University of Bergen
Department of informatics

Finding Common Grounds: The Moral Machine Case

Author: Anum Rehman
Supervisor: Ana Ozaki

UNIVERSITETET I BERGEN
Det matematisk-naturvitenskapelige fakultet

March, 2023

Acknowledgements

First of all, I would like to thank my supervisor, Ana Ozaki for her guidance and support throughout my thesis. Her invaluable involvement and encouragement is highly appreciated. .

I am indebted to her for all the solution-oriented discussions, sharing of ideas, patience, and regular follow-up. Her aid helped me in shaping up my work, aligned it with my interests, resulted in a smooth journey.

I also would like to thank my co supervisor, Philip Andreas Turk, who always facilitated my queries with the best possible answers.

I highly value the support that I received from my family and friends in fulfilling my dream.

Last but not least, I thank my beloved husband, Shoaib for providing me with his endless support and encouragement. I could not have undertaken this journey without his love and cooperation. I would like to thank University of Bergen for providing me the platform, resources to attain this academic achievement and turning me into a valuable part of the society.

`Anum Rehman`

Bergen, 2023

UNIVERSITY OF BERGEN

Faculty of Mathematics and Natural Sciences

Department of Informatics

Abstract

Autonomy is when one tackles to gain a sense of oneself as a detached, self-governing individual. With the rapid development of Artificial Intelligence, there comes a concern about how autonomous systems will make principled decisions. It demands imagining several potential outcomes for the future, anticipating possible problems and setting a course of actions to minimize the danger to the society; while good factors reliability. We need to consider the dynamic standards of dangers and reliabilities. These change from one society to another, lead to disagreement, as these differ in various ways. The goal is to design an autonomous system in a way which can reach mutual understanding and settlement while resolving friction.

In this project, I am going to study and implement an algorithm to build common ground using horn expressions. Python programming language is used for its implementation. It resolves incoherence while giving optimal coherent constraints that meet all the agents' requirements, what we call a common ground. This algorithm combines incoherent behaviour rules represented in propositional Horn.

I improve the algorithm with a proposal to resolve conflict. It is a settlement of internal conflicts which can enhance the potential of autonomy. It enforces parties to negotiate based on a framework of negotiations. Additionally, I reduce a number of known limitations in the algorithm for reaching a common ground.

I study and explore the Moral Machine experiment. It is an online experimental platform, designed to explore the moral dilemmas faced by autonomous vehicles. Moral dilemmas are important. These help to investigate individuals with their moral standing, in terms of the choices they make when presented with conflicting situations.

We apply the implemented algorithm on the moral recommendation extracted from Moral Machine experiment, where each recommendation is represented as a Horn clause.

For example, human agents strongly prefer sparing more lives to fewer in case vehicle brakes fail. This dataset is gathered from all over the world. It is highly expected to see some disagreement based on users' geographical location and demographic information such as country, education, social status, religious and political attitudes. For testing algorithm, we use combinations of different countries and discuss the results.

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

This chapter introduces the motivation and context for this thesis, leading to its research questions and research goals, followed by a summary of prior and related works.

1.1 Motivation

> "Ethics is knowing the difference between what you have a right to do and what is right to do." – said **Potter Stewart**

Ethical decisions inspire trust, fairness, responsibility, and care for others. The process of ethical decision-making recognizes these conditions. It requires a review of all available options, eliminate unethical views and choose the best ethical alternative.

Determining what is 'good' means that one needs to know what are the underlying values. Each person and socio-cultural environment prioritizes and interprets moral and societal values differently [1]. Most people can act in ways that enhance or decrease the quality of their lives or the lives of others. They generally know the difference between helping and harming. However, when facing certain dilemmas or decisions, each person draws on their standards based on their values, which results in people making different decisions in similar situations [1]. Commonly, ethical differences are the result of individual interpretations of the situation at hand, which can lead to conflicting situations.

This implies that when we tend to design a system that is intelligent enough to make decisions, we ought to consider many environmental factors and the possibility of the emergence of conflicts. The more decisions such systems make for us, the more we need to ensure that the decisions they make have a positive individual and societal ethical impact [2].

A moral autonomous system is an agent who completes a task by using a decision-making model. It follows an ethical framework by adhering to a moral principle in an ethical context. Systems capable of some level of autonomous operation should be built to respect the moral norms and values of the society in which they operate [3]. These should function without any assistance, which is one of the ultimate goals of artificial intelligence. The main concern is the standards these have to perceive as norms and values. For this, we need to reinforce a set of rules. These rules should deal with inconsistency and disagreement while reaching a common mutuality, that a system can follow. This mutuality must have respected all stakeholders' recommendations as much as it is possible for the system.

To illustrate artificial intelligence ethical dilemma, consider autonomous vehicle situations presented by Massachusetts Institute of Technology researchers in the Moral Machine Experiment. The experiment is designed to explore the multidimensional moral dilemmas faced by autonomous vehicles. Autonomous vehicles have the potential to improve the quality and productivity of the time spent in cars, increase the safety and efficiency of the transportation system, and transform transportation into a utility available to anyone, anytime [4].

1.2 Research goals

Main goal of the thesis is to understand and implement an algorithm for building common grounds, by resolving inconsistency among rules. This algorithm has been developed to solve the problem of normative incompatibility using a propositional Horn expression, which is the combination of Horn clauses.

This is a study of the problem of reaching a settlement among the rules of conduct an autonomous system should obey. Our algorithm combines possibly incoherent behavior rules represented in propositional Horn expressions. In the algorithm, we assume that each stakeholder with an interest in governing the behavior of a system contributes with a set of rules, which the system should implement. However, these rules are possibly under-specified and might be incoherent with each other. This algorithm "corrects" the rules with exceptions raised by stakeholders and finds a mutually acceptable common ground. The algorithm has some specific notions related to Horn expressions such as coherence, conflict, acyclicity, and redundancy. These are to be fulfilled to end up with a full common ground. Any autonomous system can adapt that full common ground and work in normative limits by considering social and moral tendencies.

In the thesis, we also propose some changes to the algorithm while reducing the limitations of algorithm for reaching a full common ground. These changes improve the applicability of the algorithm. After that, we implement the common ground algorithm.

We use our algorithm and test it on the Moral Machine data set [5]. Moral Machine is a platform for gathering a human perspective on moral decisions made by self-driving cars, where self-driving cars have to make decisions in life-threatening situations. These show moral dilemmas, where a driverless car must choose the lesser of two evils. For example, what should a self-driving car do if it has to decide between the lives of passengers or pedestrians in the event of collision? Should it consider the age of the people? Or the number of people involved? Are humans more important than animals? Should one child be saved instead of two adults? These are some of the scenarios that are at the center of discussion in the Moral Machine experiment.

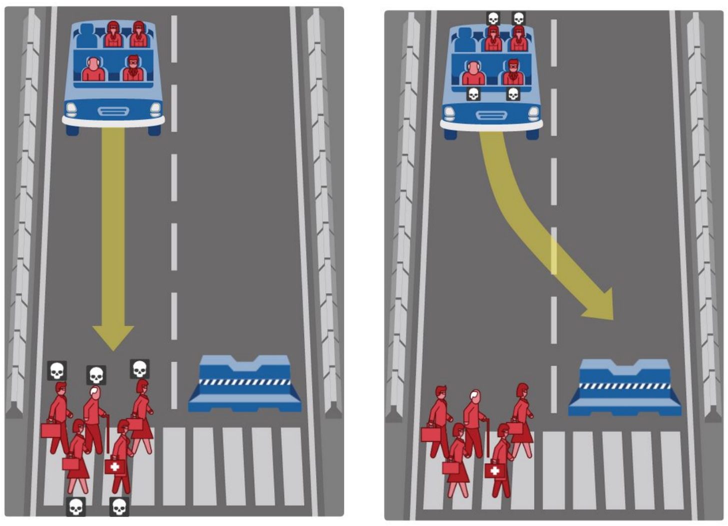


Figure 1.1: Example of a driving scenario

`https://MoralMachine.mit.edu/`

In Figure 1.1, we see that if an Autonomous Vehicle such as a self-driving car faces breaks failure and the car does not intervene, pedestrians would be killed and if the car intervenes, passengers would be killed. In this experiment, both pedestrians and passengers have some specific attributes such as being old or young and having high social status (e.g., doctor or pregnant women) or low social status (e.g., criminal or jobless person). This experiment has 13 such dilemmas that user has to decide. This platform gathered 40 million decisions in ten languages from millions of people in 233 countries and territories, which is the data set of the Moral Machine experiment [5]. We use this dataset and test our algorithm by incoherence and conflicts.

Our main contributions are:

- analysis of common ground algorithm and the necessary changes to minimize the main limitations of Algorithm 1 (Ozaki et al. [6]).
- discussion of existence of conflicting rules in concrete dataset such as the one in the Moral machine experiment and how these can be resolved.
- an implementation of our algorithm for finding common grounds.
- the case study of moral recommendation in Moral Machine experiment.
- test algorithm on Moral Machine dataset (after extracting rules) and analysis of results for different countries.

1.3 Plan

Thus far, this chapter introduces the motivation and research goal of the work described in this thesis. The remaining structure is as follows:

Chapter 2 presents an overview of the background. This includes a brief description of the syntax and semantics needed to describe the algorithm. It has an explanation of the notions with examples that are used in the algorithm such as Horn expression, coherence, and conflict. It also covers a brief introduction of the original version of the algorithm, with some related theorems and lemmas.

Chapter 3 encompasses research around extended changes in the algorithm like the resolution of conflict, cycles and redundancy. It describes how changes have been used to support the algorithm. It also covers the new version of the common ground algorithm, along with the elaborations of examples.

Chapter 4 presents the implementation of the algorithm. It explains the architecture of all the active component. It also covers how the technical work is implemented in Python programming language.

Chapter 5 presents a case study of the Moral machine experiment while explaining the Moral machine dataset and preprocessing the data to make it suitable for our algorithm. It also contains discussion on how we applied the developed algorithm in the previous sections to find common grounds for Horn expressions representing moral recommendations. The moral recommendations are taken from the dataset of the Moral Machine Experiment. We use combinations of different countries to test our algorithm.

Chapter 6 presents a literature review of works related to the problem we are attempting to solve.

Chapter 7 presents a summary of the work while discussing some limitations and challenges. It also contains suggestion of potential directions for further development.

