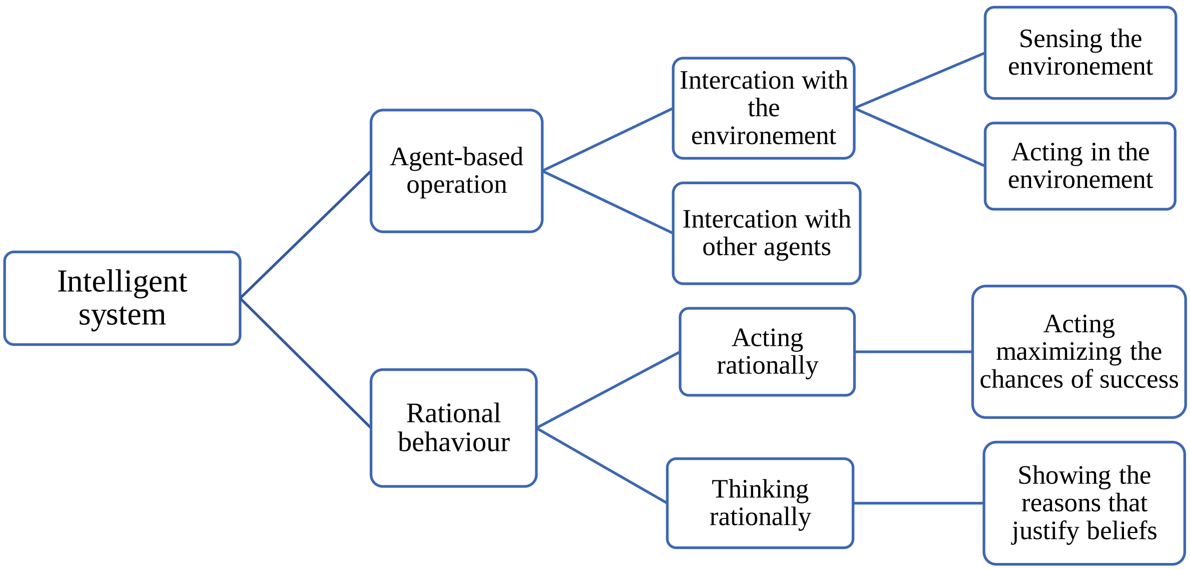

Figure 2.1: Characterization of an intelligent and interactive system (Molina, 2020)

2.2.4 Processes and Components of Intelligent Systems

Intelligent systems can be decomposed into several processes and components that work together to produce intelligent behavior. Some of the main processes and components of intelligent systems include perception, reasoning, decision-making, and control. These processes and components can be implemented using a variety of techniques, such as machine learning, computer vision, natural language processing, and robotics.

2.2.5 The Design of Interactive and Intelligent Systems

The design of interactive and intelligent systems involves several stages, including the definition of the system's requirements, the selection of appropriate technologies and components, the development of the system architecture, and the implementation of the system's functionalities. The main challenge in the design of these systems is to ensure that the system integrates different processes and components in a way that produces coherent and effective behavior.

2.2.6 Evaluation of Intelligent Systems

The evaluation of intelligent systems is an essential step in the design process. The main objective of evaluation is to assess the system's performance with respect to the requirements that have been established and to identify any limitations or issues that may affect the system's functionality. Evaluation can be performed using a variety of methods, including simulation, experimentation, and field trials.

2.2.7 Future Trends in Interactive and Intelligent Systems

Interactive and intelligent systems are an active area of research and development, with new technologies and techniques being developed and applied regularly. Some of the main trends in this field include the development of more advanced machine learning algorithms, the integration of cognitive computing and robotics, and the deployment of these systems in a wide range of applications, including healthcare, education, entertainment, and transportation.

Figure 2.2 depicts a block diagram that, at a high level of abstraction, identifies the primary operations of an intelligent system. A function termed environment understanding, for instance, is depicted in the image and encompasses both perception of the surroundings and situation appraisal. Acting on the environment is another key responsibility, and it may be broken down into two parts: choosing what needs to be done (for example, through planning or configuration), and doing it while keeping an eye on how it is going. The ability to communicate with others, which typically requires both language creation and interpretation, is yet another key function.

Interactive and intelligent systems are a rapidly evolving field that is shaping the future of human-computer interaction. The development of these systems requires a deep understanding of the fundamental concepts and technologies that enable their behavior, as well as a clear understanding of the design and evaluation processes that are involved in their creation. In this section, we have provided a global overview of the interactive and intelligent system, its concept and its technologies.

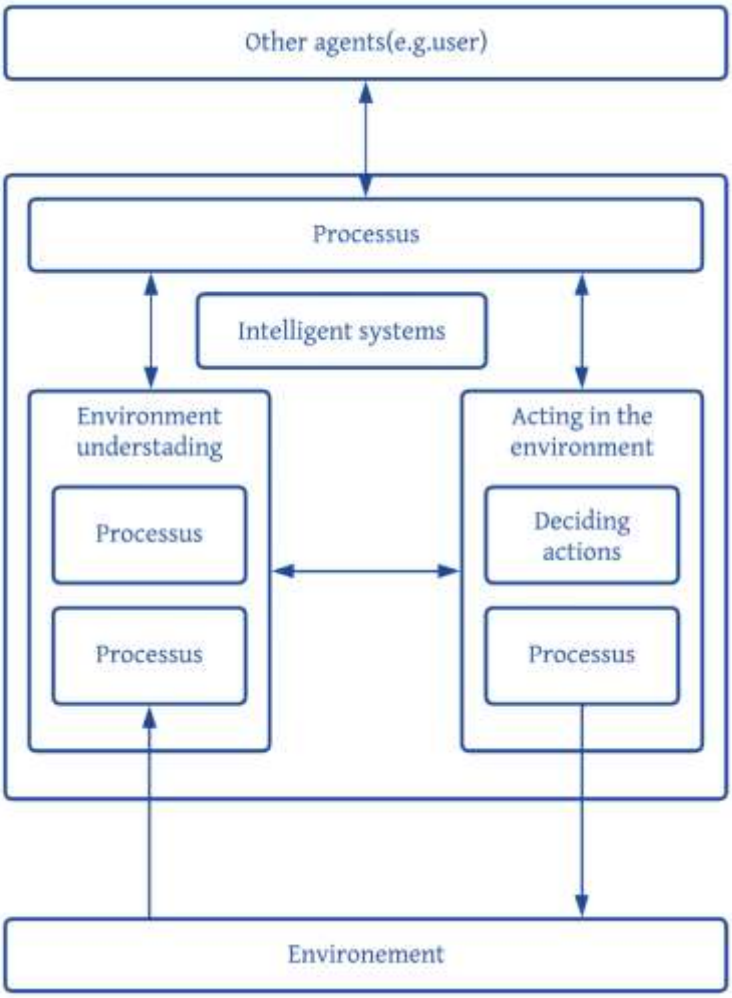

Figure 2.2: High-level abstraction diagram illustrating the fundamental operations of an intelligent system (Molina, 2020)

2.3 Formal verification

Human's reliance on intelligent systems is growing day by day. Today, it is estimated that 70% of daily life devices feature one kind of artificial intelligence or another. This reliance comes from the capacity of these devices in terms of response time, image treatment, and many other measurements. But when it comes to our safety and security, we forget about these things and focus only on one thing more important: "correctness." We ask ourselves questions such as, "Are these systems verified enough to be reliable?" or "Are these systems critical?" The absence of absolute correctness in some AI systems, such as air-traffic control systems, disaster management systems, self-driving cars, etc., can threaten our lives. Regardless of the material losses that can occur, our security comes first. Hence, the necessity arises to build a verification system that can provide information about the correctness of the critical system being built, with no chance of error.

System verification basically aims to answer one question: "Have we realized the end product properly?" or "Was correctness guaranteed in the product?" or even more, "Is the product free from errors?" System verification consists of ensuring that a certain product or model possesses certain properties. These properties are essentially instances of something else called specifications. A specification defines what the system should do and what it should not. For instance, ensuring that the system will never be in a deadlock scenario is a property that comes from the specification that the system has to be robust. Thus, we come to the conclusion that system verification is relative to specification and not an absolute property. We consider a product correct once it fulfills all its specifications. Once it violates one of its property specifications, we exclude it as a correct system. In a nutshell, system verification classifies systems into two columns: correct and not correct, as shown in (Figure 2.3) bellow.

2.3.1 Why system verification

The main goal of system verification is to establish confidence that the system is free of all possible defects or bugs and to prove its surety and sureness. For some products, it is impossible to achieve absolute correctness but rather correctness in the expected use case, which defines how much the system is correct. Unfortunately, some users accept this fact, as seen in the case of Windows 95. Microsoft itself announced that the system contained at least 5000 defects. Despite the system's success at the time, this should not be tolerated because the system should be correct and safe in every condition and in every use case.

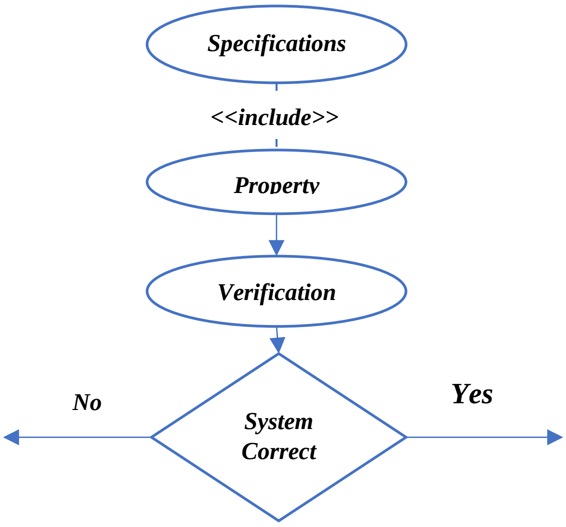

Figure 2.3: System verification process

2.3.2 Hardware and software system verification

2.3.2.1 Software verification

Software verification must be applied in the early stages of the production process, empirical studies show that fixing an error in a late stage of the production process can cost 500% (NASA Johnson Space Center, 2004) comparing to the costs of the same error when fixing it in an early stage, an error in the baggage handling system caused the postponed of an airport for 9 months and a loss of 1.1 US million dollar per day in an American airport (Baier & Katoen, 2008).

Peer reviewing and testing are the two major techniques used in the software verification.

A. *Static verification*

Peer reviewing: also known as code inspection, it consists of statically (and manually) analysing the product (the code) without executing it, the peer reviewing has proved its efficiency in catching errors and defects, and especially when it comes to catching a specific error or bug, empirical studies show that peer reviewing can detect between 31% to 91% (Baier & Katoen, 2008)of defects with a median of 60%, in the other hand it is hard, and because of its static nature it is hard to catch subtle errors such as concurrency and algorithm defects using this technique.

B. *Dynamic* verification

Test: test verification is the opposite of the peer reviewing, it consists on of dynamically (semi automatically) execute the code in order to find any mismatch between the outputs and the expected results which means the presence of error, it is generally the responsibility of an independent testing team, the main disadvantage of tests is that it used to discover the presence of errors and not there absence, otherwise tests can never be completed, and we can never test all possible scenarios, otherwise it is estimated that 30% to 50% of software projects costs are devoted to making testing thus we spent more time and efforts on making detecting errors rather than constructing.

C. *Testing Goals:*

1- Discovering errors or flaws in the program where its performance is wrong or does not conform to its specification;

2- A successful test is one that causes the system to behave erroneously and so reveals a flaw in the system.

Why testing is not enough

1. Software testing's primary objective is to find software bugs, and its secondary objective—which is achieved even if no bugs are found—is to increase confidence in the software's ability to function properly.
2. Without further context, it can be concluded that either the software is of high quality or the testing process is not thorough.
3. Edsger Dijkstra stated that although software testing can reveal the existence of bugs, it couldn’t guarantee their absence.
4. It is not possible to perform an all-encompassing test (Test is not exhaustive)

Debugging:

Debugging involves finding and fixing the faults in software. The process involves forming a hypothesis about the program's behavior, then testing it to locate the system error. Studies indicate that half of the production time for software products is spent on the design phase, while the rest is used for system verification. It has been suggested that fixing a specification error takes one time unit, while fixing a design error takes ten units, a coding error takes 100 units, and an integration testing error takes 1000 units. These figures have not been formally proven, but the idea that later errors are more difficult to fix appears to be valid.

2.3.2.2 Hardware verification

Catching errors and defects in the early stages of hardware fabrication is crucial. The further we progress in the production process, the more difficult it becomes to catch errors, and the more expensive they are to fix. "The more we progress, the more we pay." Additionally, there is no way to fix errors after delivery to the client. Unlike software verification, where we can provide an update or a new version, a simple error in the floating-point division unit cost Intel $475 million to replace the Pentium II (Baier & Katoen, 2008). This incident damaged Intel's reputation at that time.

Emulation, simulation and structural analysis are the three major techniques used in the hardware verification.

A. Emulation verification

Emulation verification involves utilizing a hardware generic reconfigurable system known as an Emulator. This system is adaptable to mimic the behavior of the product under examination. All potential inputs in various system states are administered to the product, and the resulting outputs are compared to the anticipated results. Any disparities indicate the presence of errors. However, applying numerous tests can lead to a combinatorial explosion. In such instances, it is necessary to eliminate tests that obviously will not detect defects or errors.

B. Simulation verification

Simulation verification differs significantly from emulation verification. Instead of testing the product directly, we test models constructed using hardware description languages like Verilog or VHDL (Verification Hardware Description Language). Inputs to these models can be determined manually by a user or automatically using a random generator. The outputs obtained from the simulation are then compared to the expected results outlined in the specification. Any disparities between the specification and the output signify the presence of errors.

C. Structural analysis

The structural analysis combines between different techniques such as synthesis and equivalence checking.

Empirical studies show that only 27% of the production time of a hardware product is dedicated to the design process, the rest of time is spent on system verification, all these efforts to prevent defects from happening and errors from being.

2.3.3 Formal verification

In their book "Formal Verification: An Essential Toolkit for Modern VLSI Design," (Erik Seligman et al., 2015) introduced formal verification as "the use of tools that mathematically analyze the space of possible behaviors of a design, rather than computing results for particular values." Alternatively, formal methods can be defined as the application of mathematics for modeling and analyzing information and technology systems.

2.3.3.1 Why formal verification:

Formal verification provides a large potential for:

- Obtaining an early integration of verification in the design process.
- Reducing the verification time.
- Providing more effective verification tools and techniques (high coverage).

The final report of an investigation about the use of the formal methods made by NASA (National Aeronautics and Space Administration) and FAA (Federal Aviation Authority) concludes that “Formal methods should be part of the education of every computer scientist and software engineer, just as the appropriate branch of applied maths is a necessary part of the education of all other engineers”.

Deductive methods, model cheeking and testing are the three majors formal methods used in the formal verification, in the section below we are going to make a benchmark of methods (about the principals of each method, tools, application, advantages and disadvantages-see TABLE 2.1).

TABLE 2.1: Table comparing the different tools of the formal verification

Tools	***Method***	***Tool***	***Applicable***	***Advantages***	***Disadvantages***
Deductive methods	provide a formal proof that a property p holds for (a given state in) that model	theorem, proof assistant or a proof checker	if system has form of mathematical theory	exhaustive	Difficult to implement, size limitation, expensive
Model cheeking	systematically checks on a property p in all states	model checker (SPIN, NUSMUV... etc)	if system generates a finite behavioral model	exhaustive, partially automatic	moderately difficult to implement, size limitation, no generalization
Testing	test for property p by exploring all possible behaviors	automated test generation	in almost all systems	real systems execution, errors discovery at all levels (specification, design, and implementation), easy to implement	impossible completeness

Model checking is the only formal verification technique that is of interest in this PhD dissertation; other tools are not taken into account and are not further considered.

2.3.3.2 Model checking

A. Definition

In their book "Principles of Model Checking," (Baier & Katoen, 2008) introduced model checking as "an automated technique that, given a finite-state model of a system and a formal property, systematically checks whether this property holds for (a given state in) that model." In other words, model checking is a formal verification technique that systematically explores all possible states of a system and examines them using a software tool called a model checker to verify that the system satisfies a certain property specification or violates it. In some cases, models become too large to be handled using simple means (due to insufficient memory), so it is necessary to use clever techniques and algorithms to address these issues. Model checking can eventually reveal subtle errors that cannot be verified using testing or emulations. Figure 2.4 below depicts the model checking workflow.

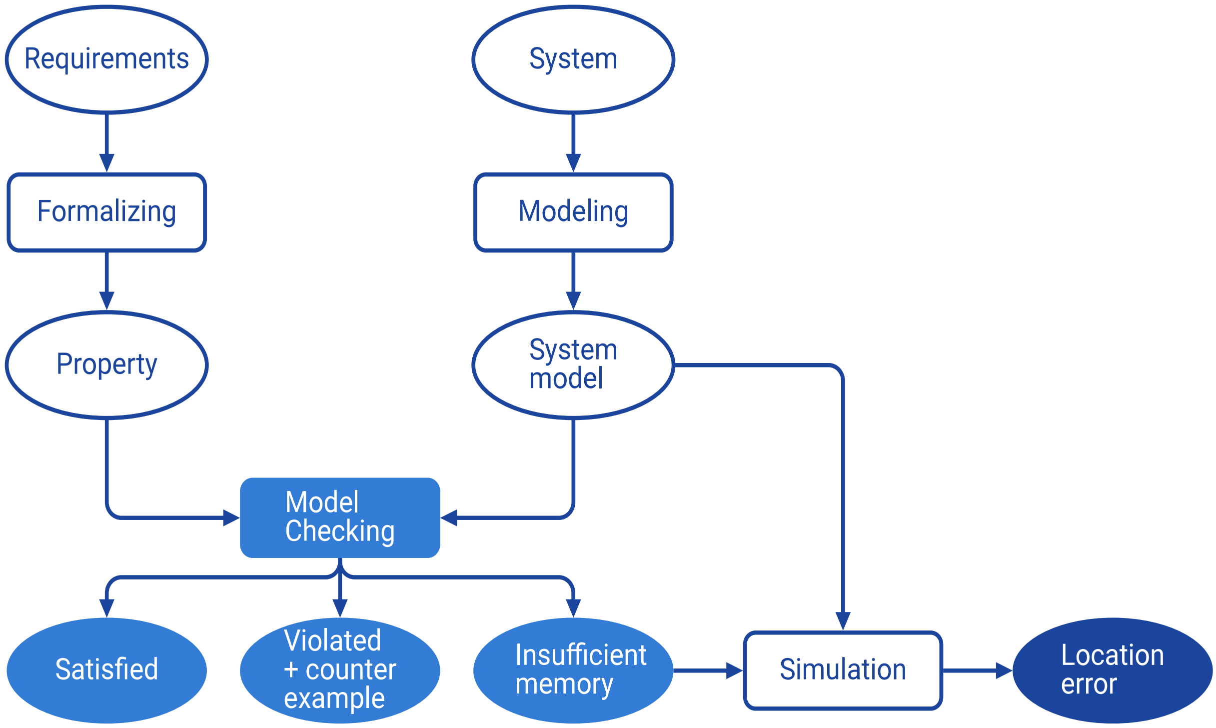

Figure 2.4: The model cheeking process (Baier & Katoen, 2008)

B. Model cheeking process

i. Formalizing requirements

We need to know what is exactly the property that we want to check, it cannot be some kind of vague statement it needs to be precise and unambiguous formulation to start from.

ii. Property specification:

Properties are described using property specification language, it is mostly used the temporal logic, Temporal logics are formalisms adapted to express properties involving the time scheduling notion, it is based mainly to verify that description holds temporal logic formula, and this is where the term "Model Cheeking" comes from, temporal logic is known for its Concision (brevity), expressiveness and simplicity.

iii. Modelization

Literally we can define modelization as "The process by which we go from general to particular, diversity to unity", thus modelization can be seen as reduction, where reality is reduced to one of its dimensions, in formal verification modelling is the process of describing the behavior of a system in an accurate and unambiguous way , it is mostly done by using finite state automata, where states gives information about the current value of the variables, the previous statements and the like, and transitions present the passage from one state to another, finite state automata generally described used a model description languages such as JAVA or WHDL, here we gather some definitions of modelization according to different scientists

- “For an observer, an object M is a model of an object O if the observer can use M to answer questions that interests it about O” (Minsky, 1965)
- “A model is a representative framework, idealized and open, recognized as approximate and schematic, but judged fruitful in relation to a given goal” (Frigg, 1920) .
- “A scientific theory is formalized as a mathematical model of reality, from which can be deduced or calculated the observable properties and of a well-defined class of processes in the physical world” (Frigg, 1920).

Making system modelling in a wrong way, maybe gives a correct output, but it will never validate the property in the model system, because it is not the system that we want to verify anymore, like that we come to what is common knowledge in information technology “Garbage in garbage out”, so it is desirable to make modelling in very conscious way.

iv. Running phase

The system models and the specification properties are given to the model checker in order to verify and validate that the system model satisfies the property specification, three outputs then are given: satisfied, violated or insufficient memory.

v. Analysing phase

The outcomes of the model checker can result in one of three possible results. Firstly, "valid" signifies that the property specification satisfies the system model. In such a case, we proceed to validate the next property, and if none remain, we can affirm that the system is correct. Secondly, "violated" indicates that a counterexample must be provided, akin to diagnostic information, elucidating why the model is violated. Various reasons could underlie the occurrence of a bug. It could stem from a modeling error, indicating that the system does not accurately reflect the design. In such instances, the model should be rectified, and the verification process must be restarted. Another possible reason is a mismatch between the property specification and the requirement, yielding a negative output. Alternatively, it could simply be a genuine mistake. Information about the error's location can be gleaned from the counterexample, rendering it a sophisticated debugging technique. Thirdly, "insufficient memory" may arise as a physical limitation in the computer employed in the verification process. This indicates that the number of states is too vast to manage. In such cases, ingenious techniques are necessary to circumvent this problem.

vi. Verification organization

In order to reproduce the realised experiences in the verification process and to have a practical model cheeking, the entire model cheeking processes should be organised, well structured and well planned, the industrial applications of the model cheeking have proved that the use of version and configuration managements are pertinent, for example descriptions about different part of the model, verification results and system's version, all these information should be preserved very carefully.

vii. Transition System

States, transitions and states are the three basic concepts in the transition systems where states are labelled with basic propositions, transitions relate between states and actions

Transition systems are direct graphs where nodes represent states, and the edges represent transitions denoting the state changes. A state encapsulates information of the system (values of the system variables) at a particular moment in time during its execution. Transition system is one of the most used modelling formalism for verification.(Seshia & Subramanyan, 2018)

viii. Petri Net

Petri Net is a powerful modelling formalism, as it can represent concurrent, asynchronous, distributed, parallel, non-deterministic, and stochastic systems and simulate their dynamic and concurrent activities.

A Petri Net is a directed graph with an initial state. The nodes can be either a place or a transition, and a place holds zero or more tokens. Places represent conditions and transitions represent events. Arcs can only link opposite kind of nodes (from a place to transition, or the opposite). A state of the system is represented by a marking M_i: number of tokens at each place at a certain moment (MURATA, 1989).

ix. Process algebra

Process algebra was developed in the late seventies by Hoare(Hoare et al., 1978). It was developed as an imperative programming language that enables specifying reactive and concurrent systems through the composition of process. All actions in CSP are an occurrence of events, where this later requires the agreement (synchronization) of both the process and its environment. Every process has a set of events that can communicate (in or out), and the algebra offers a collection of operators to compose the processes (prefixing, sequential and parallel composition, and hiding), in addition to an equivalence relation and a refinement ordering relation.

Model checking verifies if the model satisfies a formal specification (i. e. if it has certain properties). The following section defines what is a property, its types, and its specification languages.

C. Advantages and disadvantages of model checking

i. Advantages

- Model checking can provide exhaustive and systematic verification, which can catch errors that may not be found by other testing methods.
- Model checking can be applied early in the design phase, which makes it easier and less costly to find and fix errors.
- Model checking can automatically verify large and complex systems, which would be infeasible to do manually.

ii. Disadvantages

- Model checking can be computationally expensive and time-consuming, especially for large and complex systems.
- Model checking requires a high level of formal specification and mathematical expertise, which can be a barrier to its widespread use.
- Model checking can be limited to verifying the behavior of a system based on the assumptions and limitations of the model, which may not reflect the actual behavior of the system.

In summary, model checking is a powerful verification technique, but its use requires careful consideration of its advantages and disadvantages, as well as the expertise and resources available.

In this section, we have given a comprehensive overview of system verification, including its origin, various types, and main goals.

2.4 Emotional analysis

Emotional analysis is the process of determining and measuring the emotional tone of written or spoken language. This involves identifying emotions such as happiness, anger, sadness, fear, and surprise in text, speech, or other forms of communication. Emotional analysis is used to better understand how people feel about a particular topic or product, and to provide a more complete and accurate understanding of the emotional content of text or speech.

2.4.1 Six basic emotions by Eckman

Paul Ekman, a psychologist and expert in the study of emotions (Ekman, 2003), identified six basic emotions that are universally recognized across cultures:

A. Happiness

A feeling of pleasure or contentment, often accompanied by a positive outlook on life.

B. Sadness

A feeling of melancholy or sorrow, often accompanied by feelings of hopelessness and helplessness.

C. Fear

A feeling of anxiety or apprehension in response to a perceived threat. It is often accompanied by physical symptoms such as a rapid heartbeat and sweating.

D. Disgust

A feeling of revulsion or distaste, often in response to something perceived as unpleasant or contaminated.

E. Anger

A strong feeling of annoyance, displeasure, or hostility. It is often accompanied by physical sensations such as an increased heart rate and tense muscles.

F. Surprise

A feeling of amazement or disbelief, often in response to unexpected or sudden events.

In this PhD thesis, we are interested in one particular emotion, which is ***fear***, other emotions are not taken into consideration, and they are not further discussed here.

2.4.2 Methods of Emotional Analysis

There are several methods used in emotional analysis, including natural language processing (NLP), machine learning algorithms, and sentiment analysis. NLP involves using computational techniques to analyze and understand the meaning of language. Machine learning algorithms use statistical models to classify emotional tones in language. Sentiment analysis involves using NLP and machine learning to determine the polarity of a text, such as positive, negative, or neutral.

2.4.3 Emotional Analysis Process

The emotional analysis process typically consists of the following steps:

A. Data Collection

The first step in emotional analysis is to gather data in the form of text, speech, or other forms of communication. This data can be collected from a variety of sources, such as social media, customer feedback, or surveys.

B. Pre-processing

The next step is to pre-process the data to clean and prepare it for analysis. This may include removing irrelevant information, correcting errors, and transforming the data into a format that can be easily analyzed.

C. Feature Extraction

In this step, the data is transformed into a set of features that can be used to represent emotions. This may involve creating a vocabulary of emotional terms, or using machine learning algorithms to extract relevant features from the data.

D. Emotion Classification

The next step is to classify the emotions in the data. This can be done using NLP techniques, machine learning algorithms, or sentiment analysis. The goal is to assign a label to each data point, indicating the emotional tone of the text or speech.

E. Emotion Quantification

Once the emotions have been classified, the next step is to quantify them. This can be done by counting the frequency of each emotion in the data, or by calculating an overall score for each data point.

F. Results Interpretation

Finally, the results of the emotional analysis are interpreted to gain insights into the emotions expressed in the data. This may involve visualizing the results, comparing the emotions between different data sets, or identifying trends and patterns in the emotions.

The process of emotional analysis is summarized in Figure 2.5 below.

2.4.4 Applications of Emotional Analysis

Emotional analysis has a wide range of applications in fields such as psychology, market research, and customer service. In psychology, emotional analysis can be used to better understand the emotional experiences of individuals, and to develop treatments for emotional disorders. In market research, emotional analysis can be used to gain insights into consumer attitudes and preferences, and to develop more effective marketing strategies. In customer service, emotional analysis can be used to understand the emotional experiences of customers and to improve customer satisfaction.

2.5 Related works

The paper by (Akhtar et al., 2014) presents an approach for the analysis, design, and formal verification of a multi-agent based university Information Management System (IMS). The IMS is based on the BDI agent architecture, which models the system based on belief, desire, and intentions. The orchestrator agent manages the coordination between all agents and the database connectivity for the system. The correctness properties of safety and liveness are specified using First-order predicate logic. The formal verification of correctness properties provides a mathematically correct foundation for the architectural design and implementation of the IMS. The system generates reports, such as student results, teachers to student ratio, number of students admitted in a year, and number of teachers in a program. The IMS ensures no loss of data during uploading, updating, and report generation, by ensuring all required fields are updated and no field is empty in the database.

In their paper titled "Avionics self-adaptive software: Towards formal verification and validation", authors (D'Souza & Kashi, 2019) highlited an extensive literature survey, offering a comprehensive overview of research within the domain of avionics self-adaptive software. The survey encompasses diverse facets such as requirements delineated in CTL, design and architecture employing the BDI formalism, learning algorithms for adaptivity planning, prototype implementation using NetLogo, formal verification via abstraction and model checking, and validation through the assessment of adaptivity measures. Additionally, the survey delves into aspects like co-operativeness measures for adaptability, boolean abstractions, NuSMV modelling for verification, and the application of the Q-learning algorithm in a case study involving an unmanned air vehicle. This thorough exploration of the literature landscape provides valuable insights and forms a foundational basis for the paper's subsequent contributions and discussions.

The paper by (Perháč et al., 2021), introduces a logical model for an active Intrusion Detection System (IDS) incorporating category theory, coalgebras, linear logic, and Belief-Desire-Intention (BDI) logic, enabling not only intrusion detection but also autonomous response in accordance with predefined security policies. Through a motivating example involving real network intrusions, the authors illustrate the effectiveness of their approach. Emphasizing the utilization of BDI logic, the paper elucidates how beliefs, desires, and intentions are represented within the IDS model. Furthermore, the operational process of the IDS is articulated through modal formulae delineating belief, desire, and intention, while its structure can be visualized using a commutative diagram.

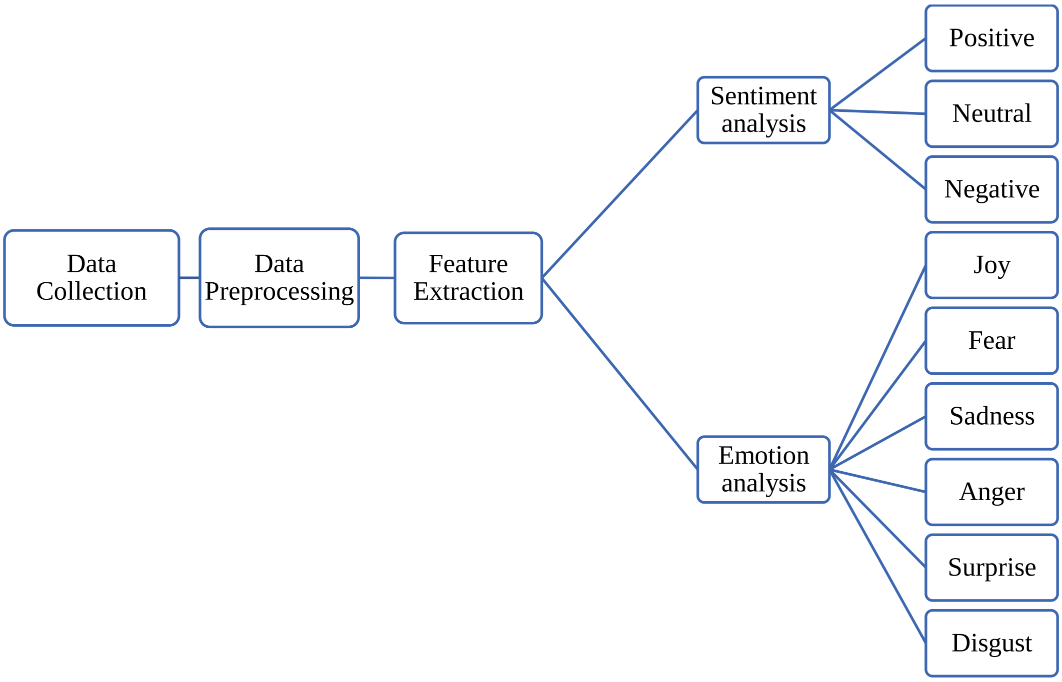

Figure 2.5: Schema showing the emotional analysis process (Mohamed Ridhwan & Hargreaves, 2021)

2.6 Conclusion

We discussed interactive systems, formal verification, and emotion analysis in general terms in this chapter. We also established the formal verification principle and provided definitions to help readers understand the key ideas of this field. We also touched on the design of a interactive and intelligent systems. By the end, we have discussed the six fundamental emotions as outlined by Eckaman.

We shall concentrate on artificial intelligence and reasoning in the following chapter.

Chapter 3

Artificial intelligence and Reasoning

Summary

Chapter 3 : Artificial intelligence and Reasoning

3.1 Introduction

With the arrival of the Internet, the world has become hyper-connected, in which each manipulated object (cars, refrigerators, clothes, social networks, etc.) generates millions of additional data every day, adding to an ocean of unlimited data.

All this data can be used to guarantee personalized services and is expected immediately. Nevertheless, how can such a bottomless ocean of data be transformed into a regular flow of relevant information to meet these expectations? The answer is artificial intelligence (AI).

In this chapter, we will discuss the basic principles and concepts of artificial intelligence, its definition, approaches, types, and programming languages of AI and also, we explain the principle of reasoning in artificial intelligence and understand how we can reason and the types of this reasoning.

3.1 Artificial intelligence

3.1.1 Definition

John McCarthy, who is regarded as the pioneer of Artificial Intelligence, described AI as a field of study and practice that aims to create intelligent machines and computer programs. He saw AI as encompassing the development of algorithms and systems that can mimic human intelligence in areas such as understanding language, recognizing images, and making decisions (Mccarthy, 2007).

However, the beginning of the discussion around Artificial Intelligence can be traced back to Alan Turing's 1950 publication (Turing, 1950). Turing, also known as the "father of computer science," raised the question "Can machines think?" and proposed the "Turing Test," where a human evaluator would try to distinguish between a response from a computer and a human through text. Despite facing criticism over the years, the Turing Test remains a significant part of AI's history and a relevant topic in philosophy due to its focus on linguistics.

The next step was for Stuart Russell and Peter Norvig to publish (Stuart J. Russell & Peter Norvig, 1995)), which went on to become one of the most influential books on AI. They explore four alternative objectives or definitions of AI in it, differentiating between computer systems based on reason and reasoning versus acting:

3.1.2 Approach of artificial intelligence

Table 3.1 below summarizes the differences between the human approach and the ideal approach to AI:

Table 3.1: Table summarizes the different approaches of the AI

Human Approach	***Ideal Approach***
Systems that think like humans	Systems that think rationally
Systems that act like humans	Systems that act rationally
Relies on human experiences and knowledge	Relies on its own experiences and knowledge, acquired through learning and interaction with its environment
Incorporates elements of human creativity, intuition, and empathy	Incorporates elements of logical and mathematical reasoning
Limited by human biases and limitations	Capable of learning and adapting, allowing it to overcome its own limitations and biases

Note that the ideal approach to AI is still a work in progress, and many current AI systems are limited in their ability to function according to this ideal approach. However, as the field of AI continues to evolve, it is likely that AI systems will become increasingly sophisticated and capable of functioning according to the ideal approach.

3.1.3 Deep learning and Machine learning

It is common to use the terms "deep learning" and "machine learning" interchangeably. Despite the fact that each word is distinct from the other, we can nevertheless deduce from the graphic below that deep learning and machine learning are both subfields of the artificial intelligence. As the deep learning is a subdomain of category of machine learning (IBM Cloud Education, 2020).Figure 3.1 shows how deep learning fits into the bigger picture of machine learning. It helps to understand how deep learning is a part of artificial intelligence alongside other machine learning techniques .

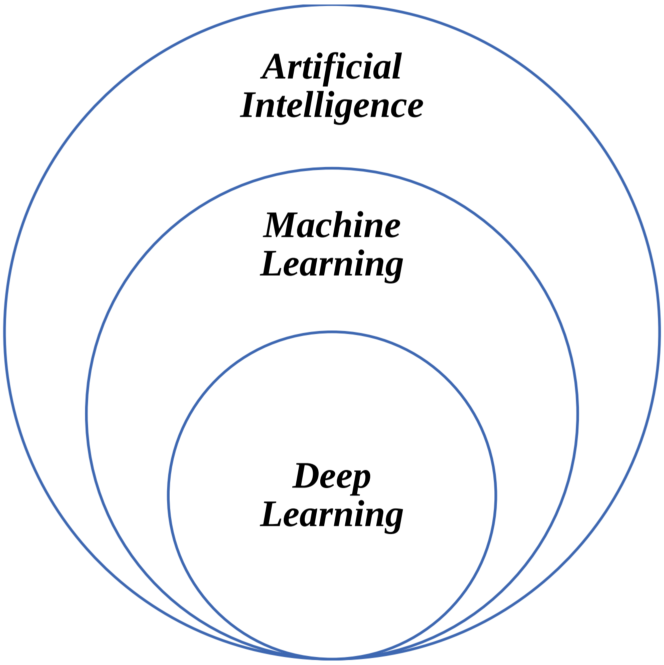

Figure 3.1: A schema of the artificial intelligence and its related fields (Balodi, 2020)

Deep learning involves the use of neural networks. The term "deep" in deep learning refers to the number of layers in a neural network. A neural network with more than three layers, including the input and output layers, is considered a deep learning algorithm. This is typically depicted by the diagram (Figure 3.2) below:

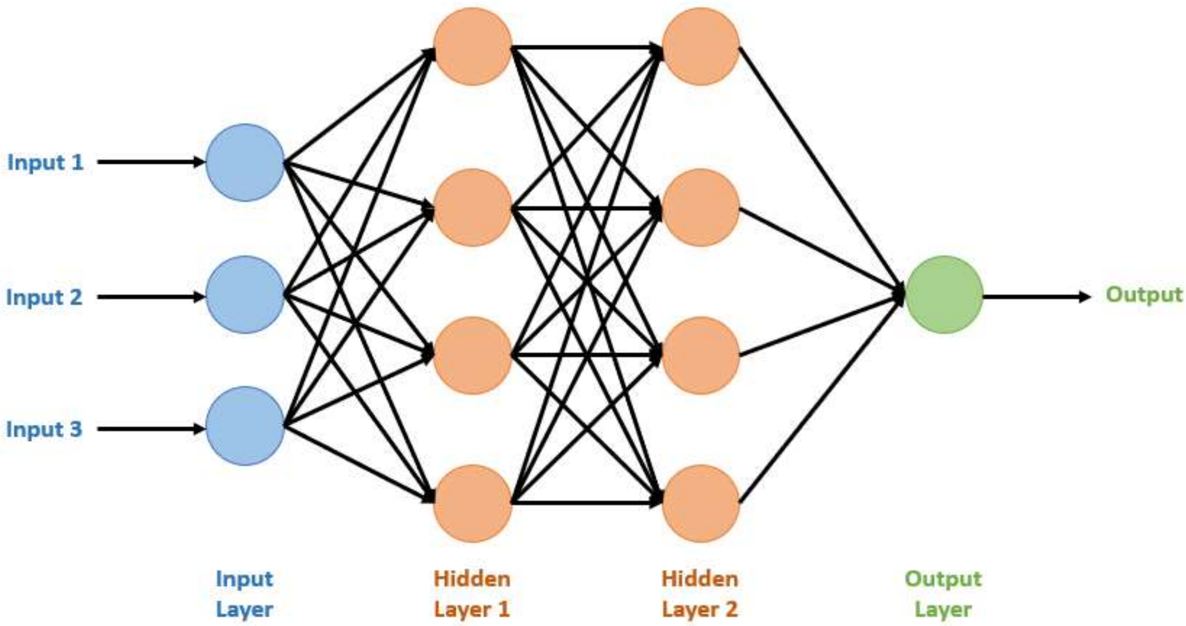

Figure 3.2: Deep neural network diagram (Lee et al., 2021)

How deep learning and machine learning differ is in how each algorithm learns. The table below (TABLE 3.2) illustrates the main differences between them both:

TABLE 3.2: Comparison of deep learning and machine learning

Feature	***Deep Learning***	***Machine Learning***
Approach	Deep neural networks inspired by the human brain	A variety of algorithms
Complexity	Capable of solving complex problems	Can handle simple to moderately complex problems
Data requirements	Requires large amounts of data	Can work with relatively small amounts of data
Automation	Has the ability to learn and improve with minimal human intervention	Often requires feature engineering and manual tuning
Performance	Often achieves state-of-the-art results	Performance varies based on the algorithm and the data

In summary, deep learning is a specialized form of machine learning that uses deep neural networks to solve complex problems, and is capable of achieving better results than traditional machine learning algorithms with large amounts of data.

3.1.4 Strong AI vs. weak AI

AI can be categorized as either weak or strong.

i. ***Weak AI*** (also known as narrow AI): This type of AI is designed to perform a specific task, such as image recognition or language translation. Weak AI systems are programmed to solve a particular problem and do not have the ability to perform tasks outside of their programming.

ii. **Strong AI** (also known as artificial general intelligence): This type of AI is capable of performing any intellectual task that a human can do. Strong AI systems have human-like intelligence and the ability to learn, reason, and make decisions.

3.1.5 Types of artificial intelligence

Arend Hintze, an assistant professor of integrative biology and engineering and computer science at Michigan State University, published an article in 2016 in which he divided AI to four different categories (Arend Hintze, 2016), starting with the widely utilized task-specific intelligent systems and ending with perceptual systems. indicated that it cannot yet exist, but it can be classed into. These are the categories:

- ***Task-specific intelligent systems:*** These systems are designed to perform a specific task, such as image recognition or language translation. They are commonly known as weak AI or narrow AI.
- ***Perceptual systems:*** These systems have the ability to perceive their environment through sensors, such as cameras and microphones, and respond to changes in the environment. They do not yet exist.
- ***Cognitive systems:*** These systems have the ability to learn and make decisions based on that learning. They do not yet exist.
- ***Self-aware systems:*** These systems have a level of consciousness and self-awareness, and are capable of introspection and decision-making based on their own beliefs and desires. They do not yet exist.

3.1.6 AI programming languages

The history of AI programming languages dates back to the 1950s when computer programs were first developed to solve mathematical problems, the following are some of the most significant AI programming languages:

- ***IPL (Information Processing Language):*** IPL was developed in the late 1950s and is considered one of the earliest AI programming languages. It was designed for symbolic processing and was used for early AI research. IPL was influential in the development of LISP and other AI programming languages.
- ***LISP (List Processing Language):*** LISP was developed in the late 1950s and is considered one of the earliest AI programming languages. It was designed to support symbolic processing, which is essential for many AI algorithms. LISP has been widely used in AI research and has been instrumental in the development of early expert systems.
- ***FORTRAN (Formula Translation):*** FORTRAN was developed in the mid-1950s and was one of the first high-level programming languages. It was designed for scientific and engineering applications and was used for early AI programming.
- ***PROLOG (Programming in Logic):*** PROLOG was developed in the early 1970s and is based on the principles of formal logic. It is particularly well-suited for knowledge

representation and reasoning, and has been widely used in the development of expert systems.

- ***Python:*** Python is a modern, high-level programming language that has become increasingly popular for AI programming in recent years. It has a large number of libraries and tools for machine learning and deep learning, making it easier for developers to build AI systems.
- ***TensorFlow***: TensorFlow is an open-source software library developed by Google for machine learning and deep learning. It has a powerful computation engine and a user-friendly interface, making it a popular choice for AI development.
- ***PyTorch***: PyTorch is an open-source machine learning library based on the Torch library, developed by Facebook. It is designed for ease of use and provides high-level APIs for building and training machine learning models.

The history of AI programming languages is marked by the development of various languages that have been instrumental in the creation of AI systems and the growth of AI as a field. These languages continue to evolve and advance to support the development of even more sophisticated AI applications.

3.1.7 Augmented intelligence vs artificial intelligence

According to some business experts, the phrase "artificial intelligence" is too intimately associated with popular culture, and as a result, the general public has unrealistically high expectations about how AI will transform the workplace and life in general.

- ***Artificial Intelligence (AI)*** refers to the development of computer systems that can perform tasks that typically require human intelligence, such as visual perception, speech recognition, decision-making, and language translation. AI systems are designed to replace human intelligence and automate certain tasks (Yau et al., 2021).
- ***Augmented intelligence***, on the other hand, refers to the integration of AI systems with human intelligence to enhance human capabilities and decision-making processes. Augmented intelligence systems are designed to work alongside humans, providing support and assistance, rather than replacing human intelligence (Yau et al., 2021).

While both augmented intelligence and AI aim to improve human productivity and decision-making, they have different goals and applications, Figure 3.3 shows how artificial intelligence and enhanced intelligence are related.

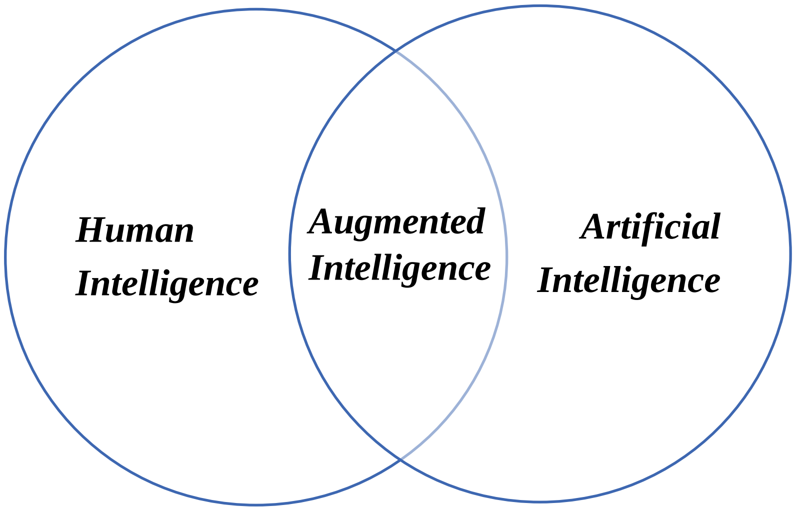

Figure 3.3: A Schema shows how artificial intelligence and enhanced intelligence are related (Carroll, 2021).

3.2 Reasoning in Artificial intelligence

Artificial Intelligence (AI) has come a long way in recent years, from simple rule-based systems to more advanced deep learning techniques. However, while AI systems can perform tasks such as image and speech recognition with remarkable accuracy, they still struggle with tasks that require reasoning and common sense. Reasoning is a central aspect of human intelligence, enabling us to make inferences, understand cause-and-effect relationships, and solve problems. AI systems that can reason would be more capable of understanding the world around them and making decisions based on that understanding.

3.2.1 Definition of Reasoning

Reasoning in Artificial Intelligence refers to the process by which a computer system uses logic and inference to draw conclusions based on data, information, or knowledge. It involves taking existing knowledge or information, applying rules or algorithms to deduce new information or make decisions, and verifying the accuracy of the results (Khemlani, 2018). Reasoning can be classified into different types, such as deductive reasoning, inductive reasoning, abductive reasoning, and probabilistic reasoning, depending on the methods and strategies used (Reid & Knipping, 2010). The goal of reasoning in AI is to enable computer systems to perform complex tasks and make decisions that involve human-like problem solving and decision making.

3.2.2 Why Reasoning?

AI reasoning is crucial for many reasons (Thomson, 2009), some of which we will list here.

- ***Problem Solving:*** Reasoning plays a critical role in problem solving for AI systems. By combining and processing information, AI systems can draw conclusions and make decisions that help them solve problems more effectively. For example, an AI system that can reason can identify the most efficient solution to a problem, or identify the root cause of a problem and generate a solution accordingly.
- ***Understanding Context:*** Reasoning enables AI systems to understand the context in which information is presented, and to use that understanding to make informed decisions. This is particularly important in real-world scenarios, where information is often incomplete or uncertain. For example, an AI system that can reason can understand the context in which a customer is making a request, and provide a more personalized response.
- ***Human-like Intelligence:*** Reasoning is a central aspect of human intelligence, and developing AI systems that can reason in a way that is similar to human reasoning would greatly enhance their ability to understand the world around them and make informed decisions. For example, an AI system that can reason can understand the cause-and-effect relationships between events, and make predictions based on that understanding.
- ***Improved Accuracy:*** Reasoning enables AI systems to make more accurate decisions, by combining information from multiple sources and considering multiple factors. For example, an AI system that can reason can consider multiple variables when making a prediction, leading to more accurate results.
- ***Explanations and Justifications:*** Reasoning can also provide explanations and justifications for the decisions made by AI systems. This is important for building trust and accountability in AI systems, and for ensuring that the decisions made by AI systems are transparent and understandable. For example, an AI system that can reason can explain why it made a particular decision, and provide evidence to support that decision.

Reasoning is an essential aspect of AI that enables AI systems to solve problems, understand context, achieve human-like intelligence, improve accuracy, and provide explanations and justifications for their decisions. The ability to reason effectively is a critical aspect of making AI systems more capable and trustworthy, and is a central focus of ongoing research in the field of AI.

3.2.3 Knowledge representation

Typical problem solving is frequently boiled down to:

- ***Knowledge Representation:*** Knowledge representation is the process of encoding information in a structured manner so that AI systems can reason about it. This involves representing information about the world in a formal language that the AI system can understand. The choice of knowledge representation depends on the particular problem that the AI system is solving, and the type of reasoning that it is performing. For example, an AI system solving a planning problem might use a representation based on state transition diagrams, while an AI system solving a natural language understanding problem might use a representation based on semantic networks.
- ***Search:*** Search is the process of finding a solution to a problem by exploring a space of possible solutions. Search algorithms are used to find the best solution to a problem based on a specific set of criteria. For example, an AI system solving a planning problem might use a search algorithm to find the optimal sequence of actions to achieve a goal, while an AI system solving a satisfiability problem might use a search algorithm to find a solution that satisfies a set of constraints.

Together, knowledge representation and search form the core of reasoning in AI, and allow AI systems to solve problems and make informed decisions. The combination of a well-designed knowledge representation and a suitable search algorithm is crucial for enabling AI systems to reason effectively and achieve high levels of accuracy. We will examine knowledge representation first, followed by how we use knowledge in reasoning, and finally, examples of these themes in use.

3.2.3.1 ***Knowledge Representation Specifications***

Knowledge representation is a critical component of reasoning in AI, as it provides the foundation for the AI system to reason about the world and make informed decisions. Here are some further specifications regarding knowledge representation:

- ***Formalism***: Knowledge representation involves encoding information about the world in a formal language that the AI system can understand. This formal language is used to represent information about objects, relationships, and rules in the world, and can take the form of a database, ontology, or other structured representation.