Chapter 2

Background

In this chapter, we present a brief overview of the main concepts this thesis is based upon. We start with basic details of the syntax. Following this, we continue with a discussion on essential notions like Horn expression, coherence, conflict, and common ground. We also use examples to explain these notions.

Additionally, we discuss some theorems, lemmas and postulates needed to develop an understanding about the algorithm for finding common ground (Ozaki et al. [6]). We discuss notions and terms, both mathematically and in simple language. Finally, we give a brief introduction to the original version of the common ground algorithm [6] and explain how it works using examples.

2.1 Syntax

Syntax is a formal description of the structure of expressions in any given language. It is important to remember, whenever we talk about syntax, it reflects symbols. We are not mentioning their meaning, which is the role of semantics.

In propositional logic, we say an atom is a boolean variable or a propositional symbol that can be true or false. A literal is an atom or its negation.

- A positive literal is just an atom (e.g., x).
- A negative literal is the negation of an atom (e.g., $\neg x$).

In propositional logic, we apply the following operators on formulas. All literals are formulas and we can build more complex formulas using operators as follows. Assume f and g are formulas. Then:

- Negation: $\neg f$
- Conjunction: $f \wedge g$
- Disjunction: $f \vee g$
- Implication: $f \rightarrow g$

are also formulas.

2.1.1 Horn Expression

In propositional logic, a clause is a finite disjunction of literals (atoms or their negations) such as $(l_1 \vee l_2 \ldots \vee l_n)$, where $l_1, l_2 \ldots l_n$ can be positive or negative literals.

A Horn clause is a clause containing at most one positive literal.

- Example of a Horn Clause: ($\neg$ **Child** $\vee \neg$ **Mail** $\vee$ **Boy**)
- Not a Horn Clause: (**Rain** $\vee$ **Sleet** $\vee$ **Snow**)

Equivalently, we can write a Horn clause in implication form using equivalence $(A \ \vee \neg B \equiv \ A \leftarrow B)$ which is more common in logic programming. We can write

($\neg$ **Child** $\vee \neg$ **Mail** $\vee$ **Boy**) as (**Child** $\wedge$ **Mail**) $\rightarrow$ **Boy**

where we know that **Child** and **Mail** are negative literals and **Boy** is a positive literal. There are two types of Horn clauses.

- **Definite Horn Clause**: Horn clause with exactly one positive literal. For example, $(p_1 \wedge \ p_2 \ldots \wedge p_n \rightarrow q)$.

- **Horn Constraints**: Horn clause without a positive literal. For example, $(p_1 \wedge p_2 \ldots \wedge p_n) \rightarrow$ **False** or it can be written as $(p_1 \wedge p_2 \ldots \wedge p_n) \rightarrow \perp$.

For a definite Horn clause ϕ, we define antecedents written as $ant(\phi)$ to be the set of all atoms such that their negation occurs in ϕ, while the concludent $con(\phi)$ is the positive literal in ϕ. Definite Horn clauses have only one concludent. Let ϕ be the Horn clause as $(p_1 \wedge \ p_2 \ldots \wedge p_n) \rightarrow q$ then $ant(\phi) := \{p_1, p_2 \ldots p_n\}$ and $con(\phi) := q$. A Horn expression is a (finite) collection of Horn clauses. It is called definite Horn expression if all clauses in it are definite.

2.2 Semantics

Semantics is a formal description of the meaning of programs in a given language. There are various types of meanings that might be assigned, but the basic one we use in propositional logic is truth and falsity.

Semantics is a set of truth functions that take in a set of variables along with some logical connective and return either true or false For example in a truth table, each row corresponds to a particular truth function, whatever a truth function assigns to a string X, it must assign the opposite value to the string $(\neg X)$, and that $(A \wedge B)$ is assigned true if and only if both A and B are assigned true.

The semantics is given by interpretations. An interpretation function $\boldsymbol{I}(f)$ returns:

- *true(1)* (says that it satisfies f)
- *false(0)* (says that it does not satisfy f)

For example, we have three variables a, b, c and formula f is defined as $a \vee b \wedge c$, suppose in interpretation, a, b are false and c is true, then in interpretation $\boldsymbol{I}$, given formula is not true.

2.2.1 Entailment and Satisfiability

In simple words, we say a sentence **A** entails another sentence **B** if, whenever **A** is true, **B** must also be true.

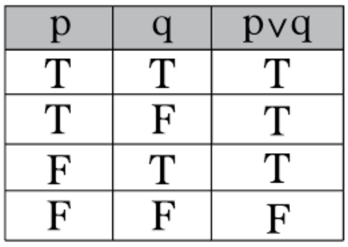
Formally, we say a set of premises Δ logically entails a conclusion Φ (written as $\Delta \models \Phi$) if and only if every interpretation that satisfies the premises also satisfies the conclusion. For example:

$$\{\mathbf{p}\} \models \{\mathbf{p} \vee \mathbf{q}\}$$

$$\{\mathbf{p}\} \not\models \{\mathbf{p} \wedge \mathbf{q}\}$$

$$\{\mathbf{p}, \mathbf{q}\} \models \{\mathbf{p} \wedge \mathbf{q}\}$$

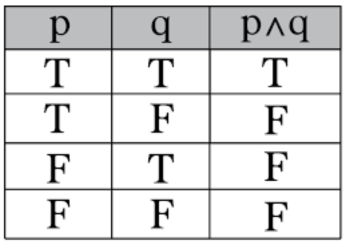
we can check for logical entailment by comparing tables of all possible interpretations.



p	q	p∨q
T	T	T
T	F	T
F	T	T
F	F	F

Figure 2.1: Truth table of Logical disjunction (**OR**)

In Figure 2.1 we can see that when we eliminate all rows that do not satisfy premises, we are only left with the rows where conclusion is also satisfied that means $\mathbf{p} \models \mathbf{p} \vee \mathbf{q}$.

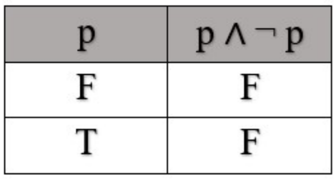


p	q	p∧q
T	T	T
T	F	F
F	T	F
F	F	F

Figure 2.2: Truth table of Logical Conjunction (**AND**)

Similarly, in Figure 2.2 if we eliminate all rows that do not satisfy premises $\mathbf{p}$, we have one row where conclusion $\mathbf{p} \wedge \mathbf{q}$ is not satisfied, means $\mathbf{p} \not\models \mathbf{p} \wedge \mathbf{q}$. Here we keep rows where premises, $\mathbf{p}$ and $\mathbf{p}$ both are satisfied, we have a row, where conclusion $\mathbf{p} \wedge \mathbf{q}$ is also satisfied, means $\{\mathbf{p}, \mathbf{q}\} \models \mathbf{p} \wedge \mathbf{q}$. In Horn logic, a Horn expression $\mathcal{F}$ entails a clause ϕ ($\mathcal{F} \models \phi$), if every interpretation that satisfies $\mathcal{F}$ also satisfies ϕ.

A sentence is said to be satisfiable Iff under some circumstances it can be true, on logical grounds. In propositional logic, a proposition $\mathbf{p}$ is "satisfiable" if its truth value is *true* for some assignments and is "unsatisfiable" if the truth value is *false* for all the assignments. We can say from both tables 2.1 and 2.2, $\mathbf{p} \vee \mathbf{q}$ and $\mathbf{p} \wedge \mathbf{q}$ both are satisfiable because some assignments are true in both cases.



p	p∧¬p
F	F
T	F

Figure 2.3: Truth table of unsatisfiable fromula

Here in table 2.3, ($\mathbf{p} \wedge \neg\ \mathbf{p}$) is unsatisfiable because none of the values are true. A popular Theorem is related from the explanation of logical entailment and satisfiability.

Theorem 1. $\Delta \models \Phi$ *if and only if* $\Delta \cup \{\neg\ \Phi\}$ *is unsatisfiable [7].*

Proof: Suppose that $\Delta \models \Phi$. If an interpretation satisfies Δ, then according to the definition of entailment it must satisfy Φ, this implies it cannot satisfy $\neg\Phi$. Therefore, $\Delta \cup \{\neg\ \Phi\}$ is unsatisfiable.

Conversely, suppose that $\Delta \cup \{\neg\ \Phi\}$ is unsatisfiable. Then every interpretation that satisfies Δ must fail to satisfy $\neg\Phi$, that is, it must satisfy Φ. Therefore, $\Delta \models \Phi$ [7].

Note: We can determine logical entailment by determining satisfiability. We are going to use that relation later in the thesis.

2.3 First Steps for Finding Common Grounds

First is to get an idea about agents that are going to use in further details. Then provide the notion of common ground along with some essential terms to define it.

2.3.1 Agents in Artificial Intelligence

In artificial intelligence, an intelligent agent is anything that perceives its environment, takes actions autonomously to achieve goals, and may improve its performance with learning and knowledge [8]. An agent could be, a coupling of a computational engine with physical sensors and actuators, called a robot, where the environment is a physical setting. It also can be a program that acts in a purely computational environment, a software agent [9]. These are autonomous entities, and we relate them whenever we want to design a system having a certain level of autonomy. The best examples of intelligent agents are *self-driving cars*, *Siri* and *Alexa*, that are virtual assistants.

In the thesis, I take in consideration multiple agents. Each agent is associated with its own set of behavior rules, which it wants to be implemented in the environment. In order to illustrate the main ideas and approaches that we follow in building our algorithm, consider an example from Ozaki et al. [6].

Example 2.3.1. A police robot (an intelligent agent) that patrols the streets and detects potentially illegal activity in a vicinity of a private house. Assume that the police robot has detected smoke and a child who is smoking. The robot has made the following deduction:

A child is smoking in a forbidden-to-smoke area.

Now assume we have stakeholders, which we may also refer to as "agents", with the following recommendations. The first agent puts forward the rule: "If there is an illegal activity, the police should be informed". In symbols:

$$\mathsf{illegalActivity} \rightarrow \mathsf{policeCall} \tag{2.1}$$

The second agent points out that it is a child who is smoking. So parents are the ones who should decide if the police should be informed. Police officers do independent judgment as a part of best serving the public. More specifically, we have the rule, "If there is an illegal activity conducted by a child, then parents should be called". In symbols:

$$(\mathsf{illegalActivity} \wedge \mathsf{child}) \rightarrow \mathsf{parentsAlert} \tag{2.2}$$

The agents agree that smoking in a forbidden-to-smoke area (for example, a bus stop) is an illegal activity. If this happens, either the police should be called or the parents should be alerted. They agree not to call both. It can be written as a Horn constraint:

$$(\mathsf{parentsAlert} \wedge \mathsf{policeCall}) \rightarrow \bot \tag{2.3}$$

Calling police and alerting parents is pointless. Since the police is obliged to call the parents of the minor.