- ***Representing Relationships***: Knowledge representation must be capable of representing relationships between objects and entities in the world. This includes representing the relationships between objects, as well as representing the relationships between attributes of those objects. For example, an AI system might represent the relationship between a person and their address, or the relationship between a city and its state.
- ***Representing Rules:*** Knowledge representation must also be capable of representing rules and constraints that apply to the world. This includes representing rules about how objects and entities in the world behave, as well as constraints on the relationships between those objects and entities. For example, an AI system might represent the rule that all people must have a unique social security number, or the constraint that a city can only belong to one state.
- ***Reasoning with Uncertainty***: In many real-world scenarios, information is uncertain or incomplete. Knowledge representation must be capable of representing this uncertainty and incorporating it into the reasoning process. For example, an AI system might represent the uncertainty of a particular piece of information, or use probabilistic methods to reason about the likelihood of different outcomes.
- ***Compatibility with Other AI Systems***: Knowledge representation must be compatible with other AI systems, to allow for easy integration and collaboration. This includes compatibility with other knowledge representations, as well as compatibility with other AI algorithms and systems.

3.2.3.2 Search

Search is a critical component of reasoning in AI, as it enables the AI system to find solutions to problems by exploring a space of possible solutions. Here are some more details about search:

- ***Search Space:*** The search space is the set of all possible solutions to a problem. Search algorithms are used to explore this space, and find the best solution based on a specific set of criteria. The size and complexity of the search space can vary widely, depending on the problem being solved.

- ***Search Algorithms:*** Search algorithms are used to find solutions to problems by exploring the search space. Different search algorithms are suited to different types of problems and search spaces. For example, an AI system solving a planning problem might use a breadth-first search algorithm, while an AI system solving a satisfiability problem might use a depth-first search algorithm.
- ***Heuristics:*** Heuristics are used to guide the search process, and help the AI system to find solutions more efficiently. Heuristics can be used to prioritize certain parts of the search space, or to estimate the likelihood of different solutions. The choice of heuristics depends on the particular problem being solved, and the type of search algorithm being used.
- ***Optimization:*** The goal of search is often to find the best solution to a problem, based on a specific set of criteria. The search algorithm must be designed to optimize the solution based on these criteria, and find the best solution in the shortest amount of time. The choice of optimization method depends on the particular problem being solved, and the type of search algorithm being used.
- ***Complexity:*** The complexity of search algorithms can vary widely, depending on the size and complexity of the search space, the type of search algorithm being used, and the optimization method being used. The complexity of search algorithms is often expressed in terms of the number of nodes in the search space that need to be explored, and the amount of time required to find a solution.

3.2.4 How can we reason?

This will, in part, depend on the knowledge representation that is selected. However, an effective knowledge representation strategy must support simple, convincing, and natural reasoning. Here are some extremely general examples of how we can reason (Reid & Knipping, 2010).

3.2.4.1 Deductive Reasoning:

Deductive reasoning is a type of reasoning that involves drawing conclusions based on premises that are known to be true. In deductive reasoning, the conclusion must logically follow from the premises, and if the premises are true, then the conclusion must also be true.

Deductive reasoning can be formalized using logical systems, such as propositional logic or first-order logic. These logical systems allow us to represent the premises and conclusions in a structured and unambiguous way, and to apply rules of inference to draw conclusions based on the premises.

In AI, deductive reasoning is often used for tasks such as theorem proving, knowledge representation, and natural language understanding. For example, in theorem proving, the goal is to prove that a given conclusion follows logically from a set of premises. In knowledge representation, the goal is to represent knowledge in a logical form that can be used for deductive reasoning. In natural language understanding, the goal is to translate natural language statements into a logical form that can be used for deductive reasoning.

Deductive reasoning has many advantages over other forms of reasoning. For example, deductive reasoning is highly precise and allows us to draw precise and well-defined conclusions. It also allows us to reason about complex problems by breaking them down into smaller, more manageable sub-problems.

However, deductive reasoning also has its limitations. For example, deductive reasoning relies on the premises being true, and if the premises are false, then the conclusion may also be false. Additionally, deductive reasoning can only draw conclusions that are logically implied by the premises, and it may not be possible to use deductive reasoning to reach a conclusion if the premises do not contain enough information.

Example

- ***Premises:***

All birds have feathers.

Penguins are birds.

- ***Evidence:***

The premises are logically true and have been established as such.

- ***Conclusion***:

Based on the premises and evidence, it logically follows that penguins have feathers.

In this example, the conclusion is logically certain as it directly follows from the premises, which are both true. Deductive reasoning is a powerful tool for solving problems, as it allows us to draw logically certain conclusions based on premises that are known to be true.

3.2.4.2 Inductive Reasoning

Inductive reasoning is a type of reasoning that involves drawing conclusions based on observations or evidence. Unlike deductive reasoning, where the conclusion must logically follow from the premises, in inductive reasoning, the conclusion is drawn based on the likelihood that it is true given the evidence.

In AI, inductive reasoning is often used for tasks such as pattern recognition, decision making, and probabilistic inference. For example, in pattern recognition, the goal is to identify patterns in data based on the observations. In decision making, the goal is to make decisions based on the best available evidence. In probabilistic inference, the goal is to estimate the probability of events based on the available evidence.

Inductive reasoning has several advantages over deductive reasoning. For example, inductive reasoning can be used to make predictions based on limited data or incomplete information. It can also be used to identify causal relationships between variables.

However, inductive reasoning also has its limitations. For example, inductive reasoning is inherently uncertain, and the conclusions drawn from inductive reasoning may be wrong. Additionally, inductive reasoning can be influenced by biases or by the way the data is collected or interpreted.

Inductive reasoning is a type of reasoning that involves drawing conclusions based on observations or evidence. Inductive reasoning is a useful tool for solving problems in AI, as it allows us to make predictions and draw conclusions based on limited or incomplete data. However, it is important to understand the limitations of inductive reasoning in order to use it effectively.

Example:

- ***Observations:***

A person notices that every time they have eaten at a certain restaurant, the food has been delicious and the service has been excellent.

- ***Evidence:***

The person has eaten at the restaurant several times and has always had a positive experience.

- ***Conclusion:***

Based on the evidence, the person inductively reasons that the restaurant is likely to have consistently good food and service.

Note that this conclusion may not hold true in all cases, but based on the observations, it is the most likely conclusion.

3.2.4.3 Analogical Reasoning

Analogical reasoning is a type of reasoning that involves comparing and relating one situation to another situation in order to make a conclusion or solve a problem. Analogical reasoning is based on the idea that two situations that are similar in some way are likely to have similar outcomes or results.

In AI, analogical reasoning is used for a variety of tasks, including problem solving, decision making, and pattern recognition. For example, in problem solving, analogical reasoning can be used to solve a new problem by relating it to a similar problem that has been solved before. In decision making, analogical reasoning can be used to make a decision by comparing the current situation to similar situations that have occurred in the past. In pattern recognition, analogical reasoning can be used to recognize patterns in data by comparing the data to known patterns.

Analogical reasoning has several advantages over other types of reasoning. For example, it allows us to make inferences based on previous experience and to transfer knowledge from one situation to another. It also allows us to make predictions based on limited data or incomplete information.

However, analogical reasoning also has its limitations. For example, analogical reasoning is inherently uncertain, and the conclusions drawn from analogical reasoning may be wrong. Additionally, analogical reasoning can be influenced by biases or by the way the situations are compared.

Example:

- ***Observations:***

A person has a problem with their car not starting.

- ***Comparison***:

The person remembers a similar problem they had with their friend's car that turned out to be a dead battery.

- ***Conclusion:***

Based on the comparison, the person analogically reasons that their car's problem might also be a dead battery.

In this example, the person is using analogical reasoning by comparing their current problem with a similar problem they encountered in the past in order to make a conclusion and solve their current problem. The comparison of the two situations allows the person to transfer their knowledge and experience from the past problem to the current problem, making it easier to diagnose and solve the issue.

3.2.4.4 Abductive reasoning

Abductive reasoning is a type of reasoning that is used to explain observations or phenomena by inferring the most likely explanation for those observations. It is a process of going from an observation to a conclusion about what might have caused the observation. Abductive reasoning is different from deductive reasoning, which is the process of deducing a conclusion from a set of premises, and inductive reasoning, which is the process of reaching a conclusion based on evidence or observations.

Here are some more specifications about abductive reasoning:

- Explanation Generation: The primary goal of abductive reasoning is to generate an explanation for a given observation. This involves considering a set of possible explanations and selecting the one that is most likely based on the available evidence.
- Hypothesis Testing: Abductive reasoning is often used in hypothesis testing, where a set of possible explanations is generated and then tested against the available evidence to determine the most likely explanation. This can involve collecting additional data or evidence to support or refute the different explanations.
- Problem Solving: Abductive reasoning can also be used in problem solving, where it is used to generate a possible solution to a problem based on the available evidence. This can involve considering a set of possible solutions and selecting the one that is most likely to solve the problem based on the available evidence.

- Inference from Evidence: Abductive reasoning involves inference from evidence, as it is used to reach a conclusion based on the available evidence. This requires considering the available evidence and making a judgement about the most likely explanation for a given observation.
- Integration with Other Reasoning Methods: Abductive reasoning can be integrated with other reasoning methods, such as deductive reasoning and inductive reasoning, to provide a more complete and accurate explanation of observations and phenomena.

Example

Observations:

Patient has a headache, fatigue, and muscle aches

Reasoning:

Use of abductive reasoning to determine the most likely explanation for the observations

Conclusion:

Patient has a viral illness such as the flu.

Explanation is based on the common symptoms of viral illnesses and is the best explanation for the patient's symptoms.

3.2.4.5 Model-based reasoning

Model-based reasoning is a type of reasoning that uses a model, or a simplified representation of a system, to make predictions, draw conclusions, and solve problems. This type of reasoning is used when the information available is insufficient to make an accurate conclusion or prediction, so a model is used to fill in the gaps.

In model-based reasoning, a model is created based on existing knowledge and observations of the system being studied. The model is then used to simulate the system and make predictions about its behavior. This allows for testing and exploration of different scenarios and helps to identify potential problems or areas of improvement.

Model-based reasoning is used in a wide range of fields, including physics, engineering, economics, and computer science. For example, in physics, a model of a system can be used to predict the behavior of objects in motion. In engineering, a model can be used to design and test new systems before they are built. In economics, a model can be used to predict changes in the market or the impact of policy changes.

The key advantage of model-based reasoning is that it allows for exploration and testing of different scenarios, which can lead to a better understanding of the system being studied and improved predictions or conclusions. However, it is important to keep in mind that the accuracy of the results obtained through model-based reasoning depends on the accuracy of the model itself.

3.2.4.6 Case-based reasoning

Case-based reasoning (CBR) is a type of artificial intelligence that uses past experiences, or cases, to solve new problems. CBR works by first storing previous cases in a database, then, when presented with a new problem, searching the database for similar cases and using the solutions to those cases to find a solution to the current problem.

CBR is commonly used in fields such as medicine, customer service, and engineering, where problems are often complex and may have multiple possible solutions. By leveraging past experiences, CBR can provide a more efficient and effective way to solve problems compared to starting from scratch every time a new problem is encountered.

Example:

A doctor is presented with a patient who has symptoms of a headache, fatigue, and muscle aches. The doctor uses case-based reasoning to find a solution. The doctor recalls a similar case from the past where the patient had a viral illness such as the flu. Based on this past experience, the doctor concludes that the current patient probably also has a viral illness and prescribes a treatment plan accordingly. This approach to problem-solving allows the doctor to quickly and effectively find a solution based on past experiences.

3.2.5 Reasoning sub-fields

3.2.5.1 Reasoning with Uncertainty

Reasoning under uncertainty refers to the process of making decisions and solving problems when there is limited or uncertain information available. It involves evaluating different options and making informed choices based on probabilities, decision theory, Bayesian reasoning, fuzzy logic, or evidence-based reasoning. The goal of reasoning under uncertainty is to find the best possible solution in uncertain situations and minimize the risks and uncertainties associated with the decision.

3.2.5.2 Nonmonotonic Reasoning

Nonmonotonic reasoning is a type of reasoning in which previous conclusions can be revised or altered in light of new information. It allows for the flexibility to adapt to changing circumstances and is used in artificial intelligence, decision making, and knowledge representation. Nonmonotonic reasoning is based on the idea that new information can change the validity of previous conclusions in a non-monotonic way, meaning that the addition of new information can sometimes result in the removal or revision of previous conclusions.

3.2.5.3 Shallow and Deep Representation of Knowledge

Shallow and deep representation of knowledge refer to the different ways in which information can be represented and processed. Shallow representation of knowledge refers to a surface-level or superficial understanding of information, where the focus is on the surface features and facts, rather than the underlying meaning or relationships. Shallow representations are typically easier to produce and can be used for quick decision making or simple tasks.

Deep representation of knowledge, on the other hand, refers to a more complete and sophisticated understanding of information, where the focus is on the underlying meaning and relationships, rather than just the surface features. Deep representations require more effort and time to produce, but can provide a more comprehensive and nuanced understanding of the information. Deep representations of knowledge can provide a more accurate and detailed understanding of information and are often more valuable in complex tasks, problem solving, and decision making. In contrast, shallow representations are typically more appropriate for simple tasks where a quick and easy-to-process understanding of the information is sufficient.

3.2.5.4 Semantic Networks

Semantic networks are used in artificial intelligence, natural language processing, and other related fields to represent and process knowledge in a more structured and organized manner. The use of semantic networks allows for knowledge to be easily reused, combined, and updated, making it a powerful tool for representing and reasoning about complex information.

Overall, semantic networks provide a flexible and intuitive way to represent knowledge, making it easier to analyze, understand, and make decisions based on that knowledge.

3.2.5.5 Blackboard approach

The Blackboard approach can be used to implement a system that can solve problems by combining information from multiple sources. The knowledge sources in the system are responsible for acquiring, representing, and reasoning about different aspects of the problem, and they communicate with each other through the blackboard. This allows the system to perform multiple reasoning processes simultaneously and make decisions based on the information available on the blackboard. The Blackboard approach is often used in complex reasoning tasks, such as natural language processing, expert systems, and decision-making systems, where multiple sources of information must be integrated and analyzed to solve a problem.

3.2.5.6 Inheritance Methods

Inheritance methods can be used to capture commonality and specificity in a domain of knowledge, and to allow for efficient and flexible representation of new knowledge. For example, if a class "Animal" has attributes such as "species" and "num_legs," classes for specific animals, such as "Dog" or "Cat," can inherit these attributes and add their own unique attributes, such as "breed" or "color." This allows the system to reason about specific animals in a more efficient and organized manner, by leveraging knowledge from the more general "Animal" class

3.2.5.7 Pattern Matching

Pattern matching is a technique used in Artificial Intelligence (AI) and computer science to search for patterns or structures in data. It involves comparing an input pattern to a set of known patterns, and determining the closest match. Pattern matching is used in a variety of AI applications, including natural language processing, computer vision, and expert systems.

3.2.6 Which Reasoning Process to Use

The fundamental strength of an expert system lies in the efficient control of the reasoning process through the use of the proper knowledge base and inference engine components. There are several paradigms for managing the inferencing process, and each has benefits and drawbacks. The most common concepts are forward chaining and backward chaining. In many situations, conflict resolution is also crucial. To regulate the rule execution, some systems offer employing a rule-scheduling method, and others offer syntactic tools.

3.2.7 *Reasoning main uses*

- ***Problem solving:*** Reasoning is used to solve problems by combining knowledge from multiple sources and making inferences based on that knowledge.
- ***Decision making:*** Reasoning is used in decision-making systems to analyze data, identify patterns, and make informed decisions based on that analysis.
- ***Knowledge representation:*** Reasoning is used in knowledge representation to model and reason about the relationships between concepts and to represent knowledge in a form that can be processed by a computer.
- ***Natural language processing:*** Reasoning is used in natural language processing to interpret and understand human language, and to generate coherent and meaningful responses.
- ***Expert systems:*** Reasoning is used in expert systems to automate the decision-making process in complex domains, such as medical diagnosis or legal analysis.
- ***Planning and scheduling:*** Reasoning is used in planning and scheduling systems to make decisions about the most efficient way to allocate resources and achieve goals.

These are some of the main uses of reasoning in AI, but there are many other applications as well. Reasoning plays a key role in enabling AI systems to make intelligent decisions and to solve complex problems in a variety of domains.

3.3 Conclusion

In this chapter, we talked generally about the basic principles and concepts of artificial intelligence; we have also outlined the principle of artificial intelligence and quoted some definitions to be able to draw the main lines of this new discipline. We also talked about reasoning in artificial intelligence using examples and we cited the types of this reasoning. At the end, we have raised the main uses of reasoning in artificial intelligence.

For the next chapter, we will focus on the tools or algorithms essential to the design of an intelligent system.

Chapter 4

BDI Logic and Temporal Logic

Summary

Chapter 4 : BDI Logic and Temporal Logic

4.1 Introduction

Logic served philosophy and theology until the 19th century. It appeared brutally and crucially at the turn of the 20th century in mathematics, with paradoxes and the question of foundations. After the Gödel theorem (Gödel, 1931) and the bankruptcy of Hilbert's program, mathematical logic became a specialized part of pure mathematics. However, the golden age of logic then comes with the development of computing.

The introduction of computers has led to the need for a thorough organization of the issues to be addressed. Logic has become a vital component in programming and verification. There exists a remarkable correlation between evidence and programs, which makes logic indispensable in comprehending calculations. In addition, logic has paved the way for several technological breakthroughs, including query languages in databases. Numerous other vital links can be identified, including circuits, complexity, games, linguistics, among others. Logic is an essential part of computer science, present everywhere.

This chapter will present the bases of computer logic: this first part will deal with BDI logic, and the second part will deal with temporal logic. In both parts, we will introduce several interpretations of the formulas, and several formal proof systems.

4.2 BDI logic

In this section, we define a propositional modal logic extending "classic" BDI logic. We consider the modal operators of belief B, goal (or desire) D as well as intention operator I. This approach to BDI logic is inspired by the work of Rao and Georgeff (Rao & Georgeff, 1995). An operator C (α, Ø) is introduced. It means that the agent can perform the activity α to reach the goal Ø. Moreover, we are in a mono-agent framework.

4.2.1 Terms

4.2.1.1 The B = Belief

Beliefs in the BDI model refer to the internal representation of an agent's knowledge about the world. Beliefs are used to model the agent's current understanding of its environment and can be updated as new information becomes available. In the context of the BDI model, beliefs represent facts that the agent believes to be true.

For example, a belief of a thermostat agent might be "the current temperature in the room is 25°C." This belief is based on data received from a temperature sensor and is updated as the temperature changes. The agent uses its beliefs to make decisions about what actions to take. For example, if the agent believes that the room temperature is too low, it might intend to turn on the heating.

4.2.1.2 The D = Desire

Desires in the BDI model refer to the goals or objectives of an agent. Desires represent what the agent wants to achieve or maintain in its environment. They represent the agent's motivations and are used to guide its behavior.

For example, a thermostat agent might have a desire to maintain the temperature in a room at a comfortable level set by the user. This desire is used to guide the agent's behavior and decision-making process. Based on its beliefs about the current temperature and the desired temperature, the agent formulates a plan of action to achieve its goal.

4.2.1.3 The I = Intention

Intentions in the BDI model refer to the plans or actions that an agent intends to take to achieve its desires. They represent the agent's decision-making process and provide a concrete plan for achieving its goals.

Intentions are based on the agent's beliefs about the world and its desires. Based on this information, the agent formulates a plan of action to achieve its goals. For example, a thermostat agent may intend to turn on the heating if its belief is that the room temperature is below the desired temperature set by the user.

4.2.2 Syntax

A language L is introduced. It contains a set of atomic propositions P, a set of actions ACT as well as a set of primitive activities A. A formula well-formed Ø of L respects the following grammar:

$$Ø ::= p \mid \neg Ø \mid Ø \rightarrow Ø \mid BØ \mid DØ \mid IØ \mid C(\alpha, Ø)$$

$$a ::= e \mid a; a$$

Note that p∈ P, e∈ ACT, and α∈A. Logical connectors ∧, and ∨ are supported accounts and are defined in the usual way.

4.2.3 Semantics

Traditional semantic models for modal logics are based on model structures based on Kripke's semantic frameworks. A Kripke semantic frame consists of a set of states (or possible worlds) W and a set of binary relations R on W (i.e., W×W). We propose a form derived from these frameworks. We consider semantic frames composed of:

–a set C of contexts. A context c is a tuple (W_c, R_c) where:

– W_c is a set of states (or possible worlds)

– R_c is a set of binary relations on W_c

– a set R_{ic} of "inter-context" binary relations of the form $W_c \times W_{c'}$ (where c ≠ c').

Note that these frames appear as a restricted form of the frames classics (W,R) where:

– $W = U_{c \in C} W_C$

– $R = R_{ic} U(U_{c \in C} R_c)$

The "mental reality" of the agent is represented by a particular c_{RM} context. Each activity α∈A is associated with a context denoted c_α. These contexts represent the realization of the activities. In particular, the set of states Wc_α of each context cα se consists of a set of initial states $S_{c\alpha}$ as well as a set of final states $F_{c\alpha}$.

In other words, we assume that the realization of an activity starts from a state s $\in S_{c\alpha}$ and ends in a state f $\in F_{c\alpha}$.

The set of accessibility relations R for each context c∈ C consists of:

– a belief accessibility relation B_c,

– a goal reachability relation D_c,

– an intention accessibility relation I_c,

– an event accessibility relation $R_c(a)$ for each action a∈ ACT.

We assume that each relation R_c (a) is constrained under the assumptions of linear past, from branched future and determinism. Moreover, for any action sequence a1, a2, we denote by misuse of language R_c (a1; a2). If we have R_c (a1; a2) (w, w''), this is equivalent to saying that there exists w' such that such that R_c (a1)(w, w') and R_c (a2)(w', w'').

The R_{ic} set of inter-context accessibility relations corresponds in our approach to a set of accessibility relations R(α) for each activity α∈A.

These relations take the form $W_{cRM} \times S_{c\alpha}$ reflecting the possibility of accessing some initial states of the cα contexts, from W_{cRM} states of the c_{RM} context.

These accessibility relations characterize what we have called the "projections

mental" of the agent.

Each context has its own vocabulary. A vocabulary is a subset of atomic propositions for which a semantic interpretation is possible. An atomic proposition not belonging to the vocabulary of a context c is assumed to be false in each of the states of c.

Thus, the models M that we use to interpret the language L are tuples (C, R_{ic}, V, π) where:

– C is the set of contexts composed of c_{RM} and cα contexts for each activity α ∈A,

– R_{ic} is the set of accessibility relations composed of $R_{ic}(\alpha)$ where α ∈A,

– V is the set of vocabularies $V_c \subseteq P$ for c ∈C,

– π is the set of interpretation functions π_c: $V_c \times W \rightarrow$ Bool for c ∈C.

4.2.4 Satisfiability

We can now define the satisfiability relation |=. Let Ø be a well formula formed (wff) of L, M a model (C, R, V, π), wc belonging to the context c of C. We note $M, w_c \models Ø$ to mean that the world w_c satisfies (or makes true) the wff Ø. She is contextually satisfied iff $M, w_c \models Ø$ for all $w_c \in W_c$. It is valid iff $M, w \models Ø$ for any pair (M, w).

The satisfiability relation |= is defined recursively as follows:

– $M, w_c \models p$ iff $p \in V_c$ and $\pi_c(p, w_c)$ = true.

– $M, w_c \models \neg Ø$ iff $M, w_c \not\models Ø$

– $M, w_c \models Ø \wedge \psi$ iff $M, w_c \models Ø$ and $M, w_c \models \psi$

– $M, w_c \models BØ$ iff for all w'_c and $B_c(w_c, w'_c)$, we have $M, w'_c \models Ø$

– M, w_c|= DØ iff for all w'_c and $D_c(w_c, w'_c)$, we have M, w'_c |=Ø

– M,w_c|= IØ iff for all w'_c and $I_c(w_c, w'_c)$, we have M, w'_c |=Ø

Before giving the semantic interpretation of the capability operator C, we introduce the notion of strategy. An activity strategy, denoted σ(a,s,f), for an activity α associated with a context c_α is a primitive action or a sequence of actions a, such that $R(a)_\alpha(s,f)$ where $s \in S_{c\alpha}$ and $f \in F_{c\alpha}$.

In other words, an activity strategy is a particular action starting from an initial state of the activity and reaching one of its final states.

The capability operator C is interpreted semantically as follows:

– M,w_c|= C(α,Ø) iff

– for all $s_{c\alpha} \in S_{c\alpha}$ and $R_{ic}(\alpha)$ (w_c, $s_{c\alpha}$) and

– for any strategy σ (a, $s_{c\alpha}$, $f_{c\alpha}$), we have M, $f_{c\alpha}$ |= Ø.

Note that this formula can only possibly be satisfied for w_c where $c{=}c_{RM}$. That is, we assume that the capacity is only satisfiable in the "real" mental states of the agent.

Let us now focus on the semantic interpretation of this operator. Informally, this operator means that the agent mentally projects itself into an initial state particular of activity α. From this state, all the (projected) strategies of the agent reach a state in which Ø is true. Our capacity modal operator is similar to a necessity operator. The necessity is expressed by the access to the initial states of the activity. In addition, the need relates to the final states of the activity and to the fact that a property must be true in each of these states. The necessary character capacity is criticized in some approaches. The argument most often put forward is that considering total success is an unrealistic assumption. In our approach, the set of final states $F_{c\alpha}$ for an activity α must be understood as the state's success of the activity. Therefore, requiring a property to be true in all states of success is an acceptable assumption.

4.2.5 Axioms

The capacity operator we propose is a normal modal operator. The properties of our operator characterize a modal logic KD (K is the axiom of reflexivity, which states that whatever is necessarily true is also true. Symbolically, it is represented as $\Box(p \rightarrow p)$ in the other hand, D is the axiom of seriality, which asserts that if a proposition is possibly true, then it is necessarily possible. Symbolically, it is represented as $\Diamond p \rightarrow \Box\Diamond p$). In particular, the operator is closed under logical implication, conjunction and disjunction. He is by elsewhere consistent. Operator consistency means, for example, that being able to carry out an activity that makes me rich implies the fact of not being able to carry out this same activity with the aim of being poor. Thus, the capacity operator thus admits the following axioms:

$$C\ (\alpha, \emptyset) \wedge C\ (\alpha, \emptyset \Rightarrow \psi) \Rightarrow C\ (\alpha, \psi) \qquad \textbf{(K)}$$

$$C\ (\alpha, \emptyset) \Rightarrow \neg C\ (\alpha, \neg\emptyset) \qquad \textbf{(D)}$$

$$C\ (\alpha, \emptyset \wedge \psi) \Leftrightarrow C\ (\alpha, \emptyset) \wedge C\ (\alpha, \psi)$$

$$C\ (\alpha, \emptyset \vee \psi) \Leftrightarrow C\ (\alpha, \emptyset) \vee C\ (\alpha, \psi)$$

In particular, the validity of axiom D is due to the nature of the accessibility relations $R_{ic}(\alpha)$ and $R_c(a)$. We impose, indeed, that $R_{ic}(\alpha)$ be serial, that is to say, for any w_{cRM} state, there is a $s_{c\alpha}$ state such that $R(\alpha)$ (w_{cRM} , $s_{c\alpha}$). Furthermore, the states reached (through the accessibility relations $R_c(a)$) from an initial state scα are, by nature, consistent states. (Karl Devooght, 2007)

4.2.6 The BDI logic model

The intentions of an agent are the desires that the agent has decided to accomplish or the actions that it has decided to do to fulfil his desires. Even if all agent's desires are consistent, the agent may not be able to accomplish all his desires at once.

The following example will bring more clarity to this model. Agent Pierre has the belief that if someone spends his or her time studying, that person can do a doctoral thesis. In addition, Pierre has the desire to travel a lot, to do a doctoral thesis and to obtain a university assistant position. The desire to travel a lot is not consistent with the two others and Pierre, after reflection, decides to choose, among these inconsistent desires, the two last. As he realizes that he cannot achieve his two desires at the same time, he decides to first do a doctoral thesis. Now Pierre intends to do a thesis and, normally, he will use all his means to achieve this. It would be irrational of Pierre, once his decision has been made, to use his time and his energy, in particular his means, to travel around the world. By setting these intentions, Peter has fewer choices to consider. because he has given up going around travel agencies to find the travel offer that suits him. ould best satisfy.

It is this very idea that is at the heart of the BDI theory of rational action, proposed for the first time by Michael Bratman (Bratman, 1987). It is a theory of practical reasoning which tries to surprise how people reason in everyday life, deciding, at every moment, what they must do. In developing his theory, Bratman shows that intentions play a fundamental role in practical reasoning because they limit the choices possibilities that a human (or an artificial agent) can do at a certain time.

A BDI architecture is then a good candidate to model the behavior of an agent smart because:

- It is based on a known and appreciated theory of rational human action.
- The theory has been formalized in a formal, rigorous symbolic logic.
- It has been implemented and used successfully in many applications.

The "classic" agent systems that have implemented the BDI architecture are:

- IRMA = Intelligent Resource-bounded Machine Architecture, and
- PRS = Procedural Reasoning System

The PRS prototype, for example, developed at Stanford Research Center, was then used in the OASIS air traffic control system at Sydney Airport in Australia, in the SPOC business management system and it has become a commercial product of the Agentis Solutions company(Politechnica University of Bucharest, 2002).

These two systems are the best known, but many other multi-agent systems and applications are made in a BDI design, Figure 4.1 bellow shows the main components of a BDI architecture.

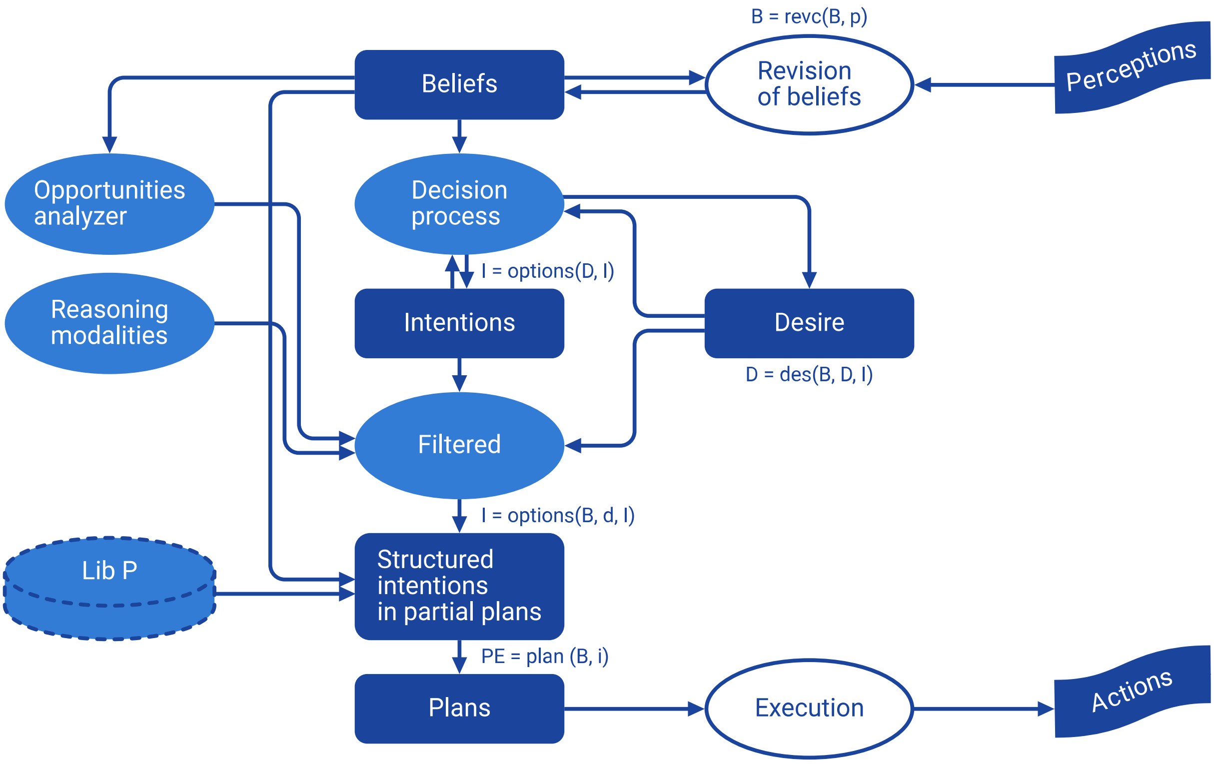

Figure 4.1 : BDI logic architecture of an agent (Bratman, 1987)

The agent has an explicit representation of its beliefs, desires and intentions. We denote by B the set of beliefs of the agent, by D the set of its desires, and by I the set of its intentions, and by B, D and I the current beliefs, desires and intentions of the agent. Sets B, D and I can be represented using various representation models of knowledge, for example by using the logic of first-order predicates, a logic order, the model of production rules, or as simple structures of data.

To solve a problem, the agent is required to generate plans that include a series of actions. The standard method of representing these actions is by specifying their effects on the environment. For example, if in an environment state, the agent moves a cart from room A to room B, the representation of this action will be "move-carriage from A to B" and the effect on the environment, will contain in particular the fact that the trolley is no longer in room A and that it is in room B. The result of an action will be that considered if the environment is deterministic. Plans designed by the agent are always knowledge structures or data structures. "LibP" is a library of planes.

This component may or may not be present in the architecture. If it exists, the process of agent's planning is simplified because he can find adequate plans at a certain situation by browsing this library of plans. In Figure 4.1, the squares represent knowledge or data structures while the ovals represent control and enforcement components.

In the following, we will consider the following functions:

- revc: B x P → B is the function of revising the beliefs of the agent when it receives new perceptions about the environment, where P represents the set of perceptions agent; it is carried out by the Beliefs Review component.
- options: D x I → I is the function that represents the agent's decision process taking into account his current desires and intentions; this function is performed by the Decision Process component.
- des: B x D x I → D is the function that can change an agent's desires if his beliefs or intentions change, to maintain the consistency of the agent's desires (we assume in our model that the agent always has consistent desires); this function is also carried out by the Decision Process component.
- filter: B x D x I → I is the most important function because it decides the intentions to chase; it is performed by the Filter component

The Filter component is the part of the architecture that is responsible for building plans partial to carry out the agent's intentions, while considering new opportunities. As a result of what he perceives of his environment and his review of beliefs, the agent can detect new opportunities that favor the realization of his intentions, or which may even prevent this realization. This analysis is performed by the Opportunities Analyzer component. These new opportunities are communicated to the Filtered. The Filter constructs partial plans to arrive at the agent's intentions with the help of the Reasoning Modalities component; the latter is responsible for carrying out the action-oriented reasoning and how plans are carried out.

- plan: B x I → PE is the function that transforms partial plans into executable plans, PE being the set of these planes; it can use, for example, a library of planes, represented by the LibP module in Figure 4.1.

A plan is a sequence of actions to be executed over time. A partial plane is a plane in which not all the planning details were specified, for example we were content of a partial order of actions in time, or some actions are not entirely detailed. Let's take the following example: we know that to go on a trip, you must pack your bags and phoning a friend to say goodbye, but there is no indication of the order in which to do these two actions, nor the number of suitcases to carry. This is a partial plan. Once we have decided that we would pack two suitcases first and then call his friend, we refined the partial plan, and thus obtained a completely specified and executable plan. The reason to begin by building partial plans is that the agent may sometimes be required to change its plans based on new perceptions gathered about the environment. The partial plan approach can provide solution to this problem. Once we have an executable plan, the Execution module will execute, one after the other, the actions of this plan in the environment (Politechnica University of Bucharest, 2002)

4.3 Temporal logic

Temporal logic is a subfield of logic that has significant philosophical relevance. Its evolution during the 20th century has connections to areas such as metaphysics, language philosophy, science philosophy, logic philosophy, and theories of action. However, tense logic, also known as temporal logic, also holds intrinsic formal value. It played a key role in the advancement of modal logic and remains a crucial part of computer science theory and various applications (Thomas Miiller, 2011)

4.3.1 Motivation and Terminology

Arthur Prior is credited with starting the formal study of the logic of time in the 1950s. He referred to his logic as "tense logic" to reflect his goal of using formal systems to advance the philosophy of time by treating the tenses in natural languages, such as English, as sentence-modifying operators. Despite the development of many other formal approaches to temporal issues, tense logic remains the natural starting point.

Prior introduced four tense operators, which are traditionally written as follows:

- P (Past): It was the case that
- H (Has always been the case): It has always been the case that
- F (Future): It will be the case that
- G (Always going to be the case): It is always going to be the case that.

These operators provide a way to describe the timing of events and relationships between events. They form the foundation of tense logic and provide a starting point for reasoning about the temporal aspects of systems.

Here is an example to illustrate the use of the tense operators introduced by Arthur Prior:

Consider the sentence

- "John will arrive at the airport at 3 PM.",

We can represent this sentence in tense logic using the operator F (Future):

- F("John arrives at the airport at 3 PM")

This formula states that it will be the case that John arrives at the airport at 3 PM.

Now consider the sentence

- "John has always been on time."

We can represent this sentence in tense logic using the operator H (Has always been the case):

- H("John is on time")

This formula states that it has always been the case that John is on time.

In both of these examples, we can see how the tense operators provide a way to describe the timing of events and the relationships between events. By using tense logic, we can reason about the temporal aspects of systems and describe the behavior of systems over time.

4.3.2 Tense Logic and the History of Formal Modal Logic

Tense logic is a type of formal logic that is concerned with the analysis of temporal aspects of systems. It was introduced by Arthur Prior in the 1950s as a way to advance the philosophy of time by establishing formal systems that treated the tenses of natural languages as sentence-modifying operators.

Prior's tense logic introduced the concept of modality, or the idea that a sentence can be qualified by the attitudes of the speaker towards its truth. This concept was later extended to modal logic, where modal operators, such as "necessarily" and "possibly," were used to describe the attitudes of the speaker towards the truth of a sentence.

The development of modal logic was further influenced by the work of philosophers such as Saul Kripke and Ruth Barcan Marcus, who introduced the idea of possible worlds. Possible worlds provide a way to model alternative possibilities, and the idea of possible worlds led to the development of possible-worlds semantics for modal logic.

In possible-worlds semantics, a possible world is a complete and consistent way that the world could have been. Modal operators, such as "necessarily" and "possibly," are used to describe the relationships between the actual world and other possible worlds. For example, a sentence of the form "It is necessarily the case that P" means that P is true in all possible worlds, including the actual world.

Tense logic and modal logic are important subfields of logic with a rich history. They provide a way to analyze the temporal and modal aspects of systems and continue to play an important role in many areas of computer science and philosophy.

4.3.3 Formal models of time

There are several different formal models of time in temporal logic (E. Cerny & X. Song, 1997), each with its own syntax and semantics. Some common models include:

- ***Linear Temporal Logic (LTL):*** a formalism used to describe properties of finite or infinite sequences of states over time. LTL formulas are typically evaluated over a sequence of states, and describe properties such as whether a certain condition holds at all times, sometimes, or never.
- ***Computational Tree Logic (CTL):*** a branching-time temporal logic used to describe properties of systems with multiple possible futures. CTL formulas are typically evaluated over a tree-like structure that represents the different possible futures of a system.
- ***Modal μ-Calculus:*** a temporal logic used to reason about the behavior of concurrent and distributed systems. The modal μ-calculus is a generalization of LTL and CTL, and

provides a way to describe the behavior of systems with multiple components that can interact with each other.

- ***Temporal Epistemic Logic:*** a temporal logic used to reason about knowledge and belief over time. Temporal epistemic logic provides a way to describe the evolution of knowledge and belief in a system, and to reason about how this knowledge and belief affects the behavior of the system over time.

Each of these logics has its own syntax, semantics, and uses. For example, LTL is often used in the design and verification of reactive systems, such as communication protocols and control systems, while CTL is used in model checking, which is a technique for automatically verifying the correctness of systems. The modal μ-calculus is used in the verification of concurrent and distributed systems, while temporal epistemic logic is used in the study of multi-agent systems and in artificial intelligence.

The ontological nature and properties of time are a source of ongoing philosophical debate and inquiry. These debates are reflected in the many different formal models of time used in temporal logics.
Different philosophical positions on time lead to different models of time, each with its own assumptions, implications, and consequences. For example, some philosophers see time as instant-based, while others see it as interval-based. Some believe that time is discrete, while others believe that it is dense or continuous. There are also debates about whether time has a beginning or an end, and whether it is linear, branching, or circular.

The models of time used in temporal logic play an important role in shaping our understanding of time, and are used in a wide range of fields, from philosophy and mathematics, to computer science and physics. Ultimately, the choice of a particular model of time will depend on the specific problem or questions being addressed, we briefly describe below (TABLE 4-1) the two most fundamental categories of formal models of time, instant-based and interval-based models, together with some of their key characteristics, before turning to the formal languages of temporal logics and their semantics.

TABLE 4-1: Table comparing instant-based and interval-based models of time

Feature	*Instant-based*	*Interval-based*
View of time	Time is viewed as a collection of distinct points	Time is viewed as a continuous quantity, composed of intervals between two points
Representation of events	Events are located at specific points in time and have no duration	Events have a definite duration and are represented by an interval of time
Advantages	Easy to visualize and understand, useful for representing events that happen at a specific point in time	Useful for representing events that have a duration, better for mathematical representation and analysis of time
Disadvantages	Limited in representing events that have a duration	Complexity in representing events that occur at a specific point in time

4.3.4 Prior's basic tense logic TL

Prior's development of tense logic was largely influenced by the role of tense in everyday language, which served as his primary philosophical inspiration. What made Prior's approach unique was his treatment of propositions as inherently tense, rather than devoid of tense. To achieve this, he introduced temporal operators into the language, which were interpreted using a modal logic framework. Although Prior referred to his theory as Tense Logic, the term Temporal Logic is now more commonly used due to its broader scope.

4.3.4.1 Semantics of TL

The semantics of temporal logic (TL) is concerned with defining the meaning of temporal logic expressions and determining when such expressions are true or false. Temporal logic is a formal language for expressing statements about events and the relationships between events in time. The most common method for defining the semantics of TL is the model-theoretic approach, where the truth of a TL formula is defined relative to a model of time. A model of time is a mathematical structure that represents a particular interpretation of time, such as the real numbers, a linear order, or a graph of events.

In the model-theoretic approach, the truth of a TL formula is determined by assigning a truth-value to its variables and evaluating the formula according to the rules of the logic. The truth-value of a formula is then determined based on the truth values of its constituent subformulas and the relationships between events represented in the model.
The semantics of TL is an important aspect of the study of temporal logic, as it provides a way to formally reason about the truth of temporal statements and make inferences about the relationships between events in time. This is particularly useful in fields such as computer science, artificial intelligence, and physics, where the analysis of time-dependent systems is a central concern.

4.3.4.2 Standard translation of TL into first-order logic

The standard translation of Temporal Logic (TL) into First-Order Logic (FOL) is a technique for embedding TL expressions into the language of FOL, so that the semantics of TL can be studied and analyzed using the tools and techniques of FOL.

The standard translation of TL into FOL involves replacing each TL operator with a corresponding FOL expression. For example, the "it was the case that" operator in TL can be translated into FOL by introducing a new unary predicate symbol that is true of a time point if the corresponding proposition is true at that time point. Similarly, the "it is the case that" operator can be translated into FOL by introducing a binary relation symbol that is true of a time point and a proposition if the proposition is true at that time point.

The advantage of this translation is that it allows us to leverage the well-developed theory and techniques of FOL, including model theory, proof theory, and automated theorem proving, to study the semantics of TL and analyze the truth of TL expressions.

It's important to note that while the standard translation of TL into FOL provides a useful tool for studying the semantics of TL, it also has some limitations. For example, the expressiveness of TL may be reduced when translated into FOL, and the translation may not always preserve all of the intuitive properties of TL. Nevertheless, the standard translation of TL into FOL remains a widely used technique in the study of temporal logic.

4.4 Conclusions

In conclusion, BDI (Belief, Desire, Intent) logic and Toporel logic are two important formal systems that have been developed to model and reason about the behavior of agents and the relationships between events in time. BDI logic provides a way to represent an agent's beliefs, desires, and intentions and to reason about the actions that an agent might take based on these mental states. Toporel logic extends traditional temporal logic to capture the topological relationships between events in time, allowing for a more expressive and nuanced representation of time-dependent systems.

Both BDI logic and Toporel logic have important applications in areas such as artificial intelligence, multi-agent systems, and computer science, where the analysis and control of time-dependent systems is a central concern. The use of formal systems such as BDI logic and Toporel logic provides a way to formally reason about the behavior of agents and the relationships between events in time, allowing for a more precise and rigorous analysis of these systems. These formal systems also provide a foundation for further research and development in the field of temporal logic and its applications.

The BDI syntax, semantic, satisfiability, and axioms were also covered in this chapter, along with a comprehensive discussion of the fundamental ideas and principles of BDI logic. Aside from being discussed, temporal logic has also been referenced along with its background and goals.

For the next chapter, we will focus on the main contribution of this dissertation which is a combination of BDI logic and the temporal logic in order to design a fearful BDI agent.

PART III: CONTRIBUTIONS

Chapter 5

A Conceptual Architecture with BDI Logic and Model Checker NuSMV for Emotional-BDI Agents

Summary

Chapter 5 : A Conceptual Architecture with BDI Logic and Model Checker NuSMV for Emotional-BDI Agents

5.1 Introduction

The BDI architecture is widely recognized as one of the most extensively researched and famous architectures for software agents due to its philosophical foundations and logical frameworks. Although this architecture has been used in numerous software systems, it was limited in terms of incorporating emotions into artificial agents, also known as Emotional Agents. In order to address this limitation, a new conceptual Emotional-BDI architecture is proposed in this chapter, which aims to expand the standard BDI architecture by incorporating emotions and psychological traits. These traits can be used to model emotions artificially and implemented computationally, and can improve people's awareness of fear, allowing for easy mapping of expert knowledge to agent behavior. Furthermore, the proposed agent architecture takes into account individual variations and external effects, and is based on the OCC model's emotional expansion of the BDI model, with the addition of the emotion of fear as a new modulator of this extended model. This framework is based on research from psychology and cognitive science, specifically the OCC theory, and encompasses many of the known mechanisms that drive emotions. This chapter is split into two parts. The first part will cover the suggested method, while the second part will provide details on the tools, programming languages used, modeling of our model, specification of the scenarios, and simulation of the results

It is imperative to acknowledge that this chapter is derived from our seminal work titled "A Model Combining BDI Logic and Temporal Logics for Decision-Making in Emergencies."(Benrouba & Boudour, 2022)

5.2 Proposed approach

The thesis aims to address a problem related to how Emotional-BDI agents can effectively deal with highly dynamic and unpleasant situations in their environment. The proposed solution involves implementing a new conceptual architecture with an additional component that extends the BDI logic by incorporating the emotion of fear. This added component can assist agents in behaving appropriately during unfavorable events. This section introduces the conceptual system model and the interpreted BDI system model. Following Su et al.'s representation, the agent in this model possesses local observations.

5.2.1 The Fearful Emotional BDI System Model

This passage provides a succinct summary of the OCC theory of emotion, which seeks to encompass the BDI system model by integrating human emotion modalities, such as fear, and specific psychological traits. Following this, the authors began constructing a novel conceptual model for BDI agents that would incorporate emotion triggers, using the interpreted-based BDIE system model (referred as the BDIE model for brevity). A schema illustrating the proposed architecture is presented in Figure 5.1

The OCC Theory: developed by Ortony, Clore, and Collins (Ortony et al., 1988), is a psychological theory that explains the circumstances that lead to 22 different types of emotions, categorized into six groups. It uses a three-branch typology that corresponds to three types of stimuli: the results of events, actions taken by agents, and features of objects. Figure 5.2 provides a visual representation of this typology.

This essay places particular emphasis on the experience of well-being emotions, specifically fear. This type of emotion arises when an individual feels apprehensive or anxious about a recent event, but disregards the potential negative consequences that may result from it. In the context of agents, if an agent is concerned about a negative outcome, it will experience fear.

The model of the Interpreted BDIE System: The main goal of this BDIE model is to represent an agent's emotions, in this case, FEAR, as a collection of runs (computing pathways), which perfectly corresponds to a system in the interpreted system model.

Here we follow the introduction given in Su et al, in their contribution entitled “Observation-based model for BDI agents”, we can address the fearful BDI interpreted system.

The mental and emotional states of an agent i with respect to a system K, defined over a set G of global states, can be represented as a tuple. The tuple comprises the agent's belief, desire, and intention, as well as their emotional states of joy and distress.

$$\mathcal{M}_i =< \mathcal{B}_i, \mathcal{D}_i, \mathcal{I}_i, fear_i >$$

The system $fear_i$ consists of sets of runs over G that correspond to situations where the agent

i experiences unpleasant emotions. In line with the OCC theory, the attainment of desirable outcomes or goals is contingent upon intentional decision-making in relation to available actions.

Then, a BDIE system S is defined as a structure $S = < K, M_1, \ldots M_n >$, where K is a system and for each i, M_i is the agent's mental states (believe, desire, and intention) and emotional states (FEAR) over K.

If we have a set of ϕ basic propositions that describe fundamental facts about the agent and its environment, an interpreted BDIE system can be constructed. The interpreted BDIE system is composed of a pair of BDIE systems along with an evaluation function, which ensures that the original set of propositions holds true at all points in the set of global states, denoted by G.