In this example, there is a clear incoherence, but not necessarily a conflict between the two agents. One thinks that the police should be called, and other agent thinks that the parents should be alerted. We consider this incoherence because the second agent reasons using a more specific rule than the first. Rules could be even further specialized and taken into account the case in which the child is not under parental supervision (for example, alone in the supermarket or home alone).

Consider the rule: "If there is an illegal activity done by a child who is unsupervised, then the police should be called". In symbols:

$$(\mathsf{illegalActivity} \wedge \mathsf{child} \wedge \mathsf{unsupervised}) \rightarrow \mathsf{policeCall} \tag{2.4}$$

A common ground between (2.1) and (2.2) can be reached by transforming (2.1) into

$$(\mathsf{illegalActivity} \land \mathsf{adult}) \to \mathsf{policeCall} \tag{2.5}$$

An interesting fact about (2.5) is that it is coherent with both (2.2) and (2.4). When there are two applicable rules but one is more specialized [6]. So far, we have a basic sense behind the common ground. That is reaching a common mutuality, that a system can follow and it has given equal consideration to all agents.

2.3.2 Background Knowledge

Background Knowledge is a limitation about the facts of the world. For example, we can say "*we cannot stay healthy while eating unhealthy*" or "*a person cannot be at work and home at the same time*".

Suppose we have a statement that says, "*John Lasagna will be a little late for the party. He died yesterday*" [10]. From the basic constraints about the world, we know that John cannot attend the party because he is dead. Therefore, only one part of the sentence can be true. Symbolically it can be written as:

$$(\mathsf{isDead} \land \mathsf{attendsParty}) \to \bot \tag{2.6}$$

Above equation 2.6 is background knowledge in this scenario, as a Horn constraint. In above example 2.3.1, equation 2.3 is background knowledge of an agent, where **parentAlert** and **policeCall** are not recommended together. Similarly, each agent would refer to its unique background knowledge with a set of rules.

In formal language, consider i number of agents in a scenario, and each agent is associated with its own set of recommendations. These are defined as definite Horn expressions (a finite collection of definite Horn clauses). These recommendations are represented as definite Horn expressions, denoted by $\mathcal{F}_i$ for each agent i. These also have the background knowledge with a set of Horn constraints. The background knowledge $\mathcal{B}$ for $\mathcal{F}_1, \ldots, \mathcal{F}_n$ is defined as a set of Horn constraints, built from atoms/literals occurring in $\mathcal{F}_1, \ldots, \mathcal{F}_n$, representing pairwise disjointed constraints (e.g., a person cannot be a child and an adult, or a child and a teenager) [6].

Excludents of atom/Literal

Suppose we have background knowledge $\mathcal{B}$ as a collection of Horn constraints and Horn expression $\mathcal{F}$ as a collection of definite Horn clauses.

Mathematically, for a given atom p occurring in $\mathcal{B}$ and $\mathcal{F}$, excludent is defined as

$$\overline{p}_{\mathcal{B}} = \{q \mid \mathcal{B} \models (p \wedge q) \rightarrow \bot)\} \text{ [6]} \tag{2.7}$$

elements of $\overline{p}$ are called *excludents* of p. We assume that for every p occurring in $\mathcal{F}$ the set, $\overline{p}$ is not empty. It means that for each atom p, there is some background knowledge in $\mathcal{B}$ as a Horn constraint. So excludents of that p will contain all the atoms that are mutually disjoint with p in $\mathcal{B}$. Whenever $\overline{p}$ is written in a clause, it assumes to be pointing towards an atom in $\overline{p}$, which means "not p". We explain using examples.

Example 2.3.2. The background knowledge of the agents in Example 2.3.1 can be modeled as [6]:

$$\begin{aligned}
&(\mathsf{parentsAlert} \wedge \mathsf{policeCall}) \rightarrow \bot, \\
&(\mathsf{child} \wedge \mathsf{adult}) \rightarrow \bot, \\
&(\mathsf{child} \wedge \mathsf{teen}) \rightarrow \bot, \\
&(\mathsf{supervised} \wedge \mathsf{unsupervised}) \rightarrow \bot.
\end{aligned} \tag{2.8}$$

so, from this background knowledge, we can extract excludents of any atom using the above definition of excludents.

In our notation, we can say,

$$\begin{aligned}
&\overline{\mathsf{child}} = \{\mathsf{adult}, \mathsf{teen}\} \\
&\overline{\mathsf{supervised}} = \{\mathsf{unsupervised}\} \\
&\overline{\mathsf{parentsAlert}} = \{\mathsf{policeCall}\}
\end{aligned} \tag{2.9}$$

Example 2.3.3. From the equation 2.6, we concluded background knowledge as

$$(\mathsf{isDead} \wedge \mathsf{attendsParty}) \rightarrow \bot \tag{2.10}$$

from here, we can define excludent as:

$$\overline{\mathsf{is\_dead}} = \{\mathsf{attends\_party}\} \tag{2.11}$$

Notions of ϕ^{-p}, ϕ^{+p} and $\phi^{q\backslash p}$

Consider having background knowledge $\mathcal{B}$ as a collection of Horn constraints and Horn expression $\mathcal{F}$ as a collection of definite Horn clauses. Suppose p be an atom occurring in $\mathcal{F}$, and ϕ be a clause in $\mathcal{F}$.

We define:

- ϕ^{+p}: a clause after adding p to antecedents of ϕ.
- ϕ^{-p}: a clause after removing p from antecedents of ϕ.
- $\phi^{q\backslash p}$: the result of replacing $q \in ant(\phi)$ by p.

We explain using an example.

Example 2.3.4. Suppose we have a clause ϕ in a Horn Expression, defined as

$$\phi : (\mathsf{illegalActivity} \land \mathsf{child}\ ) \rightarrow \mathsf{policeCall}$$

Let p and q be two atoms in a Horn Expression defined as "unsupervised" and "child" respectively. Here antecedent is $ant(\phi) = \{\mathsf{illegalActivity},\ \mathsf{child}\}$, then we can define

- ϕ^{+p}: $(\mathsf{illegalActivity} \land \mathsf{child} \land \mathsf{unsupervised}) \rightarrow \mathsf{policeCall}$.
- ϕ^{-q}: $\mathsf{illegalActivity} \rightarrow \mathsf{policeCall}$
- $\phi^{q\backslash p}$: $(\mathsf{illegalActivity} \land \mathsf{unsupervised}\ ) \rightarrow \mathsf{policeCall}$

2.3.3 Coherence

The word "coherent" comes from the Latin word meaning "to stick together "[11]. The collection of arguments has coherence if all the parts have logical connections and all the arguments, collectively, flow together well without any inconsistency. Arguments and strategies have coherence when both are making sense together and having a consistent relationship with each other.

In Horn logic, coherence is a property of a Horn clause or a Horn expression. It occurs when, in a set of rules, one should not be able to infer an atom and any of its excludents at the same time. If it occurs, such expression will be called incoherent. In simple words, if union of antecedents of one of the two clauses is taken, and the concludents include both an atom and its excludents then these two clauses are incoherent.

Example 2.3.5. From example 2.3.1, there are two clauses as:

$$\mathsf{illegalActivity} \to \mathsf{policeCall}$$

$$(\mathsf{illegalActivity} \wedge \mathsf{child}) \to \mathsf{parentsAlert}.$$

the defined background knowledge is:

$$(\mathsf{parentsAlert} \wedge \mathsf{policeCall}) \to \bot \in \mathcal{B}.$$

If union of the antecedent of the clauses is taken then it implies both policeCall and parentsAlert. This is not possible according to defined background knowledge. Therefore, these two clauses are incoherent. Furthermore, if these two clauses are in a Horn expression then the Horn expression would be incoherent.

But if we have two clauses like

$$(\mathsf{illegalActivity} \wedge \overline{\mathsf{child}}) \to \mathsf{policeCall}$$

$$(\mathsf{illegalActivity} \wedge \mathsf{child}) \to \mathsf{parentsAlert}.$$

now it is obvious that we cannot combine antecedents of both clauses, as per background knowledge,

$$\overline{\mathsf{child}} = \{\mathsf{adult}, \mathsf{teen}\}$$

We take the excludents of a symbol in the antecedent of a clause and generate weaker versions of second clause by adding these excludents to its antecedent. Now this combination of clauses should become coherent (that is the key point of this Algorithm to solve incoherence).

If we find any of the rules by agents or incoherent, then it is understood to not include respect for both agents equally. We have to be with one, which is more specific and set well together with other agents.

Notion of derivation is introduced before defining coherence in a formal language.

Notions of Derivation

Let ϕ and ψ be definite Horn clauses and $\mathcal{F}$ be a definite Horn expression and $\phi, \psi \in \mathcal{F}$. We write $\psi \Rightarrow_{\mathcal{F}} \phi$ if ϕ is a consequence of ψ, or in other words, there is a derivation of ϕ

with respect to ψ and $\mathcal{F}$. we have a condition from Ozaki et al. [6] that is used to check if the derivation of one clause, with respect to another clause, exists or not.

$$\psi \Rightarrow_{\mathcal{F}} \phi \text{ Iff } \mathcal{F} \cup ant(\psi) \models \; ant(\phi) \tag{2.12}$$

Definition 1. *Let $\mathcal{F}$ be a definite Horn expression. A definite Horn clause ϕ is* **coherent** *with a Horn expression $\mathcal{F}$ if $\mathcal{F} \setminus \{\phi\} \not\models \phi$ and*

- *there is no $\psi \in \mathcal{F}$ such that $\psi \Rightarrow_{\mathcal{F}} \phi$ or $\phi \Rightarrow_{\mathcal{F}} \psi$ while $con(\psi) \in \overline{con(\phi)}$ (note that $con(\psi) \in \overline{con(\phi)}$ implies $con(\phi) \in \overline{con(\psi)}$).*

The set $\mathcal{F}$ is **coherent** *if all $\phi \in \mathcal{F}$ are coherent with $\mathcal{F}$ (and* incoherent *otherwise) [6].*

In simple words, a clause ϕ is coherent with the Horn expression $\mathcal{F}$ if it satisfies the following:

- If ϕ is removed from the Horn expression, the remaining expression still entails ϕ.
- There is not any other clause ψ in the Horn expression such that ϕ and ψ have mutual derivative for each other, while concludents of ψ belong to excludents of concludents of ϕ in $\mathcal{F}$ or vice versa.
- Verify all the clauses in the Horn expression. If all the clauses satisfy the condition then the given Horn Expression $\mathcal{F}$ is coherent.

Above mentioned steps will be used while implementing coherence in the next chapter.

2.3.4 Conflict

Conflict is a fundamental human and social attribute. A completely conflict-free, pleasant-sounding society is impossible. In simple words, conflict may be defined as incompatibility of interests, goals, values, needs, expectations, or social ideologies [12]. According to Michel Nicholson, who was an English Journalist, "conflict is an activity which takes place when individuals or groups wish to carry out mutually inconsistent acts concerning their wants, needs, or obligations". Conflict examples in the real world can be a dispute between friends or family members, labor strikes, competitive sports, or war.

In our scenario, if agents have behavior rules that seem to have clear friction. All cannot be considered equally, leads to conflict. These are not easily resolvable for reaching a mutual agreement.

In Horn logic, conflict is a property of a set of definite Horn clauses. The notion of conflict is stronger than the notion of coherence. A conflict is an incoherence that cannot be easily resolved [6]. A scenario is going to be used for narrating the conflict example, has already been used in case of coherence.

Example 2.3.6. Suppose the rules defined by different agents are given as:

$$\begin{aligned}
&(1) \quad \mathsf{illegalActivity} \rightarrow \mathsf{parentsAlert},\\
&(2) \quad \mathsf{illegalActivity} \rightarrow \mathsf{policeCall},\\
&(3) \quad (\mathsf{illegalActivity} \land \mathsf{child}) \rightarrow \mathsf{parentsAlert}.
\end{aligned}$$

the background knowledge is defined as:

$$(\mathsf{parentsAlert} \land \mathsf{policeCall}) \rightarrow \bot$$

Now we see the set with the first two rules, (1) and (2) is in conflict. While the set with the last two rules, (2) and (3) is not. This can be solved by adding an excludent of one of the antecedents of the more specific rule to the other's antecedents. In case of (1) and (2), both rules have same antecedents and different concludents. It means both rules are in complete opposition. The pairs of rules that can be in conflict when we cannot find a "suitable" atom to add to the antecedent of an incoherent rule, as a way to further specify it and avoid incoherence. The "suitable" atoms are chosen from the excludents of the atoms in the antecedents of the more specific rule involved in the incoherence. Whenever we find conflict in rules, it is considered to be difficult to reach a mutual agreement or a common ground while respecting all agents equally.

Now, we can define conflict in formally.

Definition 2. Let $\mathcal{F}$ be a definite Horn expression. We say that $\mathcal{F}$ is **in conflict** if we can find any pair of clauses $(\phi, \psi) \in \mathcal{F}$ such that

- $\phi \Rightarrow_{\mathcal{F}} \psi$ and $con(\phi) \in \overline{con(\psi)}$ (i.e., $\mathcal{F}$ is incoherent); and
- there is no $r \in ant(\phi) \setminus ant(\psi)$ with $q \in \overline{r}$ s.t. ψ^{+q} is coherent with $\mathcal{F} \setminus \{\psi\}$ [6].

If we can find one or more such pairs then Horn expression is in conflict.

In simple words, a Horn expression is in conflict if two clauses ϕ, ψ exist in the Horn expression that meets the following two conditions:

- If ϕ, ψ have incoherence for each other, then the Horn expression is incoherent.
- If we take the relative complement (set difference) of antecedents of both clauses, we cannot find any atom such that, we add an excludent of that atom to one of the clause's antecedents, this clause becomes coherent with the Horn expression. That can happen in two cases:
  - ✷ First, if the complement set is empty, that means both rules have the same antecedents.
  - ✷ Secondly, if the complement set is not empty. We can find an atom, by adding atoms' excludent to the clause's antecedents, it does not become coherent with the Horn expression. It means (ψ^{+q} *is coherent with* $\mathcal{F} \setminus \{\psi\}$) is not satisfied.
- At the end, if we successfully get any of two clauses from the Horn expression involved in a conflict, then the Horn expression is in conflict for sure.

2.3.5 Common Ground Postulates

We now define the notion of common ground. We also discuss and motivate the postulates that characterize a common ground.

Definition 3. *Let $\mathcal{F}_1, \dots \mathcal{F}_n$ be definite Horn expressions, each associated with a stakeholder $i \in \{1, \dots, n\}$. Let $\mathcal{B}$ be a set describing background knowledge. A formula $\mathcal{F}$ is a* **common ground** *for $\mathcal{F}_1, \dots \mathcal{F}_n$ and $\mathcal{B}$ if it satisfies each of the following postulates:*

(P1) $\mathcal{F}$ is coherent;

The first postulate is intuitive: the learned set of rules should be coherent with the background knowledge. If they were not so, the theory would recommend, for example, two mutually exclusive courses of action for the same situation. In the scenario defined above, if police is called for **illegalActivity** then parents cannot be alerted. These cannot occur together based on background knowledge.

(P2) if $\bigcup_{i=1}^{n} \mathcal{F}_i$ is coherent, then $\mathcal{F} \equiv \bigcup_{i=1}^{n} \mathcal{F}_1$;

(P2) ensures that if the union of rules provided by stakeholders is coherent then this should be the common ground.

(P3) for all $i \in \{1, \dots, n\}$ and all $\phi \in \mathcal{F}_i$, we have that $\mathcal{F} \not\models ant(\phi) \rightarrow p$ with $p \in \overline{con(\phi)}$;

The motivation for (P3) we find in [13]: "the essence of morality is to treat the interests of others as of equal weight with one's own", which we here interpret as a requirement that all agent's rules are considered equally informative and should not be entirely overridden. (P3) ensures that a rule that is in strict opposition to what an agent recommends is not in the common ground. For example, suppose we have two agents with recommendations:

$$\begin{array}{ll} (1) & \mathsf{illegalActivity} \rightarrow \mathsf{parentsAlert}, \\ (2) & \mathsf{illegalActivity} \rightarrow \mathsf{policeCall}, \end{array}$$

As these are in opposition, both cannot be parts of common ground. We have to neglect one or find some other way.

(P4) for each $\phi \in \mathcal{F}$, there is $\psi \in \bigcup_{i=1}^{n} \mathcal{F}_i$ with $\{\psi\} \models \phi$;

(P4) guarantees that a rule in a common ground can always be "traced back" to a rule from an agent. A random rule should not "sneak in" to the common ground, without being explicitly supported by an agent.

(P5) for all $i \in \{1, \dots, n\}$ and all $\phi \in \mathcal{F}$, there is (a non-trivial) $\psi \in \mathcal{F}$ such that $\{\phi\} \models \psi$

The fifth postulate ensures that some part of a stakeholder's rule is in a common ground, though, in a "weaker" form. In other words, a non-trivial part of each stakeholder's rules should be in common ground. Here, "weaker" refers to adding a variable to the antecedent of the rule. For example: $(\mathsf{illegalActivity} \wedge \mathsf{child}) \rightarrow \mathsf{parentsAlert}$ is a weaker form of $\mathsf{illegalActivity} \rightarrow \mathsf{parentsAlert}$.

(P6) for all $\phi \in \mathcal{F}$, if there is $p \in ant(\phi)$ such that, for all $q \in \overline{p}$, $\mathcal{F} \cup \{\phi^{q \backslash p}\}$ is coherent and there is $i \in \{1, \dots, n\}$ such that $\mathcal{F}_i \models \phi^{q \backslash p}$ then $\mathcal{F}_i \not\models \phi^{-p}$.

(P6) avoids unintended rules becoming part of the common ground. It means the rules offered by agents that are not relevant in the given scenario should be avoided while reaching a common ground. Below is an example to explain this case.

Example 2.3.7. Consider Horn expressions as

$$\begin{aligned}\mathcal{F}_1 &= \{\phi = \mathsf{illegalActivity} \rightarrow \mathsf{policeCall}, \\ &\quad\ \psi = \mathsf{lowBattery} \rightarrow \mathsf{charge}\} \\ \mathcal{F}_2 &= \{\varphi = (\mathsf{illegalActivity} \land \mathsf{child}) \rightarrow \mathsf{parentsAlert}\}.\end{aligned}$$

background knowledge is defined as before:

$$(\mathsf{parentsAlert} \land \mathsf{policeCall}) \rightarrow \bot$$

in order to resolve the incoherence in $\mathcal{F}_1 \cup \mathcal{F}_2$, one can replace ϕ with $(\mathsf{illegalActivity} \land \overline{\mathsf{child}}) \rightarrow \mathsf{policeCall}$. Without (P6), the rules

$$(\mathsf{illegalActivity} \land \mathsf{lowBattery}) \rightarrow \mathsf{policeCall},$$

$$(\mathsf{illegalActivity} \land \mathsf{charge}) \rightarrow \mathsf{policeCall}$$

could also be used to replace ϕ as they satisfy (P1)-(P5). Though, these rules are unintended since $\mathsf{lowBattery}$ and charge are unrelated with incoherence in $\mathcal{F}_1 \cup \mathcal{F}_2$.

2.3.6 Non-Redundant Horn Expression

In common usage, redundancy refers to the repetition of the same idea or item of information within a phrase, clause, or sentence. Redundancy refers to needless repetition. It occurs when two or more words or ideas of the same meaning use together. For example, 'adequate enough' [14]. The redundant data can be either a whole copy of the original data or select pieces of data.

In propositional logic, an argument is redundant if it contains two or more rules with the same meaning. Alternatively, if one or more rules can be eliminated from the expression with the lost if essential information. Non-redundancy is containing only what is needed for something to work.

Definition 4. *In Horn expression, we have a set of clauses where each clause refers to a specific rule. Mathematically, we say a Horn expression $\mathcal{F}$ is non-redundant if, for all $\phi \in \mathcal{F}$, it is not the case that*

$$\mathcal{F} \setminus \{\phi\} \models \phi \text{ [6]}$$

If a clause from Horn expression is removed then the remaining expression should not entail that clause. If in case, by removing a clause ϕ, the Horn expression still entails that clause, it means we have any other clause in the expression that is a redundancy of that clause ϕ. For a Horn expression to be non-redundant, all of its clauses should satisfy this condition.