5.2.2 The logical system of the BDI model for Fear emotion

5.2.2.1 BDIE Logic Syntax

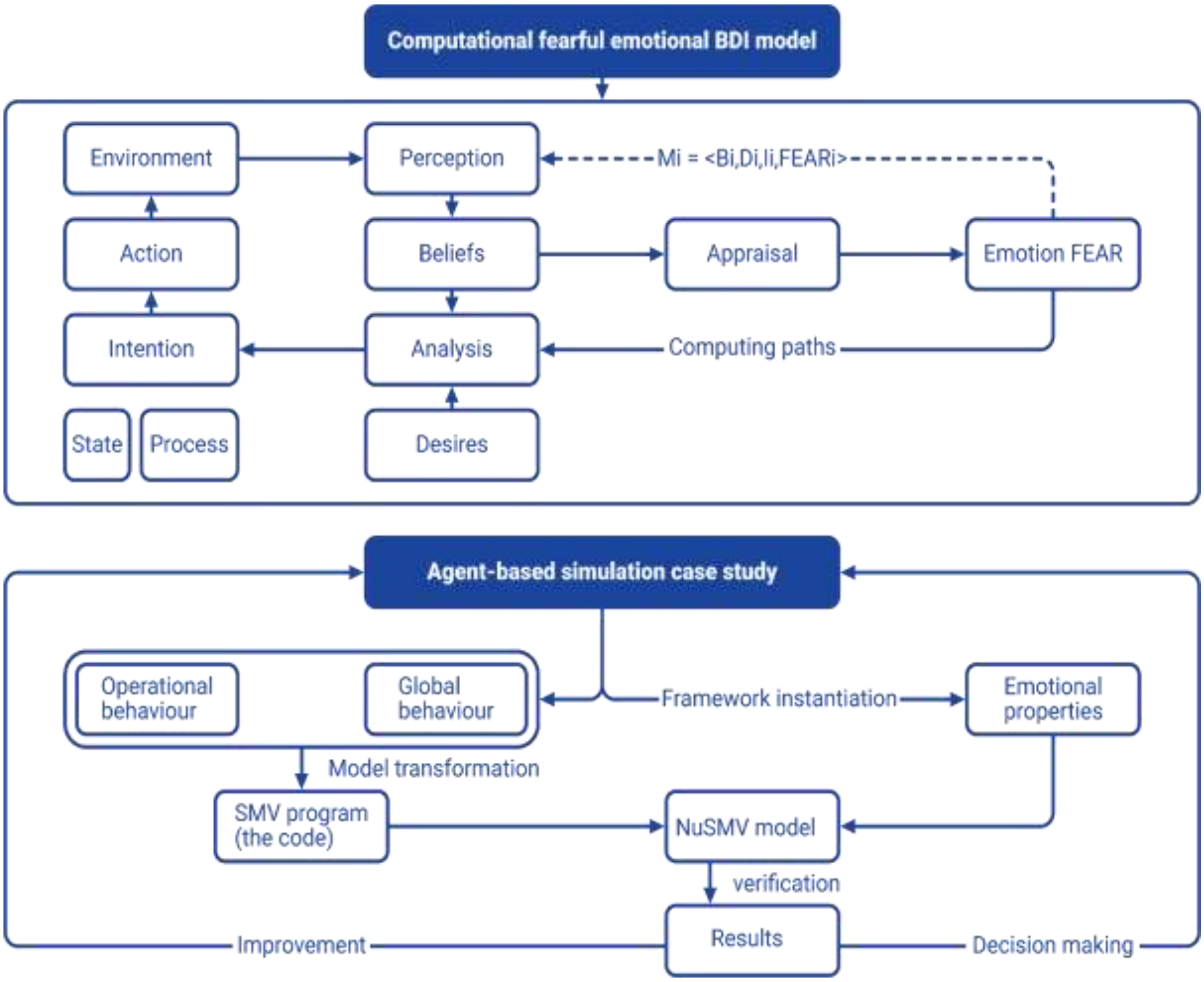

Figure 5.1: A schema of the proposed architecture

The basic building blocks of our emotional logic consist of the following syntactic components: a finite set of agents that is not empty, $A = \{1,2, \ldots, n\}$, and a non-empty finite set of atomic propositions $\phi = \{p_1, p_2, \ldots, p_m\}$. The variables i denotes the agent's number and p denotes propositional letters (propositional atoms). The following BNF (Backus Naur Form) notations shown in Eq. (5.1) define the language of BDIE logic:

$$\varphi ::= \perp |p|\neg\varphi|\varphi \wedge \varphi|\bigcirc\varphi|\varphi U\varphi|$$

$$B_i\ \varphi|D_i\ \varphi|I_i\ \varphi\ | \qquad (5.1)$$

$$Fear_i\varphi$$

Where p takes on values from over φ, i takes on values from over A. The standard method is used to define the boolean connectives ∨ (disjunction), → (material implication), ↔ (material equivalence), and ⊤ (tautology) are defined from ¬ (negation), ∧ (conjunction), and ⊥ (contradiction).

BDIE logic is the modal logic augmented with the future-time connectives ◯ (next) and **U** (until), modal operators B_i , D_i , I_i, and emotional modal operator Fear i for each agent i. Linear-time temporal logic (LTL) operators F and G can be defined as follows in Eq. (5.2)):

$$F\varphi \stackrel{\text{def}}{=} \top U\varphi \qquad (5.2)$$

$$G\varphi \stackrel{\text{def}}{=} \neg F\neg\varphi$$

In everyday language, the expression $B_i\varphi$ can be understood as "the agent i thinks or is of the opinion that φ is true". Belief is understood as subjective knowledge, alias truth in all worlds that are possible for the agent. $D_i\varphi$ Indicates, "φ is unpleasant for the agent i". In our view, every goal is about something unpleasant. Thus, if a consequence of an event is a goal, then this consequence is unpleasant. $I_i\varphi$ denotes that "φ holds under the assumption that the agent i acts based on its intention".

Fear i φ means, "The agent i feels fear for φ".

- $(I, r, u) \vDash_{BDIE} B_i\varphi$ iff $(I, r', v) \vDash_{BDIE}\varphi$ for all (r', v) such that $r' \in B_i$ and $(r, u) \sim i\ (r', v)$;
- $(I, r, u) \vDash_{BDIE} D_i\ \varphi$ iff $(I, r', v) \vDash_{BDIE}\varphi$ for all (r', v) such that $r' \in D_i$ and $(r, u) \sim i\ (r', v)$;

- $(I, r, u) \vDash_{BDIE} I_i\ \varphi$ iff $(I, r', v) \vDash_{BDIE} \varphi$ for all (r', v) such that $r' \in I_i$ and $(r, u) \sim i\ (r', v)$;

$(I, r, u) \vDash_{BDIE}\ fear_i\ \varphi$ iff $(I, r', v) \vDash_{BDIE} \varphi$ for all (r', v) such that $r' \in fear_i$ and $(r, u) \sim i\ (r', v)$;

5.2.2.2 BDIE Logic Semantics

We now proceed to interpret BDIE logic formulas in terms of the interpreted BDIE system. Given an interpreted BDIE system $I = (\mathcal{S}, \pi)$, suppose that $S = < K, M_1, \ldots, M_n >$ and for each$i, i \in \{1, \ldots, n\}$, $M_i =< B_i, D_i, D_i, fear_i, >$. Let r be a run in k and u be a natural number, in the following, we inductively define the satisfaction relation $\vDash BDIE$ between a formula φ and a pair of the interpreted BDIE system I and a point (r, u), (check TABLE 5-1).

TABLE 5-1: The satisfaction relation ⊨_BDIE holds between a formula φ and a pair consisting of an interpreted BDIE system.

BDIE system (I, r, u)	BDIE system (I, r', u)	$r' \in$
$\vDash_{BDIE} B_i\varphi$ (Belief)	$(I, r', v) \vDash_{BDIE}\varphi$	B_i and $(r, u) \sim i\ (r', v)$
$\vDash_{BDIE} D_i\varphi$ (Desire)	$(I, r', v) \vDash_{BDIE}\varphi$	D_i and $(r, u) \sim i\ (r', v)$
$\vDash_{BDIE} E_i\varphi$ (Intention)	$(I, r', v) \vDash_{BDIE}\varphi$	E_i and $(r, u) \sim i\ (r', v)$
$\vDash_{BDIE} fear_i\varphi$(the emotion: Fear)	$(I, r', v) \vDash_{BDIE}\varphi$	$fear_i$ and $(r, u) \sim i\ (r', v)$

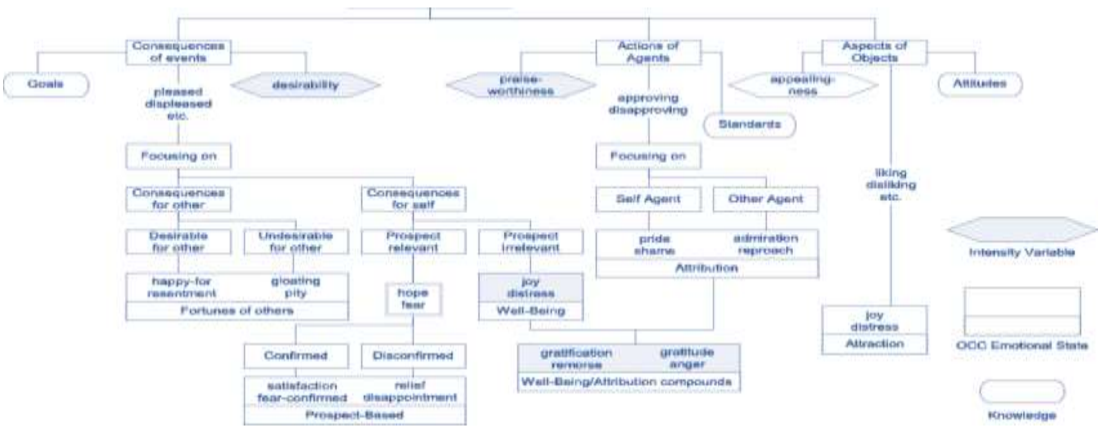

Figure 5.2: The OCC model

5.2.3 Propositions

Proposition 1: The following axioms are valid concerning both $\vDash_{\text{BDIE}}$ and $\vDash^{spr}_{BDIE}$:

- $\Delta\, i\,(\varphi \Rightarrow \psi) \Rightarrow (\Delta_{i\,\varphi} \;\Rightarrow\; \Delta_{\,i\,\psi})$, where Δ stands for B, D, I, Fear.
- Relationship between belief, and intention $B_{i\,\varphi} \;\Rightarrow\; D_{i\,\varphi}$
- Relationship between desire, intention, Fear $fear_{i\,\varphi} \;\Leftrightarrow\; D_{i\,\varphi} \;\wedge\; I_{\,i\,\varphi}$
- Temporal operators is explained in Eq. (5.3)

$$\bigcirc(\varphi \Rightarrow \psi) \Rightarrow (\bigcirc\varphi \Rightarrow \bigcirc\psi)$$

$$\bigcirc(\neg\varphi) \Rightarrow \neg\bigcirc\varphi \tag{5.3}$$

$$\varphi\mathbf{U}\psi \Leftrightarrow \psi \vee (\varphi \wedge \bigcirc(\varphi\mathbf{U}\psi))$$

Proposition 2: The following axioms are valid for $|^{spr}_{BDIE}\!: \Delta\bigcirc\varphi \;\Rightarrow\; \bigcirc\Delta\varphi$, where Δ stands for any modality of B_i , D_i , I_i , $fear_i$, The formula $D_i\bigcirc\varphi \;\Rightarrow\; \bigcirc D_i\varphi$ says that if the agent i's current goal implies φ holds at the next point in time, then at the next point in time its goal will imply φ, that is, the agent i persists on its goal.

In addition, the formula $Fear\;\bigcirc\varphi \;\Rightarrow\; \bigcirc Fear_{\varphi\, i}$ says that if the agent's current emotional state is Fear for φ holding at the next point in time, then at the next point in time its emotional state is still Fear for φ.

Moreover, the formula $Fear\;\bigcirc\varphi \Rightarrow \bigcirc F\;ear\;\;\varphi\;\;i$ says that if the agent's current emotional state is Fear for φ holding at the next point in time, then at the next point in time its emotional state is still Fear for φ.

5.3 Experimentations

In this section, we will show how to use the open-source NuSMV inspection tool and the CUDD library (a detailed explanation of which is provided in the second part of this chapter) to inspect sentiment attributes. We will also discuss some common usage scenarios for these tools.

- All the experiments were carried out on a computer with specific hardware and software specifications. The experiments were conducted on a dual-core 3.5 GHz Intel Core i5 computer with 16 GB RAM running the Windows 10 operating system. The specifications of the computer are relevant as they impact the performance and outcomes of the experiments. By providing this information, we aim to ensure the replicability of the experiments and facilitate the comparison of the results with other studies. This detail helps to better understand the computational resources required to conduct the experiments, which can be crucial in replicating and extending the findings.

5.3.1 *Case study 1: Auction*

In this section, we provide a simplified example of an auction scenario that is used to demonstrate how the BDIE model and the BDIE logic specification are constructed.

(Su et al., 2018)propose an auction scenario similar to ours, but they consider a scenario where two agents (ag1 and ag2) participate in the auction. There are four global states: s0, s1, s2, and s3. In state s0, each agent bids, and in states s1, s2, and s3, a winner will be declared. Specifically, in s1, the winner is ag1, in s2 the winner is ag2, in s3 the winner cannot be determined because ag1 and ag2 offer the same prize. Furthermore, each agent has an initial belief about what it bids, and each one desires to win.

5.3.1.1 *Framework Instantiation*

Now we define a BDIE system $S = \langle K, M_1, M_2 \rangle$, were

- K is the set of those runs r such that, for every natural number m, for each j, $j \in \{1,2,3\}$, if $r(m) = S_j$, then $r(m+1) = S_j$, which means that if the winner is announced, then it will keep so.
- $M_i = < B_i, D_i, I_i, Fear_i, Not\ Fear_i >$, for each $i, i \in \{1,2\}$.
- B_i is the set of those runs $r_i \in K$ $r_i(O) = s_0$. This implies that each agent has a belief about the maximum amount they can bid, but it does not necessarily guarantee that the agent will win the auction. Belief, in this context, simply refers to the information state of the agent.
- D_i is a subset of B_i such that, for each run, $r_i \in D_i$ there is a number m with $r_i(m) = s_i$ This indicates that each agent has a desire or goal to win the auction.
- I_i is a subset of K such that, for each run $r_i \in I_i$ and every natural number m,if $r_i(m) = s_0$, then $r_i(m+l) = s_j (j \in \{1,2,3\})$. This indicates that each agent will bid immediately.

- $Not\ Fear_i = D_i \cap I_i$, that is, $Not\ Fear_i$ is the set of runs r_i such that for every natural number m, if $r_i\ (m) = s_0$, then $r_i\ (m + 1) = s_i$. This means that the agent i doesn't feel fear if it wins the auction after bidding.
- $Fear_i =\sim D_i \cap I_i$, that is, $Fear_i$ is the set of runs r_i such that for every natural number m, if $r_i\ (m) = s_0$, then $r_i(m + l) = s_j\ (j \in \{1,2,3\}\ and\ j \neq i)$.. This means that the agent i feels Fear if it does not win the auction after bidding.

Now we can formulate these emotional attributes in the auction scenario to make our decision cycle for the two agents:

- F (NOT Fear $ag1$ (winner 1)) indicates that eventually, the agent 1 will be in a state of not fear for winning the auction.
- **F** (Fear $ag2$ (¬ winner 2)) indicates that eventually, the agent $ag2$ will be in fear of losing the auction.

5.3.1.2 Emotional Properties Verification

Our formal description of emotional triggers enables each agent to manifest emotions. To ensure the accuracy of these emotional attributes within the auction scenario, we utilized the NuSMV checker model tool. Specifically, we employed the NuSMV tool in conjunction with CUDD to validate the following two specifications.

TABLE 5-2: The results of the auction scenario when the state is S1 after 10 rounds

Number of rounds	**State**	**Number of rounds**	**State**	**Number of rounds**	**State**
1st round (S1)	S1=NOT FEAR	2nd round	S1=NOT FEAR	3rd round	S1=NOT FEAR
4th round	S1=NOT FEAR	5th round	S1=NOT FEAR	6th round	S1=NOT FEAR
7th round	S1=NOT FEAR	8th round	S1=NOT FEAR	9th round	S1=NOT FEAR
10th round	S1=NOT FEAR				

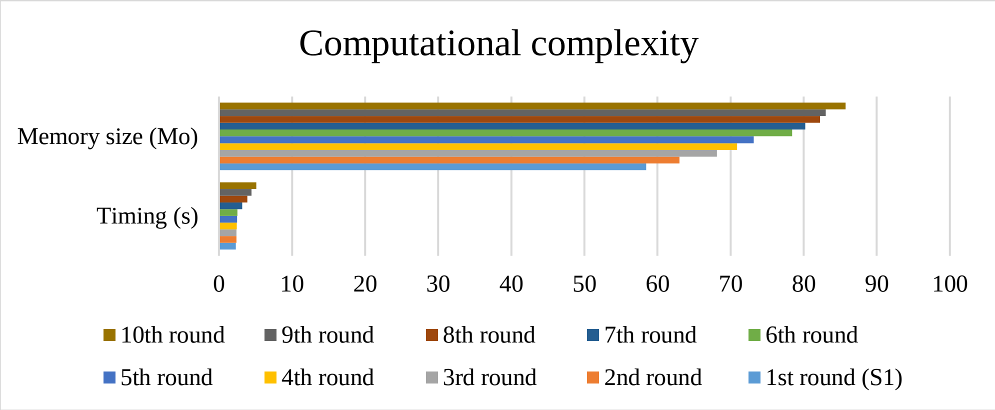

Figure 5.3: Bar chart showing the computational complexity during executing the auction scenario when the state is S1 after 10 rounds

TABLE 5-3: The results of the auction scenario when the state is S2 after 10 rounds

Number of rounds	**State**	**Number of rounds**	**State**	**Number of rounds**	**State**
1st round (S2)	S2= FEAR	2nd round	S2= FEAR	3rd round	S2= FEAR
4th round	S2= FEAR	5th round	S2= FEAR	6th round	S2= FEAR
7th round	S2= FEAR	8th round	S2= FEAR	9th round	S2= FEAR
10th round	S2= FEAR				

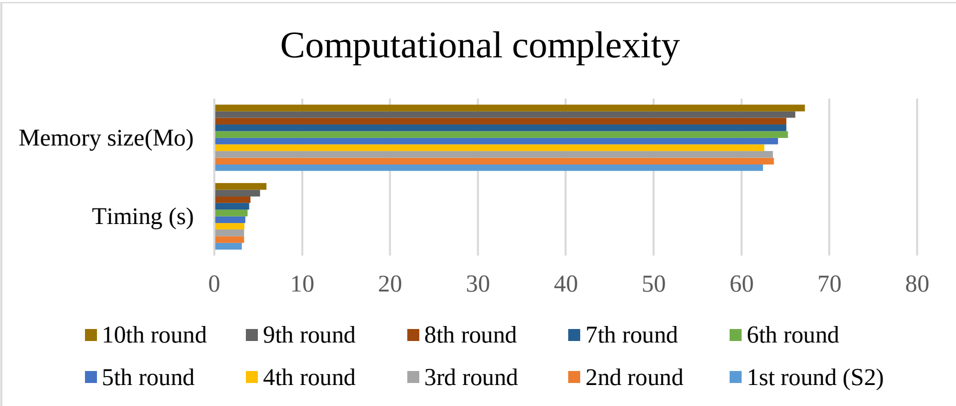

Figure 5.4: Bar chart showing the computational complexity during executing the auction scenario when the state is S2 after 10 rounds

In the first specification after each agent bids, the ag1 wins and does not feel fear, TABLE 5-2 above shows the verified properties after 10 rounds of ag1, the computation of complexity needed for running this specification is shown in Figure 5.3.

As for the second specification, after each agent bids, the ag2 loose and will eventually feel fear, TABLE 5-3 above shows the verified properties after 10 rounds of ag2, the computational complexity needed for running this specification is shown in Figure 5.4.

In the study conducted by (Su et al., 2018), they presented a case study on auctions, focusing on the emotion modalities of Joy and distress. In contrast, our approach takes a different perspective. In our scenario, each agent begins with the expectation of winning in every auction round. For instance, Agent 1, when it wins, feels content and does not experience anxiety because the desired outcome of winning is achieved through bidding. On the other hand, Agent 2 experiences Fear as it anticipates losing the game, which is an undesirable outcome, leading it to decide against participating further.

The experimental results in TABLE 5-4 verify these formulas (ag1 & ag2). The following can be observed: The results of the above two emotional specifications are all true and consistent with the OCC emotional theory.

TABLE 5-4: Verification results

ag_i	Results	Timing(s)	BDD nodes
ag1=F(agent 1 Not fear (winner=agent1))	TRUE	**2.32193**	38458
ag2=F(agent 2 Fear (looser = agent2))	TRUE	**3.12243**	38458

5.3.2 Case Study 2: Simulation System for Aircraft Maintenance Based on Fear of the Agent

The specification of this case study is to formalise the operational and control behavior of some random flight agent and focus on its own emotion in our case "FEAR" from our BDIE architecture. Control behavior expresses the general behavior of any process related to aircraft maintenance.

In this section, we define the global behavior and operational behavior of the agent. The global behavior encompasses the emotional behaviors that occur throughout the life cycle of planes, from a global perspective. On the other hand, the operational behavior describes the internal behavior of a component, focusing on a specific emotion, in this case, FEAR.

5.3.2.1 Presentation of Simulation System for Aircraft Maintenance Based on Fear of the agent

We can now address our fearful agent-based system to reason about the knowledge and temporal properties of the plane and flight agents. The formalism is illustrated as follows (see Figure 5.5): The agent will do a mapping between state-to-state (S-S) and transition-to-transition (TT) from the global behaviors to the control behaviors based on the emotion of fear, firstly it will initialise the connection with flight agent by invoking the plane ⟶, Then it will start plane checking to make sure that everything is in a good position and safe ⟶, if the time of checking is too long, the agent will consider this delay as an unpleasant situation and it will feel FEAR⟶ it will immediately schedule maintenance for any problems within the plane ⟶ after the maintenance it will invoke the plane again for analysing⟶ it will start repairing the problems⟶ it will invoke the plane again for checking⟶ now the plane is prepared⟶ Now after the flight was delayed because of maintenance the plane will be cancelled and start the loop again until the agent doesn't find the situation unpleasant thus won't feel fear and the flight will be ready.

5.3.2.2 Framework Instantiation

We define the same BDIE system $S = \langle k, M_1, M_2 \rangle$, as what we did for the previous case study, where:

- K is the set of those runs r such that, for every natural number m, for each j, $j \in \{1,2,3\}$, if $r(m) = s\ j$, then $r(m+1) = s_j$, which means that if the plane is cancelled, then it will keep so.
- $M_i = \langle B_i, D_i, I_i, fear_i, D_i, I_i, fear_i \rangle$, for each i, $i \in \{1,2\}$.
- B_i is the set of those runs $r_i \in K\ with\ r_i(0) = s_0$. This means that each plane agent believes in how long the time takes for plane checking. Notice that belief is just the information state of the agent, and there is no guarantee that the agent will find the situation unpleasant.
- D_i is a subset of B_i such that, for each run, $r_i \in D_i$there is a number m with $r_i(m) = s_i$. This means that each agent desires to cancel the plane.

- I_i is a subset of K such that, for each run $r_i \in I_i$ and every natural number m, if $r\ i\ (m)$ = $s\ 0$, then $r_i\ (m + l) = s_j (j \in \{1,2,3\})$.This indicates that the plane agent will immediately send the plane for maintenance.
- $Fear_i = \sim D_i \cap I_i$, that is, $Fear_i$ is the set of runs r_i such that for every natural number m, if $r_i\ (m) = s_0$, then $r_i\ (m + l) = s_j\ (j \in \{1,2,3\}\ and\ j \neq i)$.. This means that the plane agent feels Fear and finds the situation unpleasant if the time of checking the plane is too long.

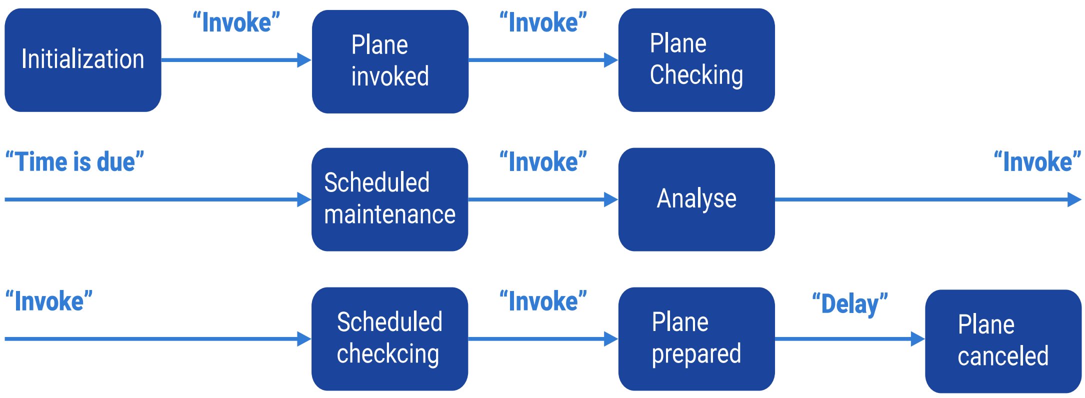

Figure 5.5: Flowchart showing the conversations between the plane agent and flight agent

5.3.2.3 Model Properties Verification

In this sub-section, the verification model transformation from the BDIE to the NuSMV model is provided to verify all the properties including the CUDD library and linked it to the MiniSat SAT solver to make the verification clearer. All the states in the state chart of global behaviors will correspond to the state names of the control behavior.

The mapping between global and control behaviors can be seen in TABLE 5-4. In addition, the states of PlaneScheduled, PlaneServiced; PlaneArrived of the global behavior correspond to the state Processed of the control behavior. The NuSMV code is given just by capturing the transitions of messages between the plane agent and flight agent.

Chapter 6 Now we can address the NuSMV code for our case study, and see how our agent makes its own decision based on the cycle of verifying the state of the plane and the emotion of the agent: which means if the time of the plane by the plane agent checking is too long, the agent will consider this delay as an unpleasant situation and it will feel FEAR, and based on this it will make the decision of cancelling the plane and if not the plane agent will always message the flight agent to proceed with the flight READYTOFLY until the decision making cycle of the agent is fulfilled.

5.3.2.4 Simulation Results

In this subsection, we illustrate the NuSMV model checker validation results by running the above SMV code for the aircraft and flight agents that we already have mentioned.

Experimental environment: The experiments were conducted on a Windows 10 operating system, utilizing a dual-core 3.5 GHz Intel Core i5 computer with 16 GB of RAM.

After executing the code and conducting simulations, the obtained results demonstrate that the Flight agent ensures that the Plane agent cancels the flight if it experiences Fear due to a delay in the plane checking time. This successful implementation significantly enhances the safety of the BDIE agent by reducing the number of cancelled flights, showcasing the practicality of the NuSMV model in simulating real-life scenarios. The experimental findings presented in TABLE 5-5 validate the states of both the plane and flight agents. The following observations can be made: when the time checking is below 0.2 seconds, the flight agents do not perceive it as a delay and do not find the situation unpleasant, thus they do not experience fear. However, if the time checking exceeds 0.2 seconds, the plane agent considers it as an unpleasant situation, triggering fear and resulting in the cancellation of the flight.

The obtained results from the model case study specifications utilizing fear are all accurate and align with the OCC emotional theory.

TABLE 5-5: Verification results

Number of Flights	State of Plane agent	State of the flight	Results	Timing (s)	BDD nodes
Ag.flight A1	FEAR(Delay)	CANCELLED	TRUE	0.245	38458
Ag.flight A2	FEAR(Delay)	CANCELLED	TRUE	0.275	45923
Ag.flightA3	FEAR(Delay)	CANCELLED	TRUE	0.255	45812
Ag.flight A4	FEAR(Delay)	CANCELLED	TRUE	0.345	35907

Ag.flight A5	FEAR(Delay)	CANCELLED	TRUE	0.314	23681
Ag.flight A6	NOT FEAR	READY TO FLY	TRUE	0.143	34517

Our fearful agent-based system goes beyond focusing solely on the behavior of individual agents. It also encompasses the interactive behavior between agents, recognizing the importance of their interactions within the system.

5.4 Discussion

The simulation trials conducted for the two case studies demonstrate the effectiveness of our approach. Our method allows for a comprehensive examination of FEAR behavior in uncomfortable situations, along with other types of emotions. To enhance the efficiency of model checking, our approach emphasizes testing system behavior from multiple perspectives rather than solely expanding formal logic. In terms of verifying global behavior, our approach shares similarities with the work of (Bentahar et al., 2013). The significant differences between our approach and theirs are summarized in **Error! Reference source not found.** below.

5.5 Conclusion

The conclusion of this chapter involves extending the BDI model of agency logic by integrating Fear emotion modalities and proposing a new computationally interpreted fearful model for emotion triggers. This enhancement enables the Emotional-BDI agents to better handle highly dynamic unpleasant situations and their surroundings.

The introduction of the fearful agent in this study demonstrates its cautious behavior, perceiving any threat as a fear factor and treating all uncomfortable events equally. This model can be utilized to develop an agent program that enables cognitive agents to automatically calculate various types of fear emotions, including unpleasantness and discomfort, throughout their operations.

To implement and validate our approach, we employed NuSMV, an open-source checker tool, for two case studies: an auction scenario and a simulation system for aircraft maintenance based on the agent's fear response. Through simulation experiments in these scenarios, we showcased the effectiveness of our method, not only in verifying emotion modalities behaviors but also in accurately examining the FEAR behavior in unpleasant situations.

Table 5.6: Main Differences between Our Approach and Bentahar et al.'s Approach

Our approach	Bentahar et al approach
Our approach is suitable to add any conditions to the agent for emotional verification	Their approach lacks effectiveness in evaluating system behavior when circumstances are interconnected with transitions.
Our approach can be easily implemented using the NuSMV software.	In the absence of any messages using their verified system, if their system is transformed into a system of transitions with conditions, NuSMV would not be capable of validating the emotional requirements for the overall behavior.
In our approach, we have equipped the agents with the ability to transform messages based on their emotional state. By incorporating this feature, agents can adapt their messaging behavior depending on their emotional state, allowing for more dynamic and contextually appropriate communication.	The approach lacks verification by consequence, making it impossible to analyze the global sense or overall impact of the system.
we propose an additional verification process that examines the overall state of the situation. This extended verification ensures a comprehensive assessment of the system, considering not only the emotional behavior but also the broader context and impact of the system's actions.	Their technique for verifying the system's operational behavior serves as a supplementary component to evaluate the system's overall state.

In conclusion, this interdisciplinary effort contributes significantly to our field by building upon the established BDI logic, which is already employed in diverse agent designs. Consequently, our model can be readily applied to any BDI agent, regardless of its specific application, thereby simplifying the development of intelligent virtual agents with proficient skills.

Section II

Validating the Proposed Emotional Agent Model: A Comprehensive Case Studies

Summary

5.1 Introduction

Formalizing knowledge related to human psychology makes it possible to consider interesting perspectives such as behavior simulation, the study, and prevention of psychological changes, etc. The fields of application are therefore very diverse: games, human-machine interaction, the world of work, and even medicine. We will more specifically focus on the modeling of a small part of emotional knowledge (fear).

The major current projects are based on the OCC model (Colby et al., 1989), and focus on the social aspect of emotions ("Affective Reasoner", (Andreas Marpaung, 2018, University of Central Florida) or the link between planning and emotions). However, in all these projects, no notion of beliefs or desires is considered, which distances them from the human behavior proposed by(Bratman, 1987). Our work uses a BDI-type agent endowed with emotions based on OCC. This project is part of a process of managing emotions.

The formalization of knowledge related to human psychology makes it possible to envisage interesting perspectives such as behavior simulation, research, and the prevention of psychological development. The field of application is therefore very diverse. games, human-computer interactions, the world of work, and even medicine.

In this section, we will validate our proposed model (emotional agent) in the case of an aircraft agent. We present the different tools and programming languages, the modelling of our model, the specification of the scenarios, and simulations of the results.

5.2 Software and programming languages

5.2.1 NUSMV

NuSMV (NuBoolean Simplified Model Verifier) is an open-source model checking tool that can be used to verify the correctness of reactive and real-time systems. The NuSMV checker model is a system for automatically checking the correctness of designs and implementations of reactive and real-time systems against formal specifications.

NuSMV uses temporal logic as the language for writing formal specifications of reactive and real-time systems. The tool provides a high-level, user-friendly interface for entering specifications, and it automates the process of checking these specifications against a given design or implementation. The NuSMV checker model operates by constructing a finite state machine representation of the system being analyzed. This state machine is then checked against the formal specification to determine whether it satisfies the desired properties. If the state machine does not satisfy the specification, NuSMV generates a counterexample that shows how the system can behave in a manner that violates the specification.

NuSMV supports a wide range of temporal logics, including linear temporal logic (LTL), computational tree logic (CTL), and timed temporal logic (TTL). This makes it possible to specify and check a wide range of properties of real-time and reactive systems, including safety properties (e.g., "the system will never reach an unsafe state"), liveness properties (e.g., "the system will eventually reach a desired state"), and timing constraints (e.g., "the response time of the system must be less than a specified value").

5.2.1.1 NUSMV Features

The main features of NUSMV are the following (Cimatti et al., 2002):

- ***Temporal Logic Support:*** NuSMV supports several temporal logics, including linear temporal logic (LTL), computational tree logic (CTL), and timed temporal logic (TTL), which allows for the specification of a wide range of properties of reactive and real-time systems.
- ***Finite State Machine Generation:*** NuSMV automatically generates a finite state machine representation of the system being analyzed, which serves as the basis for the model checking process.
- ***Model Checking:*** NuSMV checks the generated finite state machine against the specified temporal logic properties to determine whether the system satisfies the desired properties.
- ***Counterexample Generation:*** If the system does not satisfy the specification, NuSMV generates a counterexample that demonstrates how the system can violate the specification.
- ***User-Friendly Interface:*** NuSMV provides a high-level, user-friendly interface for entering temporal logic specifications and interacting with the model checking process.
- ***Modularity:*** NuSMV supports modular design and verification, which allows for the modular specification and verification of complex systems.

- ***Scalability:*** NuSMV has been designed for scalability, making it possible to handle large and complex systems with many states and temporal logic properties.
- ***Integrations:*** NuSMV can be integrated with other tools and frameworks, such as the SPIN model checker and the UPPAAL model checker.

5.2.1.2 NUSMV Architecture

The architecture of NuSMV (Cimatti et al., n.d.) can be divided into several main components:

- ***Parser:*** The parser component is responsible for parsing the input files, including the specifications and the model, and generating an internal representation of the system.
- ***Model Builder:*** The model builder component takes the internal representation generated by the parser and builds the finite state machine representation of the system.
- ***Model Checker:*** The model checker component checks the finite state machine against the temporal logic properties specified in the input file.
- ***Counterexample Generator:*** The counterexample generator component generates a counterexample trace if the system does not satisfy the specified properties.
- ***User Interface:*** The user interface component provides an interface for entering the specifications and interacting with the model checking process.
- ***Optimizer:*** The optimizer component is responsible for optimizing the model checking process to ensure that the analysis is performed as efficiently as possible.
- ***Backend Engine:*** The backend engine component provides the underlying computational engine that performs the model checking and counterexample generation.

These components are designed to work together seamlessly, with the parser passing the internal representation of the system to the model builder, which in turn passes the finite state machine to the model checker. The model checker passes the results of the analysis to the counterexample generator and the user interface, which present the results to the user.

The modular architecture of NuSMV allows for a separation of concerns, with each component focusing on a specific aspect of the model checking process. This makes it easier to maintain and improve the tool, and also enables users to extend the tool with their own custom components. Figure 5.6 illustrates in details the NUSMV architecture

5.2.1.3 Quality of Implementation

The quality of implementation in NuSMV is high, with a focus on ensuring the reliability and efficiency of the tool. The following are some of the ways that the quality of implementation is maintained and improved in NuSMV (The Embedded Systems Unit in the Digital Industry Center at FBK-IRST et al., n.d.):

Open-Source: NuSMV is an open-source tool, which means that the source code is available for anyone to inspect, use, and modify. This increases transparency and accountability, and also enables users to contribute to the development and improvement of the tool.

Documentation: NuSMV has extensive documentation, including user manuals, developer guides, and reference materials. This makes it easier for users to understand how the tool works and how to use it effectively.

Testing: NuSMV undergoes extensive testing to ensure that the tool is reliable and robust. The testing process includes both automated and manual tests, and covers a wide range of scenarios and use cases.

Bug Reporting and Fixing: NuSMV has a bug reporting and fixing process that allows users to report bugs and track the progress of bug fixes. This helps to ensure that the tool is as bug-free as possible and that users can rely on it for accurate and dependable results.

Community: NuSMV has a large and active community of users and developers, which provides a wealth of knowledge and expertise that can be leveraged to improve the tool. The community also provides support and guidance to new users and helps to promote the tool to a wider audience.

Regular Releases: NuSMV releases regular updates and new versions of the tool, which include bug fixes, new features, and performance improvements. This helps to ensure that the tool remains relevant and up-to-date, and that users have access to the latest and greatest features and capabilities.

5.2.2 CUDD Library

The CUDD library (CUDD: CU Decision Diagram Package) is a widely used software library for the manipulation of binary decision diagrams (BDDs) and multi-terminal binary decision diagrams (MTBDDs) (Bresolin & Villa, n.d.). BDDs and MTBDDs are data structures used to represent and manipulate Boolean functions, and they are widely used in computer-aided design, verification, and testing.

The main features of the CUDD library include (Somenzi, 2018):

- ***Efficient Representation of Boolean Functions:*** CUDD provides an efficient representation of Boolean functions using BDDs and MTBDDs. This makes it possible to manipulate and analyze large Boolean functions quickly and efficiently.
- ***Support for Dynamic Variable Reordering:*** CUDD supports dynamic variable reordering, which allows the order of the variables in the BDD to be changed during the manipulation process. This helps to reduce the size of the BDD and improve performance.
- ***Automated Garbage Collection:*** CUDD includes an automated garbage collection mechanism, which frees up memory when BDD nodes are no longer needed. This helps to prevent memory leaks and improve the overall performance of the system.
- ***Support for Multi-Threaded Operations:*** CUDD supports multi-threaded operations, which allows multiple

5.2.3 MINISAT Solver

Minisat is an open-source Boolean Satisfiability (SAT) solver written in C++. It's designed to solve the SAT problem, which involves determining whether there exists a set of truth assignments for a set of Boolean variables such that a given Boolean formula evaluates to true. Minisat can be used in various applications, such as automated theorem proving, software verification, and circuit design.

The core algorithm used by Minisat is the DPLL (Davis-Putnam-Logemann-Loveland) algorithm, which is a backtracking algorithm for solving the SAT problem. The DPLL algorithm starts by assigning truth values to variables in the formula and then uses the rules of Boolean logic to simplify the formula until either a contradiction is found or a satisfying assignment is discovered. If a contradiction is found, the algorithm backtracks and tries another assignment. If a satisfying assignment is found, the algorithm terminates.

Minisat also employs several optimizations and heuristics to make the solving process more efficient, such as clause learning, restart strategies, and variable scoring heuristics. Clause learning is a technique where the solver stores the information about failed assignments, which can be used to prune future search spaces. Restart strategies are used to control the frequency of backtracking, which can help avoid getting stuck in long, unproductive search paths. Finally, variable scoring heuristics are used to select variables for assignment in an attempt to make the search process more effective.

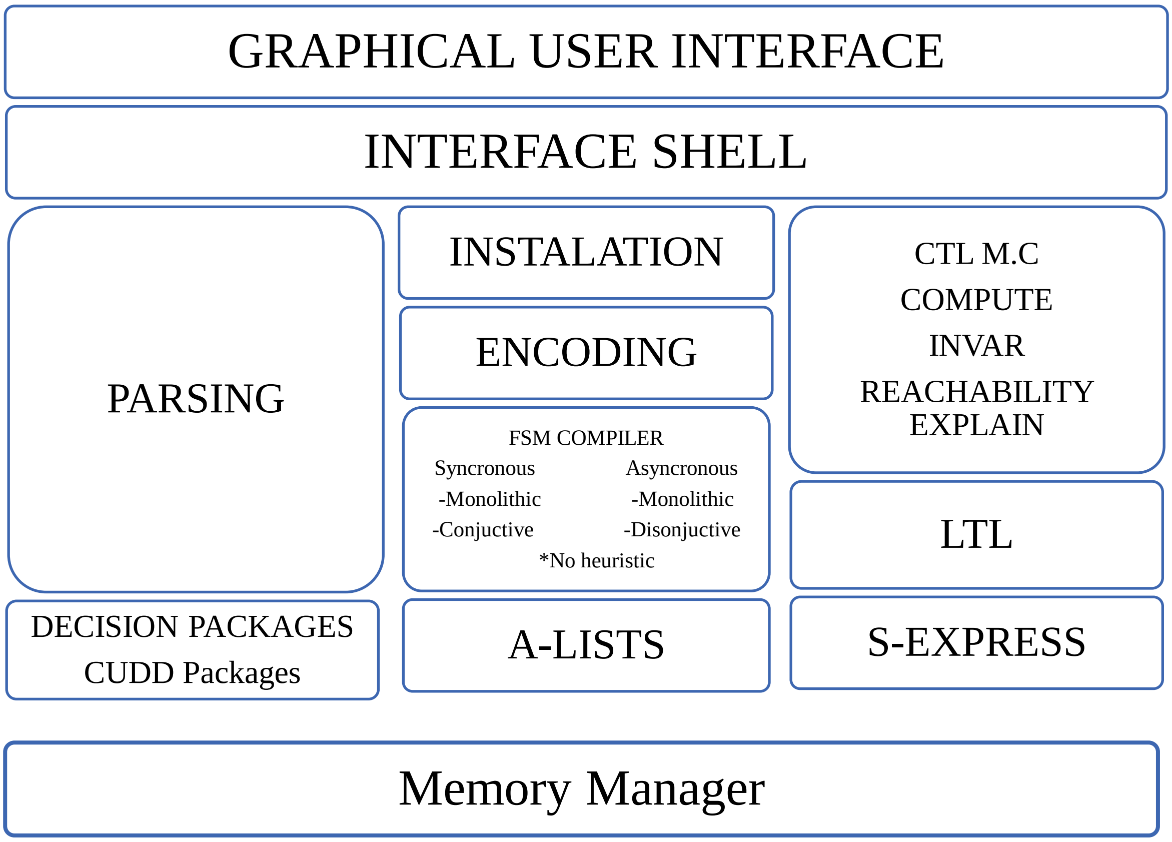

Figure 5.6: NUSMV Architecture (The Embedded Systems Unit in the Digital Industry Center at FBK-IRST et al., n.d.)

Minisat is widely used due to its efficiency and scalability, and it has been competitive in various SAT solver benchmarks. Additionally, the open-source nature of Minisat makes it a popular choice for researchers and practitioners in the SAT solving community; Minisat has several key features that make it a popular choice among SAT solvers:

- ***Efficient and Scalable:*** Minisat is designed to solve large SAT problems in a fast and efficient manner. Its core algorithm, the DPLL algorithm, is optimized with various heuristics to make the solving process more efficient, such as clause learning, restart strategies, and variable scoring heuristics.

- ***Open-source:*** Minisat is open-source software, which makes it accessible to a wide range of users, including researchers, students, and practitioners. This also means that it can be easily modified and extended to meet specific needs.
- ***Cross-platform compatibility:*** Minisat can be easily compiled and run on different operating systems, including Windows, Linux, and macOS.
- ***Simple and clean code:*** Minisat has a simple and clean codebase, which makes it easy to understand and modify. This is important for researchers who want to understand the inner workings of the solver or make modifications to it.
- ***Widely used:*** Minisat is widely used and has been competitive in various SAT solver benchmarks. This makes it a popular choice for researchers, students, and practitioners in the SAT solving community.
- ***Command line interface:*** Minisat provides a simple command line interface, making it easy to use and integrate into other tools and applications.

These features, combined with its efficient and scalable solving capabilities, make Minisat a popular choice among SAT solvers.

5.3 Psychological modelling of the fearfull agent

5.3.1 The procedural system

The procedural system is the cognitive entity of the agent built on the BDI principle: the agent perceives its environment, updates its beliefs, and then acts. Our approach's originality is that the agent's emotional state influences the reasoning process.

5.3.2 Memory

Memory allows agents to store their knowledge and consists of two parts: short-term memory (STM) and long-term memory (LTM). short term memory makes it possible to retain dynamic information (BDI model): beliefs (knowledge that the agent has of itself and of the environement in which it is), desires, and intention. Long-term memory is seen as a library of action plans corresponding to methods known to the agent. Therefore, it is not very dynamic.

5.3.3 Personal characteristics

Finally, the personal characteristics include all the behavioral variables internal to the agent. We can cut the personality of an agent into three levels, based on the speed of evolution of the parameters, which are the temperament, the feelings, and the emotions Figure 5.7 illustrates the personal characteristics of the agent.

- **A person's temperament**, considered fixed, is based on the performance model which describes temperament according to ten personality traits.
- **The emotional state**: the agent has an emotional state that will translate at each moment the emotion that the agent feels. For this, we added, to the OCC model, conditions on temperament for the generation of emotions and their intensity.
- **Feelings**: To safeguard the history of human relationships, we have introduced the notion of feelings; they are built over time and events. In addition, our agent has a psychological state (a mental state to work, part already validated by a prototype). Finally, the agent has a physical state which is used to know his position in the plane and his facial expression (manifestation of the emotional state (Maffei et al., 2014) modulated by temperament).

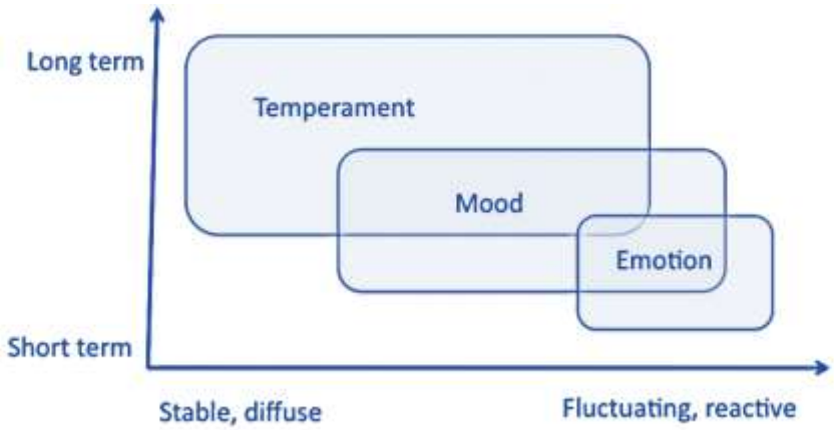

Figure 5.7: Personal characteristics of the agent

5.4 Modelling

Emotional-BDI agency describes computational agents whose behavior is guided by the interactions existing between beliefs, desires and intentions (along the lines of the classical BDI architecture (Bratman, 1987)), but where these interactions are influenced by a third-party emotional component (Pereira et al., n.d.) and(António Damásio, 1994). This component produces data which will bind the BDI interaction by imposing some of the large set of positive aspects that emotions play in reasoning and decision-making (António Damásio, 1994).

The conceptual architecture which defines the Emotional-BDI model of the agency was recently introduced by (Pereira et al., n.d.) and is mainly based on the work of emotional agents Oliveira & Sarmento's emotional agent architecture, although adapted to fit in the original BDI architecture (Bratman, 1987).

5.4.1 UML modelling

To be able to choose its action and dialogue, and interact with its environment, the agent must have a representation of what surrounds it. To represent all these types of knowledge, many formalisms exist and are proven (logic, conceptual graphs, etc.). To model our agent, we chose the formal language UML (Booch et al., 1999) because it is a universal language without ambiguity by its formalism and independent of any programming language. Each part of the agent presented in the previous part translates into a class in UML (Figure 5.8).

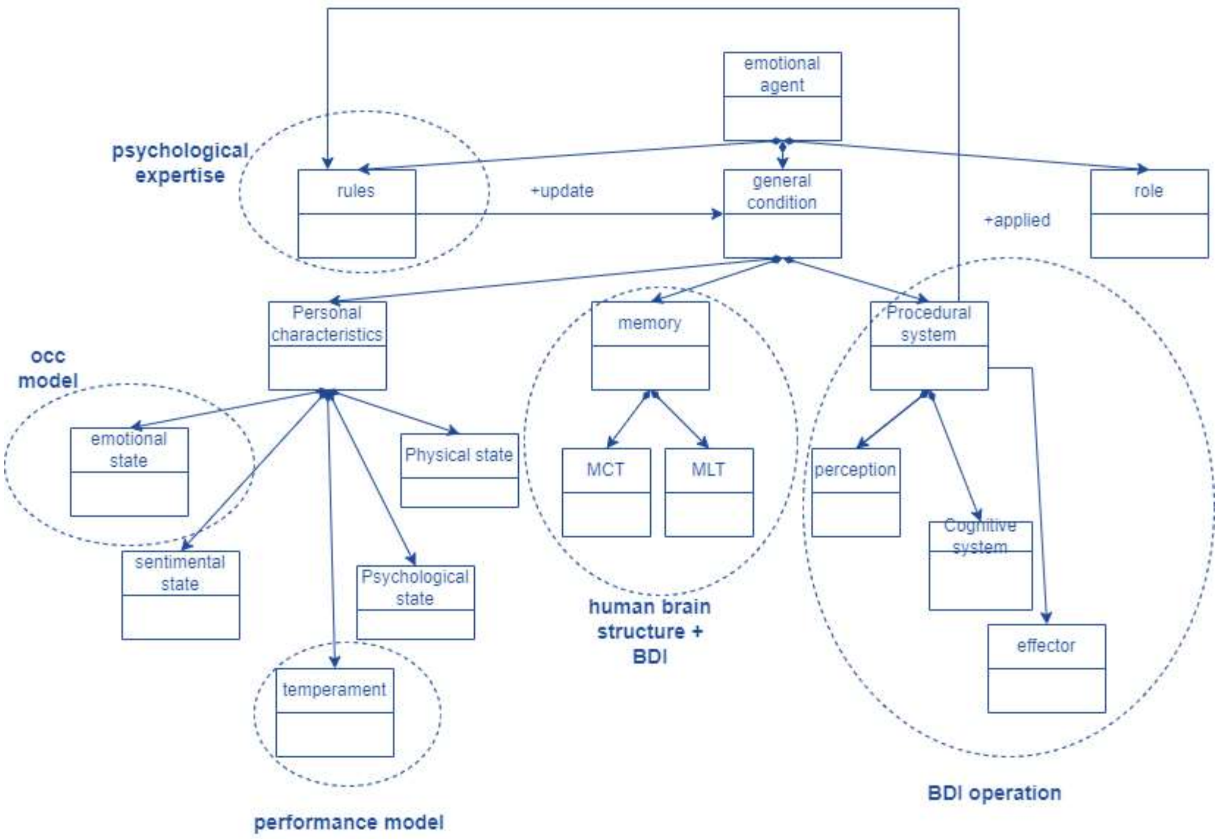

Figure 5.8: The emotional agent UML model

We will now detail the organization of memory in an agent based on the UML class diagram of memory proposed in Figure 5.8. The diagram is limited in size, not all attributes have been detailed. Static knowledge (class MLT) is composed of a list of elementary actions (linked to a

Situation Result or state of the world after this action) and action plans which are all the agents know how to do. Concerning the MCT class, it makes it possible to store all the knowledge that an agent has in its environment. We distinguish the beliefs that an agent has about itself (reference to its characteristics) and those it has about environement. In these last, it first has knowledge of the machine (class Machine State with start/stop type attributes, number of batches in a given zone, current packet status). Then, the agent stores knowledge about other agents (Role of Agents, for the theoretical know-how of others and Agents' Sentiments, Agents' Emotional State, and Agents' Position for the internal status). This is stored in the same form as the state specific to the agent (the value class agent’s which associates a given agent with a value corresponding to a feeling or emotion).

Having formalized all these psychological indicators in UML will allow us to store easily in a knowledge base and therefore to be able to exploit this knowledge.

5.4.2 Modelling Fear

In the context of Emotional-BDI (Belief-Desire-Intention) agents, a fearful agent is one that acts in a self-preservation manner when triggered by the emotion of fear. This means that the agent's actions are driven by the desire to protect itself from potential threats or harm. The intensity of fear experienced by the agent can vary based on individual differences, and this variation can result in different classes of Emotional-BDI agents.