Example 2.3.8. Suppose a Horn expression with rules defined as:

$$\mathsf{illegalActivity} \to \mathsf{parentsAlert},$$
$$(\mathsf{illegalActivity} \wedge \mathsf{child}) \to \mathsf{parentsAlert}.$$

This Horn expression is redundant because

$(\mathsf{illegalActivity} \to \mathsf{parentsAlert}) \models (\mathsf{illegalActivity} \wedge \mathsf{child}) \to \mathsf{parentsAlert})$

The second clause is an unnecessary repetition of the first clause.

2.3.7 Acyclic Horn Expression

Acyclicity is the state of being acyclic, containing no cycles.

Definition 5. *In Horn logic, a Horn expression is acyclic if there is no sequence of clauses $\phi_1, \ldots, \phi_n \in \mathcal{F}$ such that $con(\phi_i) \in ant(\phi_{i+1})$, for all $1 \leq i < n$, and $\phi_1 = \phi_n$.*

In simple words, if there is a sequence of clauses or rules, such that the concludent of one clause is contained in the antecedents of the other clause. This forms to have a chain from one clause to another. When the first clause is the same as the last clause. It becomes a closed chain, which refers to a cycle.

A Horn expression with such sequence is called cyclic, otherwise acyclic. This is explained with an example given below.

Example 2.3.9. Suppose a Horn Expression $\mathcal{F} = \bigcup_{i=1}^{7} \phi_i$, and clauses ϕ_i are defined as:

$$\begin{aligned}
&\phi_1 = \{p \to q\}, && \phi_2 = \{(p \wedge u) \to \overline{s}\},\\
&\phi_3 = \{(t \wedge u) \to p\}, && \phi_4 = \{q \to r\},\\
&\phi_5 = \{r \to p\}, && \phi_6 = \{s \to q\},\\
&\phi_7 = \{(t \wedge u) \to \overline{q}\}.
\end{aligned}$$

In the first step, we see, $\phi_1 \rightarrow \phi_4$ as $con(\phi_1) \in ant(\phi_4)$, so (ϕ_1, ϕ_4) is making a sequence together using definition of acyclic Horn expression. Next we see $\phi_4 \rightarrow \phi_5$ as $con(\phi_4) \in ant(\phi_5)$, so (ϕ_4, ϕ_5), combined into a sequence as (ϕ_1, ϕ_4, ϕ_5). Lastly, we see $con(\phi_5) \in ant(\phi_1)$ combined into a sequence as $(\phi_1, \phi_4, \phi_5, \phi_1)$

This is a closed sequence, making a cycle. First clause in this sequence is same as of first clause.Therefore, $\mathcal{F}$ is a **Cyclic Horn Expression**. Horn expression can also be treated as a directed graph in order to trace cycles. Where $(ant(\phi_i), con(\phi_i))$ are vertices of the graph, and we draw edges based on the condition if $con(\phi_i) \in ant(\phi_{i+1})$. Then we check for the cycles in a directed graph.

2.3.8 Theorems

We have defined notions of conflict, non-redundant and acyclic Horn expression, Now we relate three important theorems that are used as essential conditions, for a common ground to exist.

Theorem 2. *[6] There are non-redundant, not in-conflict, but cyclic Horn expressions for which no common ground exists.*

Theorem 3. *[6] There are acyclic, non-redundant, but in-conflict Horn expressions for which no common ground exists.*

Theorem 4. *[6] There are acyclic, not in conflict, but redundant Horn expressions for which no common ground exists.*

From these theorems, we conclude that it is necessary for a Horn expression to be acyclic, non-redundant, and not in conflict, for a common ground to exist. If any of these three conditions are removed then a common ground may not exist. From now on, we consider a Horn expression non-redundant, acyclic and having no conflict. We discuss incoherence and how can it be resolved to find a common ground.

2.3.9 Dependency Graph

A dependency graph is a data structure formed by a directed graph. It describes the dependency of an entity on other entities within the same system. The underlying structure of a dependency graph is a directed, where each node points to a dependent node [15].

For example, a graph with nodes A, B, C, D and edges $(A, B), (A, C), (B, D)$, which looks like:

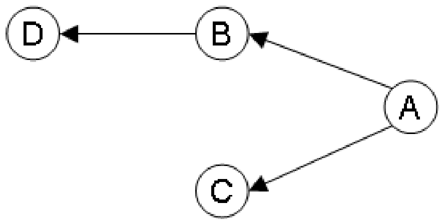


Figure 2.4: Dependency Graph

From Figure 2.4, we can derive the relation: A depends on B and C, while B depends on D. We can alternatively say that A is the parent node of B and C, while B is the parent node of D, and A has no parent in this graph.

Similarly, in Horn logic, we use dependency graphs, where nodes are clauses and edges are dependencies of clauses towards each other.

Formally, we define, the dependency graph as:

Definition 6. *Let ϕ and ψ be definite Horn clauses and let $\mathcal{F}$ be a definite Horn expression The* ***dependency Graph*** *of $\mathcal{F}$ is the directed graph (V, E), where*

- *V is the set of all pairs (ψ, ϕ) such that $\psi \Rightarrow_{\mathcal{F}} \phi$ and $con(\psi) \in \overline{con(\phi)}$ and,*
- *E is the set of all $((\psi', \phi'), (\psi, \phi))$ such that $\phi' \neq \phi$ and ϕ' occurs in a derivation of ϕ w.r.t. ψ and $\mathcal{F}$.*

We say that $v' \in V$ is a parent for $v \in V$ if $(v', v) \in E$.

We say that the pair (ψ, ϕ) is **safe** for $\mathcal{F}$, if (ψ, ϕ) has no parent in the dependency graph of $\mathcal{F}$.

This is how, we draw dependency graphs and search for safe pairs of clauses to resolve incoherence in a Horn expression. For this, we can relate a lemma from Ozaki et al. [6].

Lemma 1. *Let $\mathcal{F}$ be an acyclic Horn expression. If $\mathcal{F}$ is incoherent, then there are $\psi, \phi \in \mathcal{F}$ such that (ψ, ϕ) is* ***safe****.*

From this it is concluded that if Horn expression $\mathcal{F}$ is incoherent then at least one safe pair is guaranteed to exist.

Weaker Version of a Clause

We also introduce the notion of **"Weaker Version of a Clause"** before defining the algorithm. Suppose there is a clause ϕ, we modify it to ϕ', by adding atoms to the antecedent of a clause ϕ. The resulting clause ϕ' is such that $\{\phi\} \models \phi'$ but $\{\phi'\} \not\models \phi$. It may refer to ϕ' as the result of 'weakening' ϕ, or simply say that ϕ' is a 'weaker' version of ϕ

For example, $\phi' : (p \wedge q \wedge s) \rightarrow r$ is weaker version of $\phi : (p \wedge q) \rightarrow r$

2.4 The Original Common Ground Algorithm

We have defined and explained all the essential notions and related theorem, and now we can define the original version of the common ground Algorithm from Ozaki et al. [6].

Algorithm 1 Building coherent $\mathcal{F}$

Input: Horn expression sets $\mathcal{F}_1, \ldots \mathcal{F}_n$ and $\mathcal{B}$.

Output: A common ground for $\mathcal{F}_1, \ldots \mathcal{F}_n$ and $\mathcal{B}$ or $\emptyset$

```
1: $\mathcal{F} := \mathcal{F}_1 \cup \ldots \cup \mathcal{F}_n$
2: if $\mathcal{F}$ is cyclic or redundant or in conflict then
3:     return $\emptyset$ (A common ground may not exist).
4: end if
5: while $\mathcal{F}$ is incoherent do
6:     Find $\psi, \phi \in \mathcal{F}$ such that $(\psi, \phi)$ is safe.
7:     Replace $\phi$ by all $\phi' \in \{\phi^{+p} \mid p \in \bar{l}, l \in ant(\psi) \setminus ant(\phi)\}$ coherent with $\mathcal{F} \setminus \{\phi\}$
8: end while
9: return $\mathcal{F}$
```

2.4.1 Description

We now explain the algorithm in simple words using steps.

- It takes a finite number of definite Horn expressions $\mathcal{F}_1, \ldots \mathcal{F}_n$ and background knowledge $\mathcal{B}$ as input.

- It verifies that $\mathcal{F} = \bigcup_{i=1}^{n} \mathcal{F}_i$ is not in conflict, not redundant, and not acyclic. If any of the condition is satisfied, then a common ground may not exist
- At each iteration of the "while" loop (Line 5), Algorithm 1 first selects clauses $\psi, \phi \in \mathcal{F}$ such that (ψ, ϕ) is safe (Line 6), using dependency graphs, and By **Lemma 1**, we know that at least one safe pair is guaranteed to exist.
- When it finds the safe pair then in Line 7, it resolves incoherence by replacing ϕ with all weaker versions of this clause that are coherent with the Horn expression being constructed.
- After exiting the while loop it returns a coherent expression. This is what we call a common ground for this Horn expression $\mathcal{F}$.

2.4.2 Examples

Example 2.4.1. Consider $\mathcal{F}_1 - \mathcal{F}_7$ as $\mathcal{F} = \bigcup_{i=1}^{7} \mathcal{F}_i$ is not in conflict, not redundant but cyclic and incoherent. We draw a dependency graph using the definition.

$$\mathcal{F}_1 = \{\underbrace{p \to q}_{\phi_1}\},$$
$$\mathcal{F}_2 = \{\underbrace{(r \wedge \overline{s}) \to \overline{q}}_{\phi_2}\},$$
$$\mathcal{F}_3 = \{\underbrace{r \to p}_{\phi_3}\},$$
$$\mathcal{F}_4 = \{\underbrace{(q \wedge \overline{t}) \to \overline{p}}_{\phi_4}\},$$
$$\mathcal{F}_5 = \{\underbrace{q \to r}_{\phi_5}\},$$
$$\mathcal{F}_6 = \{\underbrace{(p \wedge u) \to \overline{r}}_{\phi_6}\}.$$

$$\phi_2 \Rightarrow_{\mathcal{F}} \phi_1 : (r \wedge \overline{s}) \to \overline{q}, r \to p, p \to q,$$
$$\phi_4 \Rightarrow_{\mathcal{F}} \phi_3 : (q \wedge \overline{t}) \to \overline{p}, q \to r, r \to p$$
$$\phi_6 \Rightarrow_{\mathcal{F}} \phi_5 : (p \wedge u) \to \overline{r}, p \to q, q \to r$$

$$\begin{tikzcd}
\phi_2, \phi_1 \arrow[dr] & & \phi_4, \phi_3 \arrow[ll] \\
& \phi_6, \phi_5 \arrow[ur] &
\end{tikzcd}$$

Figure 2.5: A dependency graph with no safe clause

This illustrates there is no pair of safe clauses, so there is no common ground for the Horn expressions. According to Lemma 1, if the theory is incoherent and the Horn expression is acyclic (which also means avoiding cycles in the dependency graph) then there is a safe pair of clauses in it. It is clear from example that if a Horn expression is either in conflict, cyclic or redundant, then there is no need to search for safe clauses for resolving incoherence. Reason is the nonexistence of common ground according to algorithm.

Example 2.4.2. Consider $\mathcal{F}_1 - \mathcal{F}_7$ as in We have that $\mathcal{F} = \bigcup_{i=1}^{7} \mathcal{F}_i$ is not in conflict, not redundant, acyclic, but incoherent.

$$\begin{aligned} &\mathcal{F}_1 = \{p \to s\}, && \mathcal{F}_2 = \{(p \wedge u) \to \overline{s}\}, \\ &\mathcal{F}_3 = \{(t \wedge q) \to \overline{s}\}, && \mathcal{F}_4 = \{t \to p\}, \\ &\mathcal{F}_5 = \{(t \wedge \overline{u}) \to \overline{p}\}, && \mathcal{F}_6 = \{s \to q\}, \\ &\mathcal{F}_7 = \{(p \wedge t) \to \overline{q}\}. \end{aligned}$$

We draw a dependency graph using the definition.

$$\begin{aligned} &\mathcal{F}_1 = \{\underbrace{p \to s}_{\phi_1}\}, \\ &\mathcal{F}_2 = \{\underbrace{(p \wedge u) \to \overline{s}}_{\phi_2}\}, \\ &\mathcal{F}_3 = \{\underbrace{(t \wedge \overline{q}) \to \overline{s}}_{\phi_3}\}, \\ &\mathcal{F}_4 = \{\underbrace{t \to p}_{\phi_4}\}, \\ &\mathcal{F}_5 = \{\underbrace{(t \wedge \overline{u}) \to \overline{p}}_{\phi_5}\}, \\ &\mathcal{F}_6 = \{\underbrace{s \to q}_{\phi_6}\}, \\ &\mathcal{F}_7 = \{\underbrace{(p \wedge t) \to \overline{q}}_{\phi_7}\}, \end{aligned}$$

$$\begin{aligned} &\phi_2 \Rightarrow_{\mathcal{F}} \phi_1 : (p \wedge u) \to \overline{s}, p \to s \\ &\phi_3 \Rightarrow_{\mathcal{F}} \phi_1 : (p \wedge u) \to \overline{s}, t \to p, p \to s \\ &\phi_5 \Rightarrow_{\mathcal{F}} \phi_4 : (t \wedge \overline{u}) \to \overline{p}, t \to p, \\ &\phi_7 \Rightarrow_{\mathcal{F}} \phi_6 : (p \wedge t) \to \overline{q}, p \to s, s \to q, \end{aligned}$$

$$\begin{tikzcd} \phi_2,\phi_1 \arrow[r] & \phi_7,\phi_6 \\ \phi_5,\phi_4 \arrow[r] & \phi_3,\phi_1 \arrow[u] \end{tikzcd}$$

Figure 2.6: A dependency graph with safe clauses

The nodes (ϕ_2, ϕ_1) and (ϕ_5, ϕ_4) have no parent. Algorithm 1 iterates twice, each time selecting one of these pairs (the order does not change the result). It returns a common ground $\mathcal{F}^*$ for $\mathcal{F}_1, \ldots, \mathcal{F}_7$, which is union of:

$$\begin{aligned} &\mathcal{F}_1^* = \{(p \wedge \overline{u}) \to s\}, && \mathcal{F}_2 = \{(p \wedge u) \to \overline{s}\}, \\ &\mathcal{F}_3 = \{(t \wedge \overline{q}) \to \overline{s}\}, && \mathcal{F}_4^* = \{(t \wedge u) \to p\}, \\ &\mathcal{F}_5 = \{(t \wedge \overline{u}) \to \overline{p}\}, && \mathcal{F}_6 = \{s \to q\}, \\ &\mathcal{F}_7 = \{(t \wedge u) \to \overline{q}\}. \end{aligned}$$

Example 2.4.3. It is important to ensure if the input of Algorithm 1 is non-redundant then the output is also non-redundant. We illustrate this with the following example. Consider

$$\begin{aligned}\mathcal{F}_1 &= \{(q \wedge r) \to s,\ (p \wedge q \wedge \overline{r}) \to \overline{s}\},\\ \mathcal{F}_2 &= \{(p \wedge q) \to s\}.\end{aligned}$$

In this case Algorithm 1 would resolve the incoherence by replacing the clause in $\mathcal{F}_2$ by $(p \wedge q \wedge r) \to s$. The latter rule is redundant because it is implied by $\mathcal{F}_1$.

Chapter 3

Finding Common Grounds

In this chapter, we are going to discuss changes in the original version of common ground algorithm (that has been explained in the previous chapter) by reducing some limitations. In the original algorithm 1, we could only find a common ground for a horn expression that are not in conflict, acyclic, and non-redundant according to theorems 2, 3 and 4. These are the basic limitations of the algorithm. Now changes are going to be introduced in the algorithm to address cyclic and redundant horn expressions. Additionally, if a conflict exists, algorithm can resolve the conflict with specified conditions. We discuss and explain each feature separately, and then define the new version of the algorithm.

3.1 Cyclic and Redundant Expressions

As per definition, a horn expression is cyclic if any collections of clauses in the horn expression is forming a cycle. Similarly, if there is a repetition in clauses then the horn expression is redundant. According to stated theorems 2 and 4, a common ground is not guaranteed to exist if the horn expression is either cyclic or redundant.

Cycles are detected and removed from a horn expression to improve the algorithm. For removing cycles, all main clauses are deleted which cause the cycle. The remaining clauses would make an acyclic horn expression which is good for finding a common ground. At this point, it is unclear which specific clause needs to be removed to make the expression acyclic. In general, removal of any clause from the cycle would make the expression acyclic. In this case, we might lose some important information from the agents that is crucial for horn expression. This is just a proposed way of making expression acyclic. To achieve maximum efficiency, it would be good idea to set preferences on removal of clauses for removing cycles. In practice, it is very rare to find a cycle in the Horn expression. In this thesis, case study of Moral machine experiment has been conducted where we do not find any cycle in the horn expression (from the dataset of moral machine experiment). Similarly, we remove the redundant clauses from the horn expression if redundancy exists. Examples are given below to provide explanation.

Example 3.1.1. Suppose a Horn expression $\mathcal{F} = \bigcup_{i=1}^{5} \phi_i$, and clauses ϕ_i are defined as:

$$\begin{aligned} &\phi_1 = \{p \to q\}, \quad \phi_2 = \{(p \wedge u) \to \overline{s}\}, \\ &\phi_3 = \{q \to r\}, \quad \phi_4 = \{r \to p\} \\ &\phi_5 = \{s \to q\}. \end{aligned}$$

there is a sequence of classes as $(\phi_1, \phi_3, \phi_4, \phi_1)$, which is making a cycle.

$$p \to q \ (\phi_1), \ q \to r \ (\phi_3), \ r \to p \ (\phi_4).$$

In this case, algorithm removes the last clause ϕ_4 of the sequence which is completing a cycle. The remaining horn expression is acyclic. Horn clauses are usually not ordered, so in practice one can use a preference ordering to select the clause to be removed.

Example 3.1.2. Suppose a Horn expression $\mathcal{F} = \bigcup_{i=1}^{5} \phi_i$, and clauses ϕ_i are defined as:

$$\begin{aligned} &\phi_1 = \{p \to q\}, && \phi_2 = \{(p \wedge u) \to \overline{s}\}, \\ &\phi_3 = \{(p \wedge s) \to q\}, && \phi_4 = \{(r \wedge s) \to p\}, \\ &\phi_5 = \{s \to q\}, && \phi_6 = \{(r \wedge s \wedge t) \to p\} \end{aligned}$$

we can see that ϕ_3 is implied by ϕ_1 and ϕ_6 is implied by ϕ_4.

$$p \to q \ (\phi_1) \models (p \wedge s) \to q \ (\phi_3)$$

$$(r \wedge s) \to p \ (\phi_4) \models (r \wedge s \wedge t) \to p \ (\phi_6)$$

In this case, algorithm removes all the clauses that are redundant. These are ϕ_3 and ϕ_6 in the given horn expression. The remaining horn expression is non-redundant.

3.2 Resolving Conflicts

A Horn expression is in conflict if there is a pair of clauses in it that is in conflict. A common ground is not guaranteed to exist if there is a conflict in the Horn expression. So, resolution of conflicts is a way to deal with this limitation. To resolve conflict, we propose a solution containing some specific conditions. A collection of symbols is assumed which is occurring with clauses in the horn expression. It is denoted by $\mathcal{C}$ and each clause in the horn expression $\mathcal{F}$ has an associated symbol in $\mathcal{C}$. Additionally, a second background knowledge is assumed other than $\mathcal{B}$, that is denoted by $\overline{\mathcal{B}}$. It consists of a set of definite horn clauses built from symbols occurring in $\mathcal{C}$. As background knowledge contains basic constraints about the world. $\overline{\mathcal{B}}$ actually consists of the clauses, imply from a specific symbol to generalized symbol.

For example, there is a symbol p in $\mathcal{C}$ and a clause ϕ:- $p \implies q$ in $\overline{\mathcal{B}}$. ϕ can be interpreted as q is the generalized form of p. As a real-life example:

$$\mathsf{child} \implies \mathsf{humanBeing}$$

$$\mathsf{norway} \implies \mathsf{scandinavianCountry}$$

Resolution of conflict is based on the idea of finding a pair of clauses that are in conflict. According to definition 2, a weaker form of a clause is generated using generalized symbols from the background knowledge $\overline{\mathcal{B}}$. If there is still conflict in the horn expression after using generalized symbols, then generalized symbols are replaced with specific form of symbols from $\mathcal{C}$. The algorithm keeps checking the expression until the horn expression has no conflicts. New version of the algorithm is introduced next, along with examples to explain it.