The study of fear as an emotion is important in the context of artificial intelligence and autonomous agents, as it plays a significant role in shaping the behavior of these agents. By modeling different classes of fearful agents, it is possible to better understand the role of fear in shaping an agent's behavior and to design more effective and realistic artificial agents.

5.4.2.1 Threats and unpleasant facts

Fear and other negative emotions arise when there are unwelcome or unmanageable facts or happenings in the environment. Here, only negative facts and threats are taken into account.

Threats are environmental facts or occurrences that directly impact one or more of the agent's essential goals, endangering the agent's ability to protect itself. Depending on their severity for the agent, these dangers may have varying weights. Specifically, we take into account:

- ***Dangerous***: a threat is risky when the agent believes that some condition ψ leads inevitably to the falsity of a fundamental desire ϕ, and also believes that ψ will also be inevitably true in the future.

DangerousThreat $(\psi, \phi) \equiv \text{FDES}(\phi) \wedge \text{BEL}(\psi \rightarrow \text{AF}(\neg\phi)) \wedge \text{BEL}(\text{AF}\psi)$

- ***Serious:*** a threat is serious if the same conditions of a dangerous one hold, except that the agent believes that ψ may eventually be true in the future, and not always.

SeriousThreat $(\psi, \phi) \equiv \text{FDES}(\phi) \wedge \text{BEL}(\psi \rightarrow \text{AF}(\neg\phi)) \wedge \text{BEL}(\text{EF}\psi)$

- ***Possible****:* a threat is possible if the fundamental desire continues at stake, but the agent believes that it may hold only in the future.

PossibleSeriousThreat $(\psi, \phi) \equiv \text{FDES}(\phi) \wedge \text{BEL}(\psi \rightarrow \text{EF}(\neg\phi)) \wedge \text{BEL}(\text{EF}\psi)$

- Unpleasant facts represent facts or events which put one or more desires at risk of non-achievement. The agent may exhibit distinct behavior towards such an unpleasant fact, for protecting its desires. Here we consider the following:
- ***Highly Unpleasant:*** something becomes highly unpleasant if the agent believes that the source of the unpleasantness will always occur in the future and will always in the future put in cause some desire.

HighlyUnpleasant $(\psi, \phi) \equiv \text{DES}(\phi) \wedge \text{BEL}(\psi \rightarrow \text{AF}(\neg\phi)) \wedge \text{BEL}(\text{AF}\psi)$

- ***Strongly Unpleasant***: something becomes strongly unpleasant if the agent believes that the source of the unpleasantness will eventually occur in the future and will always in the future put in cause some desire.

StronglyUnpleasant $(\psi, \phi) \equiv \text{DES}(\phi) \wedge \text{BEL}(\psi \rightarrow \text{AF}(\neg\phi)) \wedge \text{BEL}(\text{EF}\psi)$

- ***Possibly Unpleasant:*** something becomes possibly unpleasant if the agent believes that the source of the unpleasantness will eventually occur in the future and, in the case of occurring, maybe it will put in cause some desire.

Possibly Unpleasant $(\psi, \phi) \equiv \text{DES}(\phi) \wedge \text{BEL}(\psi \rightarrow \text{EF}(\neg\phi)) \wedge \text{BEL}(\text{EF}\psi)$

These concepts will be used in what follows to model the triggering of fear and to show that special processing strategies may be applied when facing certain conditions.

5.4.2.2 Special atomic actions

We define the following special purpose actions, which represent specific behavior exhibited by the agent under certain conditions.

Self-preservation: the self-preservation behavior is activated when the agent is fearing the failure of some of its fundamental desires. We can see this as atomic action which mainly reacts to threats in a self-protective way. In EBDI, this special action is represented by SServ.

Motivated Processing Strategy: this processing strategy is employed by the agent when some desire which directs its behavior must be maintained but may be at risk. This strategy is computationally intensive, as it should produce complex data-structures for preserving desires. In EBDI, this kind of processing is abstracted into the specialized atomic action model.

Direct Access Strategy: this processing strategy relies on the use of fixed pre-existing structures/knowledge. It is the simplest strategy and corresponds to a minimization of the computational effort and fast solutions. In EBDI, this kind of processing is abstracted into the specialized atomic action model.

Considering the above actions as being atomic actions is of course a big abstraction to the complexity of Emotional-BDI agents. These actions are usually complex planning and revision strategies.

5.4.2.3 Specifying a fearful agent

We will now introduce the fearful Emotional-BDI system, but first we are going to present the intuition behind it.

A fearful agent is an agent who exhibits very careful behavior, considering any threat as a fear factor, and considering all uncomfortable events at the same level. Being careful, the agent also acts towards solving threats and unpleasant facts with the best of its means, and even if the means for the best action are not available, the agent always tries to put itself in a safe, or self-preservation condition.

Let the following equivalences be defined

Any Threat $(\psi, \phi) \equiv$ Dangerous Threat$(\psi, \phi) \vee$ Serious Threat$(\psi, \phi) \vee$Possible Serious Threat(ψ, ϕ)

Any Unpleasant $(\psi, \phi) \equiv$ Highly Unpleasant$(\psi, \phi) \vee$ Strongly Unpleasant$(\psi, \phi) \vee$Possibly Unpleasant(ψ, ϕ)

We define a fearful Emotional-BDI system as follows

Let ϕ, ψ be well-formed formulae and $\alpha \in ARa$ be a regular action. The specification of a fearful Emotional-BDI agent is given by all the axioms of an Emotional-BDI agent, plus the following new axioms:

– any threat to a fundamental desire ϕ makes the agent fear $\neg\phi$

- $AnyThreat(\psi, \phi) \rightarrow FEAR(\neg\phi)$

– any threat ψ to a fundamental desire ϕ is also a fear of the agent

-$AnyThreat(\psi, \phi) \rightarrow FEAR(\psi)$

– if the agent after deliberating, using the motivate processing strategy, believes that either ψ will not verify or that ϕ and ψ are compatible, then he considers ψ as an unpleasant fact for the achievement of ϕ hmpsdeli$((BEL(\neg\psi) \vee BEL(\psi \wedge \phi)) \wedge DES(\phi)) \rightarrow$ Any Unpleasant(ψ, ϕ)

– if the agent is threatened and has no effective resources for executing a deliberation based on direct access strategies (which could bring good solutions for avoiding the threat), the agent can execute a self-preservation action

Any Threat$(\psi, \phi) \wedge \neg$EffCap(dasdel) $\rightarrow$ hspreservi$\top$

– if the agent has effective capabilities, it executes instead the direct access strategy based deliberation

Any Threat$(\psi, \phi) \wedge$ EffCap(dasdel) $\rightarrow$ hdasdeli$\top$

It is important to stress that the presented fearful Emotional-BDI agent is not the unique extension to the basic Emotional-BDI agent. This system only characterises agents whose fear is triggered by any threat, and that reconsider its desires at the first unpleasant fact that interferes with them. If, for instance, we substituted Any Threat$(\psi, \phi) \rightarrow FEAR(\neg\phi)$ by Dangerous Threat$(\psi, \phi) \rightarrow FEAR(\neg\phi)$, we would be specifying Emotional-BDI agent which only fear dangerous threats.

5.5 Specification of scenarios

Based on our formal description of emotional triggers, each agent is capable of expressing feelings. These emotional characteristics can be validated in the auction scenario using the NuSMV checker model tool, Figure 5.9 illustrates an overview of the code in NUSMV language.

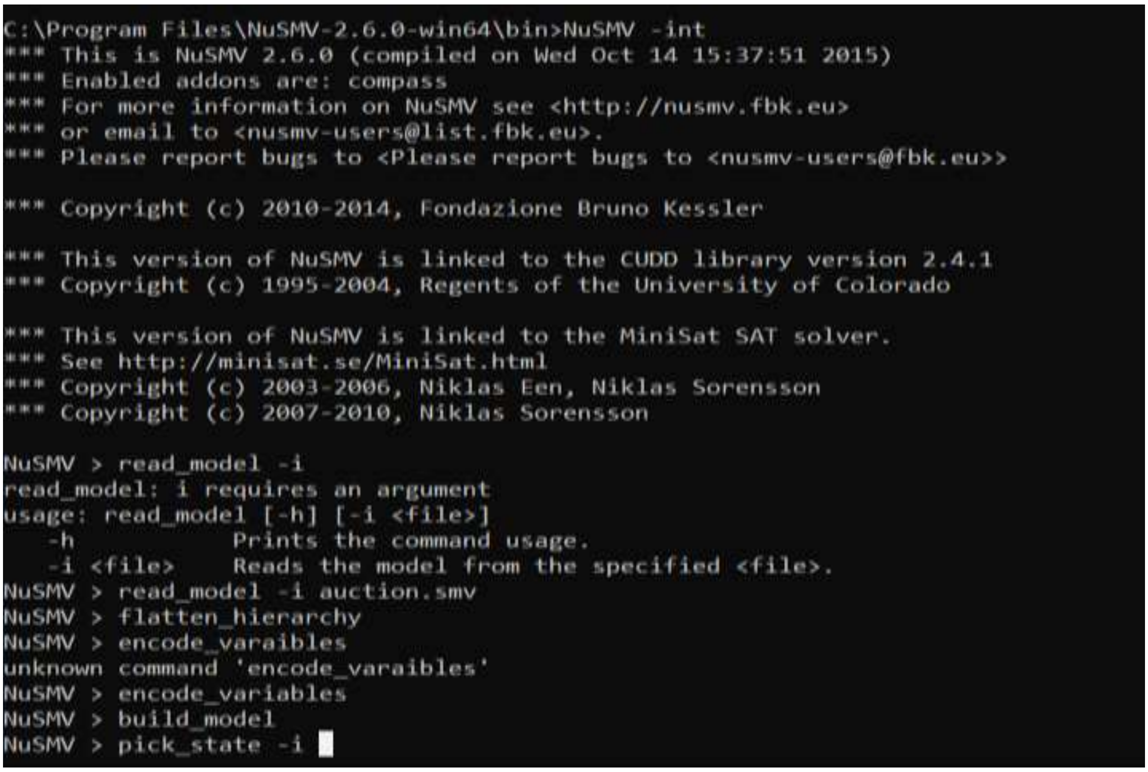

Figure 5.9: Preview of the code in NuSMV language

The B.1 and B.2 listing codes presents the code in NuSMV language for the **agent 1 and agent 2** (Check Figure 5.10) **,** There are four global states: s0, s1, s2, and s3. In the state s0, each agent bids, and in the states s1, s2, and s3, a winner is announced. Specifically, in s1, the winner is ag1, in s2, the winner is ag2.

A.1 listing code for agent 1

```
MODULE auc_scen(ag1_ct_state)
VAR
   state :{
        S0,
        S1,
        S2,
        win,
        Not Fear,
          };
   ASSIGN
     init(state) :=S0;
     next(state) :=case
   state=S0 : S1 ;
   state=S1 :ag1_ct_state;
   state=ag1_ct_state :win;
   state=win : Notfear;
```

```
 TRUE :Notfear;
esac;
```

Command Prompt - NuSMV -int

```
A model must be read before. Use the "read_model" command.
NuSMV > read_model -i auction.smv
NuSMV > flatten_hierarchy

file auction.smv: line 14: "ag1_ct_state" undefined

NuSMV > read_model -i auction.smv
NuSMV > flatten_hierarchy
NuSMV > encode_variables
NuSMV > build_model
NuSMV > pick_state -i

***************  AVAILABLE STATES  *************

  ================= State =================
  0) -------------------------
  state = S0


There's only one available state. Press Return to Proceed.

Chosen state is: 0
NuSMV > print_reachable_states -v
######################################################################
system diameter: 5
reachable states: 5 (2^2.32193) out of 6 (2^2.58496)
  ------- State    1 ------
  state = NotFear
```

Figure 5.10: The results of the auction scenario when the state is S1

B.1 listing code for agent 2

```
MODULE auc_scen(ag2_ct_state)
VAR
   state :{
         S0,
         S1,
         S2,
         loose,
         Fear,
           };
   ASSIGN
     init(state) :=S0;

     next(state) :=case
   state=S0 : S1 ;
   state=S1 :ag2_ct_state;
   state=ag2_ct_state :loose;
   state=loose : fear;
   TRUE :fear;
```

esac;

```
NuSMV > simulate -i -k 1
********  Simulation Starting From State 1.1   ********

***************  AVAILABLE STATES  *************

  ================= State =================
  0) -------------------------
  state = Fear


There's only one available state. Press Return to Proceed.

Chosen state is: 0
NuSMV > print_reachable_states
######################################################################
system diameter: 2
reachable states: 2 (2^1) out of 6 (2^2.58496)
######################################################################
NuSMV > print_reachable_states -v
######################################################################
system diameter: 2
reachable states: 2 (2^1) out of 6 (2^2.58496)
  ------- State    1 ------
  state = Fear
```

Figure 5.11: The results of the auction scenario when we the state is S2

Each agent expects to win at the start of each auction round. If Agent 1 wins, for instance, he is happy and does not experience dread because the desired outcome, that he wins, occurs after bidding. Agent 2, on the other hand, experiences Fear since they lose after bidding, which is undesired, results are displayed in Figure 5.11 .

These calculations are validated by the experimental results in TABLE 5-2. (ag1 & ag 2). The following is discernible: Both of the above emotional specifications' findings are accurate and in line with the OCC emotional theory.

5.6 Simulation Experiment Results

By running the SMV codes for the plane and flight agent that we have stated previously, we demonstrate the verification findings from the NuSMV model checker in this subsection.

```
Microsoft Windows [Version 10.0.19043.1706]
(c) Microsoft Corporation. All rights reserved.

C:\Users\User>cd..

C:\Users>cd C:\Program Files\NuSMV-2.6.0-win64\bin

C:\Program Files\NuSMV-2.6.0-win64\bin>NuSMV -int
*** This is NuSMV 2.6.0 (compiled on Wed Oct 14 15:37:51 2015)
*** Enabled addons are: compass
*** For more information on NuSMV see <http://nusmv.fbk.eu>
*** or email to <nusmv-users@list.fbk.eu>.
*** Please report bugs to <Please report bugs to <nusmv-users@fbk.eu>>

*** Copyright (c) 2010-2014, Fondazione Bruno Kessler

*** This version of NuSMV is linked to the CUDD library version 2.4.1
*** Copyright (c) 1995-2004, Regents of the University of Colorado

*** This version of NuSMV is linked to the MiniSat SAT solver.
*** See http://minisat.se/MiniSat.html
*** Copyright (c) 2003-2006, Niklas Een, Niklas Sorensson
*** Copyright (c) 2007-2010, Niklas Sorensson

NuSMV > _
```

Figure 5.12: Illustration of the verification results from NuSMV model

In this sub-section, we illustrate the verification results from NuSMV model checker by executing the SMV codes for the plane and flight agent that we have mentioned above Figure 5.12 depicts an Illustration of the verification results from NuSMV model.

Experimental environment: All trials were conducted on a dual-core Intel Core i5 PC running Windows 10 at 3.5 GHz and 16 GB of RAM.

To conclude, the results above show that the Flight agent assured that the Plane agent will cancel the flight if he was in a state of Fear due to the delay of time of plane checking (See Figure 5.13). It has successfully improved the safety of the BDIE agent in terms of canceled flights, which illustrates the feasibility of the implementation of the NuSMV model of the simulation system in these real-life situations.

The experimental results in TABLE 5-3 verify the states of plane and flight agents' states. The following can be observed: when time checking is below 0.2 seconds the flight agents will not consider this as a delay and don't find the situation unpleasant so they will not fear but if the time checking exceeds 0.2 seconds The plane agent will consider this delay as an unpleasant situation, and it will feel fear and cancel the flight. the results of the above model case study specifications using fear are all true and consistent with the OCC emotional theory.

```
NuSMV > simulate -i -k 1
********  Simulation Starting From State 1.1   ********

***************  AVAILABLE STATES  *************

  ================= State =================
  0) -------------------------
  state = delay


There's only one available state. Press Return to Proceed.

Chosen state is: 0
NuSMV > print_reachable_states -v
######################################################################
system diameter: 2
reachable states: 2 (2^1) out of 6 (2^2.58496)
  ------- State    1 ------
  state = fear
  ------- State    2 ------
  state = delay
  -------------------------
######################################################################
```

Figure 5.13: Shows the results will be a delay after the agent felt fear due to the time plane

5.7 Discussion

Emotional agents (BDI) are agents whose behavior is guided by several properties: beliefs, desires, intentions, and the role of emotions in reasoning, also through decision-making, can express their emotions using an emotional trigger that is formally described. this model that we propose was developed based on EBDI logic to specify Emotional-BDI agents in general and a special kind of Emotional-BDI agent under the effect of fear. The focus of this work is the expressiveness of EBDI and its use to establish properties that agents must exhibit under the effect of emotion.

The NuSMV Verifier Model Tool and CUDD were used to verify an agent's emotional attributes during the specified scenarios.

5.8 Conclusion

We have presented in this chapter a successful model of emotions: we are starting from a recognized psychological theory, and we have translated it into a generic and reusable logical formalism while remaining as faithful as possible to the psychology, we then implemented this theoretical model in a software agent, then we evaluated the emotions of this agent to derive conclusions on the OCC theory and our BDI model.

The results of this first evaluation open many prospects for improvement, at least for the part that we can modify our formalization of the OCC typology. But on the other hand, we also want interest now in the formalization of the same logic of other psychological theories, to compare their predictions.

We are interested in the appraisal theory of Lazarus, which often seemed to us finer or even more in agreement with the opinions of the judges during assessments. This theory is however significantly more complex than the OCC typology, not being created to allow AI researchers to implement it. It involves complex concepts of responsibility and self-involvement... which will be difficult to formalize in our current BDI logic, unless to integrate into our theory of action the concept of "agency". It is, therefore, necessary to ask the question of the relationship between the benefits obtained and the additional costs generated using such a theory. Is it necessary for a software agent to express the fine distinctions between shame and guilt, or between jealousy and envy, which Lazarus highlights? Finally, the solution for these agents will perhaps be to use a compromise between all the theories, sometimes using simple but sufficient notions, and sometimes more complicated definitions of certain critical emotions.

In any case, we think that insofar as we currently wish, for various reasons, to make the agents as credible as possible, it is without a doubt interesting to explore more complex psychological theories than the traditional OCC typology. Moreover, psychology itself could perhaps draw benefit from such research, through the possibility’s evaluation of theories, to better understand human emotions.

Chapter 6

Social Media Sentiment Analysis: A Filtering Mechanism Based on Similarity-Based IBM API

Summary

Chapter 6 : Social Media Sentiment Analysis: A Filtering Mechanism Based on Similarity-Based IBM API

6.1 Introduction

Nowadays, sentiment analysis has grown to be one of the most active research areas in text mining and natural language processing. Two factors are the key causes of its appeal. First, it has demonstrated its usefulness and efficiency in determining the precise emotion in a particular situation. Second, it has numerous uses in fields including psychology, law, forecasting, and illness prediction. These themes have been the subject of extensive research, which yielded significant discoveries. However, emotional sentiment analysis of social media content did not take sufficient attention by researchers, Moreover , the excessive usage of social media has brought on an imbalance in people's mental health, which is a significant problem for this generation, academics, and society, Several studies have identified a connection between using social media and unfavorable consequences such as increased anxiety, stress, depression, and loneliness, so accordingly, younger generations' use of social media raises serious concerns regarding its damaging effects. Based on these findings, we are proposing a filtering mechanism that aims to eliminate any social media content that users would perceive as possibly emotionally harmful. The proposed approach is twofold: Firstly, we obtain social media content, evaluating it using IBM API, Secondly, we apply a similarity metric distance in order to determine how similar (or dissimilar) the social media content is to the best emotional content, before removing any of it that exceeds a certain threshold, the proposed approach is demonstrated through a case study employing Twitter as a social media platform. The results confirms that the proposed approach can be applied to avoid the emotional harmfulness caused by social media. This extensive research's primary goal is to evaluate the proposed solution we have conducted to raise the caliber of the content displayed to users emotionally.

It is essential to recognize that this chapter originates from our groundbreaking research entitled “Analyzing Emotional Sentiment in Social Media Content for Mental Health Safety.” (Benrouba & Boudour, 2023).

6.2 Methodology

This section describes our approach to enhancing the emotional quality of content that is displayed to the user. Initially, we obtained social media content through the Twitter API. We then analyzed and sorted the content into five fundamental emotional categories, namely joy, sadness, anger, fear, and disgust. This was done by utilizing the natural language understanding API from IBM, which is an open-source tool for textual emotion recognition. Subsequently, we transformed the emotional data set into an emotions array. Furthermore, we defined a perfect emotion array by extracting 450 words from the English language. To evaluate the similarity between the emotions array of each tweet content and the perfect emotions array, we computed the Euclidean distance. We set a threshold value of 1.08. If the distance is less than or equal to this threshold, the content is considered suitable for display. However, if the distance is greater than the threshold, a warning message is displayed, indicating that the content may cause emotional harm.

6.2.1 Data Preprocessing

To begin building our application, we required data from social media. Therefore, we utilized the Twitter API "tweets" to gather all of the necessary data to proceed with our analysis. The next step, and in order to analyze and classify the content into five primary emotion categories previously mentioned, we leveraged IBM's natural language understanding API.

6.2.2 IBM Natural Language Understanding API

IBM Watson Natural Language Understanding API for sentiment analysis is a cloud-based service that uses natural language processing and machine learning algorithms to analyze text and determine the sentiment behind it. The API can analyze text data in various forms, such as articles, reviews, social media posts, and customer feedback, and provides insights on the overall sentiment of the content.

The sentiment analysis feature in the API can detect the emotional tone of a text, including whether the sentiment is positive, negative, or neutral. It also provides a confidence score that indicates how confident the API is in its analysis. Below are the main reasons why we chose to utilize the IBM API in our approach:

- ***Accurate analysis:*** The API uses advanced natural language processing and machine learning algorithms to analyze text data, which can lead to accurate sentiment analysis.

The API can detect subtle nuances in language and understand the context in which the text is being used.

- ***Customization:*** The API provides the flexibility to customize the sentiment analysis based on the domain or industry of the text data being analyzed. It allows users to create custom models that are tailored to their specific use case, which can improve the accuracy of the sentiment analysis.
- ***Scalability:*** The API is a cloud-based service, which means it can handle large volumes of text data and analyze it in real-time. This makes it ideal for organizations that need to analyze large volumes of text data in a short period of time.
- ***Integration:*** The API can be easily integrated into other applications and platforms, such as chatbots, customer service tools, and social media monitoring tools. This can help organizations improve their customer engagement and response times.
- ***Cost-effective:*** The API is a cost-effective solution for sentiment analysis, as it eliminates the need for organizations to invest in expensive hardware and software to perform sentiment analysis. Instead, organizations can use the API as a pay-per-use service, which can help reduce costs.

6.2.3 Sentiment Analysis of tweet content

In order to conduct sentiment analysis on the dataset obtained from the Twitter API, we utilized the IBM natural language understanding API to extract emotions from approximately 450 commonly used English words that express positive emotions. Each tweet was then converted into an emotion array $X[i,j]$ for further analysis. Once the data for each tweet was collected in the form of an emotion array $X[i,j]$ from the IBM natural language understanding API, the content was then analyzed in the subsequent step.

6.2.4 Emotion Analysis of tweet content

In this step, we aimed to filter emotionally charged content by compiling a list of over 450 English words commonly used to express positive emotions. We then used the IBM API to analyze this list, and the resulting output is defined as the ideal emotions array. The values of this array are as follows:

$"emotions": \{$

$"sadness": 0.005441,$

$"joy": 0.997533,$

"$fear$": 0.009288,

"$disgust$": 0.001843,

"$anger$": 0.002108 }

We defined a perfect emotion array as a matrix Y,where Y is a $m * n$ matrix, where each row represents a sample and each column represents the 5 emotions respectively. corresponds to the value of all 5 emotions respectively.

6.2.4.1 Evaluation of emotion

In order to classify the emotion array, we got from our given tweet in the dataset, we will compute the Euclidean distance between every tweet in the dataset array $X[i, j]$ from the tweet data content and the perfect emotion array $Y\ [i, j]$

6.2.4.2 Euclidean Distance

The Euclidean distance is a measure that can be used to determine the similarity or dissimilarity between two vectors, X and Y, as defined in equation 1. Essentially, it calculates the sum of the squared differences between corresponding elements of the two vectors. The reason for using the Euclidean distance is that it has been shown to be effective for text classification and clustering, and is one of the most commonly used distance metrics. It can also be viewed as an inverse similarity function.

To classify tweets as emotionally safe or harmful content, we set a threshold value of 1.083274218344552 based on our experimentation and analysis results. This threshold value was chosen specifically after evaluating the performance of different threshold values:

- First, we have collected more than 800 words that can be used to express negative emotions.
- Next, we analyzed the words, to get the following output

{"$sadness$" : 0.268492,

"joy" : 0.006753,

"$fear$" : 0.296736,

"$disgust$" : 0.012359,

"$anger$" : 0.201871}

• The next step in the process is to calculate the euclidean distance between the perfect emotion array and the output that we get from the previous step, this value is going to be defined as the threshold.

The value that we have obtained (1.08) expresses how much the contents are different, as we have previously mentioned the greater the distance, the greater the difference (check equation 1, E.D stands for Euclidean distance). Now if the distance is equal to or less than (1.08) then we assume that the tweet content is emotionally safe. Otherwise, we display a warning message that this content could be emotionally harmful.

$$\mathrm{E.D}(\mathrm{x},\mathrm{y}) = \sqrt{\sum_{i=1}^{d}(x_i - y_i)^2 + \sum_{j=1}^{d}(x_j - y_j)^2}$$

Figure 6.1 illustrates in detail the proposed approach. We built the application using Nodejs and Angular framework and made it available on Github.