3.3 Improved version of Algorithm

Algorithm 2 Building coherent $\mathcal{F}$

Input: Horn expression sets $\mathcal{F}_1, \ldots \mathcal{F}_n$, $\mathcal{B}$, $\overline{\mathcal{B}}$ and $\mathcal{C}$.

Output: A common ground for $\mathcal{F}_1, \ldots \mathcal{F}_n$, $\mathcal{B}$, $\overline{\mathcal{B}}$ and $\mathcal{C}$ if $\mathcal{F}_1, \ldots \mathcal{F}_n$ are acyclic, non-redundant and not in conflict. Otherwise a common ground after removing cycles, redundancies and resolving conflict from $\mathcal{F}_1, \ldots \mathcal{F}_n$

```
1:  $\mathcal{F} := \mathcal{F}_1 \cup \ldots \cup \mathcal{F}_n$
2:  while $\mathcal{F}$ is cyclic do
3:      Find cycle $\omega := \{\phi_1, \ldots, \phi_n\} \in \mathcal{F}$
4:      remove $\phi_n$ (see description of algorithm) from $\mathcal{F}$
5:  end while
6:  while $\mathcal{F}$ is redundant do
7:      Find $\phi_i \in \mathcal{F}$ such that $\mathcal{F} \setminus \{\phi_i\} \models \phi_i$
8:      remove $\phi_i$ from $\mathcal{F}$
9:  end while
10: $\mathcal{F}^* := \mathcal{F}$
11: while there is $\phi_i, \phi_j$ in $\mathcal{F}$ such that $(\phi_i, \phi_j)$ is in conflict do
12:     if $\phi_j$ in $\mathcal{F}^*$ then
13:         Replace $\phi_j$ in $\mathcal{F}$ by $\phi_j^{+s}$ where $c_j \rightarrow s \in \overline{\mathcal{B}}$
14:     else
15:         Replace $(\phi_i, \phi_j)$ in $\mathcal{F}$ by $(\phi_i^{c_i/s}, \phi_j^{c_j/s})$ 2.3.2 where $c_i, c_j \in \mathcal{C}$
16:     end if
17: end while
18: while $\mathcal{F}$ is incoherent do
19:     Find $\psi, \phi \in \mathcal{F}$ such that $(\psi, \phi)$ is safe.
20:     Replace $\phi$ by all $\phi' \in \{\phi^{+p} \mid p \in \bar{l}, l \in ant(\psi) \setminus ant(\phi)\}$ coherent with $\mathcal{F} \setminus \{\phi\}$
21: end while
22: return $\mathcal{F}$
```

3.4 Description of Algorithm

Above mentioned algorithm is described in the following steps.

- Algorithm takes a finite number of definite Horn expressions $\mathcal{F}_1, \ldots \mathcal{F}_n$, background knowledge $\mathcal{B}$, second background knowledge $\overline{\mathcal{B}}$ and collection of symbols $\mathcal{C}$ as input.
  - ✳ Here in $\mathcal{F}_1, \ldots \mathcal{F}_n$, there are definite horn clauses as rules offered by n agents.
  - ✳ Background knowledge $\mathcal{B}$ consists of horn constraints, built from symbols occurring in $\mathcal{F}_1, \ldots \mathcal{F}_n$. Each clause is formed by an atom and its excludent. For example, a person cannot be both, a child or an adult, will occur in $\mathcal{B}$ as $(\mathsf{child} \ \wedge \ \mathsf{adult}) \rightarrow \bot$.
  - ✳ $\mathcal{C}$ consists of a finite collection of symbols. Each symbol is associated with a clause in horn expression. Symbols in $\mathcal{C}$ might not be unique which means multiple clauses can have similar symbols.
  - ✳ Background knowledge $\overline{\mathcal{B}}$ consists of a finite number of definite horn clauses built from symbols in $\mathcal{C}$. Each clause has only one antecedent and one concludent. Antecedent is a symbol from $\mathcal{C}$ and the concludent is a generalized symbol of the antecedent. For example a chair is furniture and a mobile phone is an electronic device will occur in $\overline{\mathcal{B}}$ as *chair* $\rightarrow$ *furniture* and *mobilePhone* $\rightarrow$ *electronicDevice*.
- Algorithm 2 gives output as a common ground for $\mathcal{F}_1, \ldots \mathcal{F}_n$, $\mathcal{B}$, $\overline{\mathcal{B}}$ and $\mathcal{C}$ if $\mathcal{F}_1, \ldots \mathcal{F}_n$ are acyclic, non-redundant and not in conflict. Otherwise, it returns a common ground after removing cycles, redundancies and resolving conflict from $\mathcal{F}$, if any of these conditions exists.
- From line 2 to 5, algorithm checks for cycles in $\mathcal{F}$ using definition 5. If a cycle exists, then there exists a sequence of clauses making a cycle. We assume the sequence is ordered and last edge is completing the cycle which can refer to as the back edge. We remove that last clause of sequence from $\mathcal{F}$. It uses while loop for cycles and removes all possible cycles.
- From lines 6 to 9, algorithm uses while loop to check for redundancy in $\mathcal{F}$ using definition 4 and removes redundant clauses.
- From line 10 to 17, algorithm deals with conflict. It follows the following steps.

    - In line 10, a copy of $\mathcal{F}$ is generated to keep track of updated clauses.
    - In line 11, while loop checks for conflict in all possible pair combinations of clauses in $\mathcal{F}$.
    - When it finds a conflicting pair then it creates the weaker version of the second clause of the pair on line 13. It also checks if the weaker version of the second clause of the pair has been generated before or not. It searches the corresponding symbol c of the clause (that is going to be updated) from $\mathcal{C}$ and clause φ from background knowledge $\overline{\mathcal{B}}$ that has c as antecedent. We weaken the second clause by adding concludent of φ to its antecedent. In background knowledge $\overline{\mathcal{B}}$, clauses are as implications from a specific symbol to a generalized symbol. Therefore, conflict is resolved by generalized symbols.
    - In case, a new conflict is emerged after resolving conflict by generalized symbol. It updates both clauses of the pair using specific symbols on line 15. It keeps checking until there is no conflicting pair left.
- From line 18 to 21, algorithm checks for incoherence and resolve it. For this, at each iteration of the while loop, it first selects clauses $\psi, \phi \in \mathcal{F}$ such that (ψ, ϕ) is safe in line 19. After finding a safe pair, it resolves incoherence by replacing (second clause of pair) ϕ with all weaker versions of this clause that are coherent with the Horn expression in line 20.
- At the end, algorithm returns $\mathcal{F}$ that is a common ground for $\mathcal{F}_1 \cup \ldots \cup \mathcal{F}_n$.

3.5 Examples

Examples 3.1.1 and 3.1.2, showed how Algorithm 2 works to resolve cyclic and redundant horn expressions respectively. Here is an example of Algorithm 2 dealing with conflict.

Example 3.5.1. Consider a Horn expression $\mathcal{F} = \bigcup_{i=1}^{2} \phi_i$, and clauses ϕ_i are defined as:

$$\phi_1 := \mathsf{illegalActivity} \rightarrow \mathsf{parentsAlert},$$
$$\phi_2 := \mathsf{illegalActivity} \rightarrow \mathsf{policeCall}$$

collection of symbols $\mathcal{C}$ is defined as

$$\mathcal{C} := \{\mathsf{child},\ \mathsf{adult}\}$$

both background knowledge $\mathcal{B}$ and $\overline{\mathcal{B}}$ are defined as:

$$\begin{aligned}\mathcal{B} := \{&(\mathsf{parentsAlert} \wedge \mathsf{policeCall}) \rightarrow \perp,\\ &(\mathsf{child} \wedge \mathsf{adult}) \rightarrow \perp,\\ &(\mathsf{under18} \wedge \mathsf{above18}) \rightarrow \perp\}\end{aligned}$$

$$\overline{\mathcal{B}} := \{\mathsf{child} \rightarrow \mathsf{under18},\ \mathsf{adult} \rightarrow \mathsf{above18}\}$$

In $\mathcal{C}$ there are symbols for each clause. It indicates that in ϕ_1, $\mathsf{illegalActivity}$ is done by the child and in ϕ_2, $\mathsf{illegalActivity}$ is done by the adult. $\overline{\mathcal{B}}$ shows that $\mathsf{under18}$ and $\mathsf{above18}$ are the generalized symbols for child and adult respectively.

Here $\mathcal{F}$ is acyclic, non-redundant but in conflict. We have a pair (ϕ_1, ϕ_2) that is in conflict according to background knowledge $\mathcal{B}$. Now algorithm 2 resolves conflict by generating a weaker version of ϕ_2 using generalized symbols. corresponding symbol of ϕ_2 in $\mathcal{C}$ is adult' so it searches for the clause in $\overline{\mathcal{B}}$ for generalized symbol that is $(\mathsf{adult} \rightarrow \mathsf{above18})$. So $\mathsf{above18}$ is the generalized symbol here and clause is weakened it.

$$\phi_2 : (\mathsf{illegalActivity} \wedge \mathsf{above18}) \rightarrow \mathsf{policeCall}$$

Updated horn expression $\mathcal{F}$ appears as:

$$\begin{aligned}\phi_1 &:= \mathsf{illegalActivity} \rightarrow \mathsf{parentsAlert},\\ \phi_2' &:= (\mathsf{illegalActivity} \wedge \mathsf{above18}) \rightarrow \mathsf{policeCall}\end{aligned}$$

This is not in conflict. It still has incoherence that needs to be resolved to find a common ground. Algorithm 2 searches for the safe pair (ϕ_2, ϕ_1) and resolve incoherence by weakening the clause using background knowledge $\mathcal{B}$ and expression is updated as:

$$\begin{aligned}\phi_1' &:= (\mathsf{illegalActivity} \wedge \mathsf{under18}) \rightarrow \mathsf{parentsAlert}\\ \phi_2' &:= (\mathsf{illegalActivity} \wedge \mathsf{above18}) \rightarrow \mathsf{policeCall}\end{aligned}$$

This is a common ground for $\mathcal{F}$. It concludes that both agents agreed on the conditions that if '$\mathsf{illegalActivity}$' is done by a person who is '$\mathsf{above18}$' then call the police. If the person is '$\mathsf{under18}$' then alert parents. In this example, algorithm resolved conflict using generalized symbols which did not cause any further conflict. Now consider another example where algorithm needs to resolve conflict using specified symbols.