6.3 Results and Discussion

Text emotion detection is one of the hardest tasks in natural language processing because even humans struggle to analyze sentiments accurately, here are some of the limitations that we faced when we were building the application:

- The language and Intentions: We faced a lot of problems concerning languages because the intention of the tweets sometimes varies on what real means.

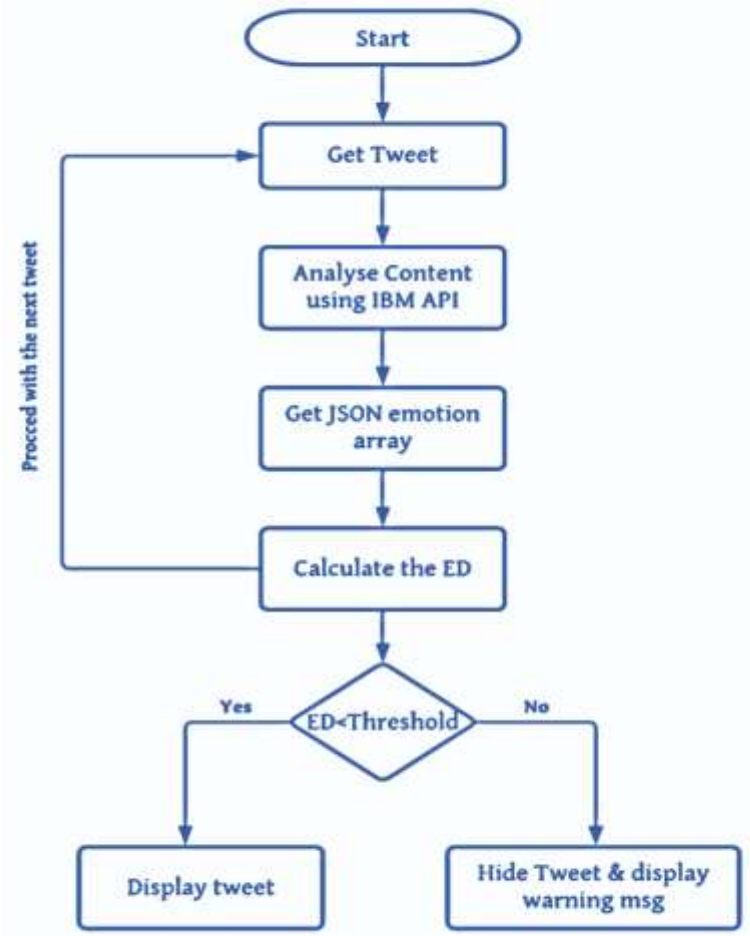

Figure 6.1: Flowchart of the proposed approach

- The media content: Text is not the only content in social media there are also photos and videos, which can be emotionally harmful in the same way as text.

Furthermore, there are certain threats to the validity of our approach primarily in inherent differences between the compared tools. First, the method of calculating the Euclidean distance, which could not have been avoided, might have produced certain inconsistencies. Second, the nature of this tool type itself might have affected the results. IBM tool does not treat neutral as a special type, so we had to determine a threshold which was 1.08, the dataset combined the emotions of anger and disgust into one emotional type (label). (Neviarouskaya et al., 2009). had a similar problem with certain inconsistencies in the emotional model. Most IBM tool errors come from cases where additional context is required for the correct interpretation of textual emotion. This emphasized the need to include word sense disambiguation in our future research. Further research is needed to determine how much the difference between the data set and the second data set emotion (the difference sometimes is very small) affects

TABLE 6.1: Examples of tweets that might have negative impact on users

Tweet Content	***Emotion Extraction***	***E.D***
Europe facing its worst drought in 500 years – study https://t.co/jY0vqojAYC https://t.co/ZAXZslBekK	sadness: 0.603305, joy: 0.061472, fear: 0.160444, disgust: 0.040136, anger: 0.05097	1.1226546857542616
Spectator sues Nick Kyrgios over ’700 drinks’ accusation during Wimbledon final https://t.co/0xPsmIj9vF	sadness: 0.523502, joy: 0.060197, fear: 0.034915, disgust: 0.124649, anger: 0.257796	1.1081969079211509
From the Cold War to the “war on terror” the United States has used Somalia as a battleground for its geopolitical schemes a with profoundly destructive consequences for Somalis	sadness: 0.516421, joy: 0.018284, fear: 0.327, disgust: 0.013176, anger: 0.044437	1.1501696659515066
Extreme “whiplash” between drought and floods makes it harder to recover from climate disasters	sadness: 0.634944, joy: 0.074562, fear: 0.181029, disgust: 0.006534, anger: 0.055189	1.1315847573085278
RT @Seeker: Yelp will flag crisis pregnancy centers listings to avoid misleading abortion seekers https://t.co/Gs5ALAM5mc https://t.co/p85d..	sadness: 0.162586, joy: 0.403488, fear: 0.29928, disgust: 0.097456, anger: 0.067443	1.2540924644562697

IBM’s final accuracy. The third kind of error is due to the vocabulary and style used in the language of fairy tales which is, arguably, significantly different from the contemporary use of language. These errors might be reduced by expanding our lexicon, However, it is uncertain whether this would make the algorithm sufficiently powerful to take it out of the context of fairy tales and into the “wilderness” of the Web. We have gathered a collection of tweets that we intuitively thought might hurt users emotions, some of which we included in TABLE 6.1(We have done the same process with content that we thought might be emotionally positive to certain users, results are included in TABLE 6.2). The results demonstrate the efficiency and effectiveness of our approach in filtering any suspicious content that may cause emotional harm to end users and by consequence protect users mental health to a certain extent.

TABLE 6.2: Examples tweets that might have positive impact on users.

Tweet Content	***Emotion Extraction***	***E.D***
PLA's participation in the Vostok-2022 drills aims to deepen practical and friendly cooperation with the militaries of participating countries, enhance strategic coordination and strengthen the ability to deal with various security threats: Defense Ministry spokesperson	sadness: 0.02296, joy: 0.932942, fear: 0.118979, disgust: 0.023001, anger: 0.024796	0.1321871190055975
Three years ago today @EmmaBostian and I met for the first time, Hoping we get to spend Christmas together this year in Sweden	sadness: 0.200453, joy: 0.863676, fear: 0.023482, disgust: 0.017436, anger: 0.009059	0.23757168240133333
Congratulations to my PhD adviser @geoffreyhinton for winning the Royal Medal! Totally unsurprising :)	sadness: 0.233232,joy: 0.676328, fear: 0.018988,disgust: 0.019905,anger: 0.019745	0.4075833218557894
Have the courage to walk away from the drama and people who create it around you. It's most often about them and not you. Surround yourself with those who support you & believe in you. Focus on the positive. Love those who treat you well, pray for the ones who don't. They need it.	sadness: 0.134433, joy: 0.625341, fear: 0.081753, disgust: 0.018836, anger: 0.097448	0.412062464684664

Based on the findings displayed in the table below, we observed that the Euclidean distance calculated for every tweet did not exceed the threshold we had set. This indicates that the emotional classification of these tweets is positive and the content is emotionally safe, and thus acceptable to be displayed. This finding is a significant step forward in our efforts to develop a filtering mechanism that can effectively eliminate emotionally harmful content from social media platforms.

During our analysis of tweets, we observed that the joy value was consistently higher than other negative emotions such as sadness, anger, disgust, and fear. This supports the positive results we obtained. We also examined each sentence in the tweets individually and found that the content conveyed positive emotions. For instance, the first tweet highlighted the PLA's participation in drills with other militaries to deepen cooperation, which is good news. The second tweet described the feeling of a first meeting with a loved one, while the third tweet was a message of congratulations. Finally, the fourth tweet was a motivational message and offered helpful advice for feeling better about oneself. Overall, these findings suggest that our emotional analysis of tweets can provide valuable insights into the positive emotions conveyed in social media content. To conclude this observation, we observed that tweets with a significantly higher joy value than other emotions were automatically classified as positive and emotionally safe based on our established threshold for the Euclidean distance.

6.4 Conclusion

Artificial intelligence (AI) has gained a lot of attention in recent years due to its potential to imitate human cognitive processes and solve complex computational and analytical problems. However, it is easy for individuals who are not familiar with AI technologies to assume that these systems can effortlessly extract valuable insights from data. In this article, we presented a solution for creating an emotion filter that detects and monitors harmful based-text emotions extracted from tweets. Our approach involved a classification algorithm that receives a text tweet as input and classifies it according to five emotional types. The resulting emotional array can then be used to determine the emotional type by calculating the Euclidean distance.

Numerous academics have employed various techniques in previous studies on extracting emotions from text, with many recent studies using supervised and unsupervised classifiers based on machine learning. For instance, (Yuan et al., 2016) used unsupervised techniques to measure similarity in their research on sentence-level and review-level categorization, utilizing the Scikitlearn program, a Python-based accessible software library. However, in our study, the IBM API analysis and similarity calculation using the Euclidean distance performed better than other machine learning methods, eventually outperforming the Scikit-learn program.

Therefore, for our study, we decided to categorize tweets into their relevant sentiment and emotion classes using the score we got from calculating the Euclidian distance. In addition, our contribution is based on the way IBM natural language understanding API libraries are created and how they are used in conjunction with heuristic rules.

Our approach is illustrated through a software system, IBM API. Besides the affect recognition engine, IBM's natural language understanding API is, to the extent of our knowledge, the most reliable free open-source textual emotion recognition API published on the web. The evaluation study showed promising results in terms of high classification accuracy, highlighting the importance of text emotion recognition. Future efforts will be centred on refining the algorithm in the IBM natural language understanding API software for better accuracy.

Chapter 7

Conclusion and future works

Summary

Conclusion and future works

7.1 General summary of the thesis

An overview of the planned contributions of this thesis, the findings of these studies and upcoming research are presented in this section. In this thesis, we enhanced the BDI model of agency logic by including Fear emotion modalities and proposed a new model of emotion triggers that can be read computationally. Agent's formal verification has been implemented using a verification tool NuSMV model checker. The effectiveness and advantages of the proposed approach over previously published work have been demonstrated through an experimental comparison. The second contribution presented in this thesis is an innovative system designed to eliminate social media content that may be perceived as emotionally harmful by end users. This system is built on the IBM Natural Language Understanding API, which provides advanced language analysis capabilities to identify and categorize emotional content in social media posts. By integrating this API into the filtering mechanism, the system is able to effectively detect and remove potentially harmful content before it reaches the user. This represents a significant advancement in the field of sentiment analysis and emotional content filtering, as it provides a powerful tool for protecting users from harmful content in real-time. Furthermore, the integration of the IBM Natural Language Understanding API also enables the system to adapt and learn over time, continuously improving its accuracy and effectiveness. Overall, the second contribution presented in this thesis demonstrates the potential for advanced technology to positively impact social media usage and improve the well-being of users.

The main goal of this PhD thesis is to implement a verification mechanism for the interactive (emotional), intelligent system so it better cope with unpleasant situations. The contribution of this thesis is based on new developed approach and existing ones. The proposed approach is scalable; it can be applied in any BDI agent, regardless of its application, making the building of intelligent virtual agents with effective skills easier. The main contributions of this thesis can be summarized as follows:

- ***The extension of the BDI model of agency:*** As part of the research presented in this thesis, the BDI model was selected as the basis for the proposed approach. This decision was made due to the model's robust behavior, rapid development, and most importantly, its extension capability. The BDI model has proven to be an effective approach for developing interactive and intelligent systems, providing a strong foundation for the proposed approach. Furthermore, the model's extension capability allowed for the incorporation of additional features and functionality, which was a crucial aspect of the proposed approach. In particular, the BDI model was extended to incorporate the Fear emotion using the OCC theory. This extension enabled the proposed approach to effectively reason about emotional states and responses, enhancing the overall effectiveness and reliability of the system. By incorporating the Fear emotion, the proposed approach was able to effectively handle emotionally charged situations, providing a valuable insight into the potential applications of interactive and intelligent systems in complex and dynamic environments

- ***The implementation of the formal verification***: In order to ensure the proper verification of all properties within the system, a formal verification checker was implemented as a crucial part of the research presented in this thesis. This involved the use of advanced tools and techniques such as model checking, which provided a comprehensive and rigorous approach to verifying the system's properties. Specifically, the NuSMV tool was utilized in combination with CUDD linked to the MiniSat SAT solver, which enabled efficient and effective verification of the system. These tools and techniques allowed for a thorough analysis of the system's behavior, ensuring that all properties were verified and that the system functioned as intended. This represents a significant advancement in the field of formal verification, as it demonstrates the effectiveness of advanced tools and techniques in verifying complex interactive and intelligent systems. The implementation of formal verification represents a crucial aspect of the research presented in this thesis, ensuring the reliability and effectiveness of the proposed system.
- ***The Simulation of case studies:*** In order to provide a comprehensive demonstration of the functioning concept of the proposed approach, two scenarios were implemented as part of the research presented in this thesis. The first scenario involved an auction environment, which was used to illustrate the construction of the BDIE model and BDIE logic specifications. This scenario demonstrated the ability of the proposed approach to effectively reason about multiple agents and their interactions in a complex environment, while also highlighting the importance of formal verification in ensuring the correctness and effectiveness of the approach. The second scenario involved a simulation of an aircraft maintenance environment, where the fearful agent was able to reason about various knowledge and temporal properties of the plane and flight agents. This scenario demonstrated the potential for the proposed approach to be applied in real-world contexts, highlighting its versatility and effectiveness in a practical setting. By simulating a complex

environment with multiple agents and dynamic interactions, the scenario provided a valuable insight into the potential applications of the proposed approach in a variety of different contexts. These two scenarios represented a critical aspect of the research presented in this thesis, demonstrating the effectiveness and versatility of the proposed approach in a variety of different settings. The scenarios also highlighted the importance of formal verification in ensuring the reliability and correctness of the approach, while providing valuable insights into the potential applications of interactive and intelligent systems in complex real-world environments.

The experiments are conducted on widely used benchmarks, and comparisons are made with widely used techniques in the literature. The outcomes demonstrate the advantages of each suggested strategy. To improve the suggested contribution, the work of this thesis can be expanded and improved. It is easy to create intelligent virtual agents with useful skills since the proposed BDI agent is ready to be implemented in just about any BDI agent, regardless of its application.

7.2 Future works

Looking ahead to future works, there are several areas of potential development and expansion for the research presented in this thesis. One of the primary goals is to further enhance the fearful agent's emotional capabilities, so that it can better accommodate a wider range of emotions beyond fear. This would involve extensive research into the specific emotions that the agent should be able to detect and respond to, as well as the most effective methods for implementing these capabilities. Additionally, the inclusion of more human parameters such as personality, beliefs, psychology, and other relevant factors would allow for a more nuanced and comprehensive understanding of how the agent interacts with humans in different contexts. This would require significant research into the various factors that influence human behavior and decision-making, as well as the development of new techniques and tools to integrate these parameters into the agent's decision-making processes. Overall, these future works have the potential to significantly improve the effectiveness and versatility of the fearful agent, and to further advance research in the field of interactive and intelligent systems.

Note on publications

The research materials presented in this thesis have been previously published in various prestigious conferences and scientific journals within the academic community. These publications have provided significant contributions to the relevant fields, expanding the knowledge base and furthering the advancement of research in this area. The research has been thoroughly peer-reviewed, and the resulting publications have received positive feedback and recognition from other experts in the field. Additionally, the presentation of the research in these conferences has provided a platform for discussion, collaboration, and the exchange of ideas among researchers, promoting further development of the field. Overall, the multiple publications of this thesis's research materials signify the relevance and importance of the research, as well as the author's commitment to sharing their findings with the broader scientific community.

International journals with peer review

- Benrouba, F., & Boudour, R. (2022). A Model Combining BDI Logic and Temporal Logics for Decision-Making in Emergency Situations. International Journal of Advances in Soft Computing & Its Applications, 14(3).
- Benrouba, F., & Boudour, R. (2023). Emotional sentiment analysis of social media content for mental health safety. Social Network Analysis and Mining, 13(1), 17.

International conferences with peer review

- Benrouba, F., & Boudour, R. (2022). Emotional sentiment analysis of twitter posts. Fifth Artificial Intelligence Doctoral symposium (AID'2022°).
- Benrouba, F., & Boudour, R. (2022). Emotion Analysis of Social Media Content for Mental Health Improvement. International Conference on Information Systems and Advanced Technologies (ICISAT).

National conferences with peer review

- Benrouba, F., & Boudour, R. (2021). Integrating human parameters in intelligent agents using BDI model of agency: A survey. First national conference on artificial intelligence and information technologies (CNIAT20).

Bibliography

Akhtar, N., Ghori, A. S., & Salamat, N. (2014). REQUIREMENT ANALYSIS, ARCHITECTURAL DESIGN AND FORMAL VERIFICATION OF A MULTI-AGENT BASED UNIVERSITY INFORMATION MANAGEMENT SYSTEM. In *International Journal of Computer Science & Information Technology (IJCSIT)* (Vol. 6, Issue 5).

António Damásio. (1994). *Descartes' Error: Emotion, Reason, and the Human Brain*.

Arend Hintze, M. S. U. (2016). *Understanding the Four Types of Artificial Intelligence*.

Baier, Christel., & Katoen, J.-Pieter. (2008). *Principles of model checking*. MIT Press.

Balodi, A. (2020). *Application of Introduction Artificial Intelligence Machine Learning in Real Life Introduction Artificial Intelligence Machine Learning Tools & Frameworks Application of AI and ML Questions and Answers Re 'Application of Artificial Intelligence and Machine Learning in Real Life' by Introduction Artificial Intelligence Machine Learning Tools & Frameworks Application of AI and ML Questions and Answers Re*. https://www.researchgate.net/publication/340684782

Benrouba, F., & Boudour, R. (2022). A Model Combining BDI Logic and Temporal Logics for Decision-Making in Emergency. *International Journal of Advances in Soft Computing and Its Applications*, *14*(3), 32–48. https://doi.org/10.15849/IJASCA.221128.03

Benrouba, F., & Boudour, R. (2023). Emotional sentiment analysis of social media content for mental health safety. *Social Network Analysis and Mining*, *13*(1), 17. https://doi.org/10.1007/s13278-022-01000-9

Bentahar, J., Yahyaoui, H., Kova, M., & Maamar, Z. (2013). Symbolic model checking composite Web services using operational and control behaviors. *Expert Systems with Applications*, *40*(2), 508–522. https://doi.org/10.1016/j.eswa.2012.07.069

Booch, Grady., Rumbaugh, James., & Jacobson, Ivar. (1999). *The unified modeling language user guide*. Addison-Wesley.

Bratman, M. E. (1987). Intention, Plans, and Practical Reason. *The Philosophical Review*, *100*(2), 277. https://doi.org/10.2307/2185304

Bresolin, D., & Villa, T. (n.d.). *CUDD Colorado University Decision Diagram Package Software per Sistemi Embedded Corso di Laurea in Informatica*.

Carroll, N. (2021). *Augmented Intelligence: An Actor-Network Theory Perspective*. https://www.researchgate.net/publication/351069721

Cimatti, A., Clarke, E., Giunchiglia, E., Giunchiglia, F., Pistore, M., Roveri, M., Sebastiani, R., & Tacchella, A. (2002). *NuSMV 2: An OpenSource Tool for Symbolic Model Checking* (pp. 359–364). https://doi.org/10.1007/3-540-45657-0_29

Cimatti, A., Clarke, E., Giunchiglia, F., & Roveri, M. (n.d.). *NUSMV: a new Symbolic Model Verifier*. http://nusmv.irst.itc.it/.

Colby, B. N., Ortony, A., Clore, G. L., & Collins, A. (1989). The Cognitive Structure of Emotions. *Contemporary Sociology*, *18*(6), 957. https://doi.org/10.2307/2074241

D'Souza, M., & Kashi, R. N. (2019). Avionics self-adaptive software: Towards formal verification and validation. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11319 LNCS*, 3–23. https://doi.org/10.1007/978-3-030-05366-6_1

E. Cerny, & X. Song. (1997). *Temporal Logic Temporal Logics*.

Ekman, Paul. (2003). *Emotions revealed : recognizing faces and feelings to improve communication and emotional life*. Times Books.

Erik Seligman, Tom Schubert, & M V Achutha Kiran Kumar. (2015). *Formal Verification: An Essential Toolkit for Modern VLSI Design*.

Frigg, R. (1920). *Models and Theories; A Philosophical Inquiry*.

Gödel, K. (1931). *On formally undecidable propositions of Principia Mathematica and related systems I.*

Hoare, C. A. R., Graham, S. L., & Rivest, R. L. (1978). *Programming Teclmiques Communicating Sequential Processes*.

IBM Cloud Education. (2020). *What is artificial intelligence (AI)?*

Khemlani, S. S. (2018). Reasoning. In *Stevens' Handbook of Experimental Psychology and Cognitive Neuroscience* (pp. 1–45). Wiley. https://doi.org/10.1002/9781119170174.epcn311

Lee, S. Y., Byeon, S., Kim, H. S., Jin, H., & Lee, S. (2021). Deep learning-based phase prediction of high-entropy alloys: Optimization, generation, and explanation. *Materials and Design*, *197*. https://doi.org/10.1016/j.matdes.2020.109260

Maffei, C., Roder, E., Cortesan, C., Passera, F., Rossi, M., Segrini, M., Visintini, R., & Fossati, A. (2014). Kinematic Elicitation of Basic Emotions: A Validation Study in an Italian Sample. *Psychology*, *05*(09), 1065–1078. https://doi.org/10.4236/psych.2014.59119

Mccarthy, J. (2007). *WHAT IS ARTIFICIAL INTELLIGENCE?* http://www-formal.stanford.edu/jmc/

Minsky, M. (1965). *MATTER, MIND AND MODELS*.

Mohamed Ridhwan, K., & Hargreaves, C. A. (2021). Leveraging Twitter data to understand public sentiment for the COVID-19 outbreak in Singapore. *International Journal of Information Management Data Insights*, *1*(2). https://doi.org/10.1016/j.jjimei.2021.100021

Molina, M. (2020). *What is an intelligent system?* http://arxiv.org/abs/2009.09083

MURATA, T. (1989). *Petri Nets: Properties, Analysis and Appl k a t ions TADAO MURATA, FELLOW, IEEE Invited Paper*.

NASA Johnson Space Center. (2004). *Error Cost Escalation Through the Project Life Cycle*.

Neviarouskaya, A., Prendinger, H., & Ishizuka, M. (2009). *Compositionality Principle in Recognition of Fine-Grained Emotions from Text*. http://www.connexor.eu/technology/machinese/machinesesyntax/

Ortony, A., Clore, G. L., & Collins, A. (1988). *The Cognitive Structure of Emotions* (A. Ortony, Clore Gerald L., & A. Collins, Eds.). Cambridge University Press. https://doi.org/10.1017/CBO9780511571299

Pereira, D., Oliveirat, E., Moreira, N., & Sarmentot, L. (n.d.). *Towards an Architecture for Emotional BDI Agents*.

Perháč, J., Novitzká, V., Steingartner, W., & Bilanová, Z. (2021). Formal model of IDS based on BDI logic. *Mathematics*, *9*(18). https://doi.org/10.3390/math9182290

Politechnica University of Bucharest. (2002). *Architectures des agents et langages*.

Rao, A. S., & Georgeff, M. P. (1995). *BDI Agents: From Theory to Practice*. www.aaai.org

Reid, D. A., & Knipping, Christine. (2010). *Proof in Mathematics Education Research, Learning and Teaching.* BRILL.

Seshia, S. A., & Subramanyan, P. (2018). UCLID5: Integrating Modeling, Verification, Synthesis and Learning. *2018 16th ACM/IEEE International Conference on Formal Methods and Models for System Design (MEMOCODE)*, 1–10. https://doi.org/10.1109/MEMCOD.2018.8556946

Somenzi, F. (2018). *CUDD: CU Decision Diagram Package Release 2.7.0*.

Stuart J. Russell, & Peter Norvig. (1995). *Artificial Intelligence A Modern Approach Third Edition*.

Su, Y., Hu, B., Dai, Y., & Rao, J. (2018). *A Computationally Grounded Model of Emotional BDI-Agents* (pp. 444–453). https://doi.org/10.1007/978-3-319-95930-6_41

The Embedded Systems Unit in the Digital Industry Center at FBK-IRST, The Model Checking group at Carnegie Mellon University, the M. R. G. at U. of G., & The Mechanized Reasoning Group at University of Trento. (n.d.). *NuSMV: a new symbolic model checker*. Https://Nusmv.Fbk.Eu/.

Thomas Miiller. (2011). *The continuum companion to philosophical logic*.

Thomson, A. (2009). *Critical Reasoning*. Routledge. https://doi.org/10.4324/9780203870990

Turing, A. M. (1950). Computing Machinery and Intelligence. In *Parsing the Turing Test* (pp. 23–65). Springer Netherlands. https://doi.org/10.1007/978-1-4020-6710-5_3

Yau, K.-L. A., Lee, H. J., Chong, Y.-W., Ling, M. H., Syed, A. R., Wu, C., & Goh, H. G. (2021). Augmented Intelligence: Surveys of Literature and Expert Opinion to Understand Relations Between Human Intelligence and Artificial Intelligence. *IEEE Access*, *9*, 136744–136761. https://doi.org/10.1109/ACCESS.2021.3115494

Yuan, S., Huang, H., & Wu, L. (2016). Use of Word Clustering to Improve Emotion Recognition from Short Text. *Journal of Computing Science and Engineering*, *10*(4), 103–110. https://doi.org/10.5626/JCSE.2016.10.4.103