Example 3.5.2. Consider an illegal activity that is done by the child, and is detected by a police robot. There are three different agents with different recommendations. Agents' recommendations are based on the location, in which the illegal activity by the child is detected. These recommendations are defined as a Horn expression $\mathcal{F} = \bigcup_{i=1}^{3} \phi_i$ and clauses ϕ_i are defined as:

$$\begin{aligned}\phi_1 &:= \mathsf{illegalActivity} \rightarrow \mathsf{parentsAlert},\\ \phi_2 &:= \mathsf{illegalActivity} \rightarrow \mathsf{policeCall},\\ \phi_3 &:= \mathsf{illegalActivity} \rightarrow \mathsf{teacherNotify}\end{aligned}$$

collection of symbols $\mathcal{C}$ is defined as

$$\mathcal{C} := \{\mathsf{houseRegion},\ \mathsf{shoppingMall},\ \mathsf{school}\}$$

both background knowledge $\overline{\mathcal{B}}$ and $\mathcal{B}$ are defined as:

$$\begin{aligned}\overline{\mathcal{B}} := \{&\mathsf{houseRegion} \rightarrow \mathsf{privatePlace},\\ &\mathsf{shoppingMall} \rightarrow \mathsf{publicPlace}\}\\ &\mathsf{school} \rightarrow \mathsf{publicPlace}\}\end{aligned}$$

$$\begin{aligned}\mathcal{B} := \{&(\mathsf{parentsAlert} \wedge \mathsf{policeCall}) \rightarrow \perp,\\ &(\mathsf{parentsAlert} \wedge \mathsf{teacherNotify}) \rightarrow \perp,\\ &(\mathsf{policeCall} \wedge \mathsf{teacherNotify}) \rightarrow \perp,\\ &(\mathsf{school} \wedge \mathsf{houseRegion}) \rightarrow \perp,\\ &(\mathsf{school} \wedge \mathsf{shoppingMall}) \rightarrow \perp,\\ &(\mathsf{houseRegion} \wedge \mathsf{shoppingMall}) \rightarrow \perp\}\end{aligned}$$

$\mathcal{C}$ indicates that in ϕ_1, '$\mathsf{illegalActivity}$' done by the child is detected with in the premises of a house. In ϕ_2, '$\mathsf{illegalActivity}$' done by the child is detected in the shopping mall and in ϕ_3, '$\mathsf{illegalActivity}$' done by the child is detected in the school. $\overline{\mathcal{B}}$ shows '$\mathsf{publicPlace}$' is generalized symbol for school and shopping mall while '$\mathsf{privatePlace}$' is the generalized symbol for house area.

Here $\mathcal{F}$ is acyclic, non-redundant but in conflict. There are two pairs (ϕ_1, ϕ_2) and (ϕ_1, ϕ_3) that are in conflict according to background knowledge $\mathcal{B}$. It resolves the conflict by weakening the second clauses of both pairs (that are ϕ_2 and ϕ_3) using generalized symbols from $\overline{\mathcal{B}}$. Both ϕ_2 and ϕ_3 have same generalized symbols. The resulting $\mathcal{F}$ is:

$$\begin{aligned}
\phi_1 &:= \mathsf{illegalActivity} \rightarrow \mathsf{parentsAlert},\\
\phi_2' &:= (\mathsf{illegalActivity} \wedge \mathsf{publicPlace}) \rightarrow \mathsf{policeCall},\\
\phi_3' &:= (\mathsf{illegalActivity} \wedge \mathsf{publicPlace}) \rightarrow \mathsf{teacherNotify}
\end{aligned}$$

Algorithm 2 again searches for conflicting pairs and finds a pair that is (ϕ_2, ϕ_3),. These both clauses have been updated before by generalized symbols. Therefore, algorithm uses specific symbols now. For this, it weakens both clauses by replacing generalized symbols with specific symbols. The resulting horn expression is:

$$\begin{aligned}
\phi_1 &:= \mathsf{illegalActivity} \rightarrow \mathsf{parentsAlert},\\
\phi_2'' &:= (\mathsf{illegalActivity} \wedge \mathsf{shoppingMall}) \rightarrow \mathsf{policeCall},\\
\phi_3'' &:= (\mathsf{illegalActivity} \wedge \mathsf{school}) \rightarrow \mathsf{teacherNotify}
\end{aligned}$$

Now $\mathcal{F}$ is not in conflict, but still have incoherence. For resolving incoherence, it finds two safe pairs $(\phi_2, \phi_1))$ and (ϕ_3, ϕ_1). It generates the weaker version of ϕ_1 using any of the safe pairs and updates $\mathcal{F}$ as:

$$\begin{aligned}
\phi_1' &:= (\mathsf{illegalActivity} \wedge \mathsf{houseRegion}) \rightarrow \mathsf{parentsAlert},\\
\phi_2'' &:= (\mathsf{illegalActivity} \wedge \mathsf{shoppingMall}) \rightarrow \mathsf{policeCall},\\
\phi_3'' &:= (\mathsf{illegalActivity} \wedge \mathsf{school}) \rightarrow \mathsf{teacherNotify}
\end{aligned}$$

which is a common ground for $\mathcal{F}$. It concludes that all three agents agreed on the conditions that if '$\mathsf{illegalActivity}$' done by the child with in the premises of the house, parents should be alerted. If it is conducted within the area of a shopping mall, then police should be called and if it is in the school area then teacher should be notified.

Chapter 4

Implementation

One of the main goals of this thesis is to write an implementation of the updated common ground algorithm 2. Selection of programming language is an important consideration for writing an efficient implementation of the algorithm. *Python*, a powerful programming language has been chosen. It has several benefits due to availability of loads of libraries and packages. For example, a library called *Sympy* is used for symbolic mathematics, *Networkx* to study large complex networks as graphs and so on.

In the following chapter, an overview of the design adopted for the implementation is presented. It also gives a brief description of the most essential auxiliary structures and functions. To implement the algorithm, a design has been adopted where each feature is implemented step by step in the form of functions. Mathematical definitions are used in implementing the features. All these definitions have been discussed in previous chapters. **My implementation code is available on GitHub.**

4.1 Coherence

Definition of coherence is recalled, to implement coherence:

Let $\mathcal{F}$ be a definite Horn expression. A definite Horn clause ϕ is **coherent** with a Horn expression $\mathcal{F}$ if $\mathcal{F} \setminus \{\phi\} \not\models \phi$ and

- there is no $\psi \in \mathcal{F}$ such that $\psi \Rightarrow_{\mathcal{F}} \phi$ or $\phi \Rightarrow_{\mathcal{F}} \psi$ while $con(\psi) \in \overline{con(\phi)}$ (note that $con(\psi) \in \overline{con(\phi)}$ implies $con(\phi) \in \overline{con(\psi)}$).

The set $\mathcal{F}$ is **coherent** if all $\phi \in \mathcal{F}$ are coherent with $\mathcal{F}$ (and *incoherent* otherwise).

Definition is divided into two conditions, *Entailment* and *Derivation*:

- **Entailment**

  Implementation follows the following steps for entailment.

  ✳ A function is defined which takes the Horn expression as input. It checks for each clause. If that clause is removed from the expression, then the remaining expression should not entail that clause. Function *entails* is imported directly from the library *Sympy* [16] to check entailment, using the satisfiability condition by theorem 1.

  ✳ If the condition is true for all the clauses then the given Horn expression is satisfying the entailment condition.

```python
from sympy.logic.inference import entails

def check_Hornexpr_entailment(F):
    count = 0
    for clause in F:
        F_without_clause = [x for x in F if x is not clause]
        if not entails(clause, F_without_clause):
            count +=1
    if count == len(F):
        return True
    else:
        return False
```

Listing 4.1: Structure for *Entailment*

- **Derivation**

  The following part is implemented for derivation:

  $$\boxed{\text{there is no } \psi \in \mathcal{F} \text{ such that } \psi \Rightarrow_{\mathcal{F}} \phi \text{ or } \phi \Rightarrow_{\mathcal{F}} \psi \text{ while } con(\psi) \in \overline{con(\phi)}}$$

  The following condition is recalled to check deviation:

  $$\boxed{\psi \Rightarrow_{\mathcal{F}} \phi \text{ Iff } \mathcal{F} \cup ant(\psi) \models ant(\phi)}$$

  Right-hand side of this condition is used, to check derivation. Implementation follows the following steps.

- ✳ A function is defined which takes a clause as input and returns antecedents and concludents of the clause. The complete code can be seen here in Appendix A.1. A second function is defined which takes atom as input and get all its excludents, using given background knowledge (Appendix A.3). To get all atoms in background knowledge, a function *extract_symbols* extracts all distinct atoms from any expression (Appendix A.2).
- ✳ A third function is defined which takes two clauses as input. It checks if the derivative of one clause with respect to other exists or not (Appendix A.4). All possible pair combinations of the clauses in $\mathcal{F}$ are checked, to verify the derivation condition. If condition is true for all pairs, then the derivation condition is satisfied for coherence.

```python
def check_Hornexpression_derivation(F):

    count = 0
    combinations = generate_combinations(F)
    for comb in combinations:
        if not clauses_derivation(comb[0], comb[1], F) and not
        clauses_derivation(comb[1], comb[0], F) or comb[0] == comb[1] :
            count += 1
    if count == len(combinations):
        return True
    else:
        return False
```

Listing 4.2: Structure for *Derivation*

- Both entailment and derivation functions are combined to check the coherence of the Horn expression.

```python
def HornExpression_coherence(F, background_know):

    if check_Hornexpr_entailment(F) and check_Hornexpression_derivation(F):
        return True
    else:
        return False
```

Listing 4.3: Structure for *Coherence*

4.2 Cyclic Behavior and Redundancy

Horn expression should be acyclic to find a common ground. There is a need to check if any cycle formed by clauses, to check the cyclicity. In Algorithm 2, cycles are removed if exists. Implementation of this part contains the following steps.

- A function *check_cycle* is defined that detects cycles and removes these from $\mathcal{F}$. A function *DiGraph* from *Networkx* [17] is used to draw directed graphs from clauses of the Horn expression. Antecedents and concludents of the clauses act as nodes of the directed graph, while edges are drawn as implications occur in $\mathcal{F}$.

- A built-in function, *is_directed_acyclic_graph* from *Networkx* [18], is used to check the cyclic behavior of the graph.

- If there is no cycle in the graph then the function returns the Horn expression as it is. If there is a cycle then it uses control loop to check for all possible cycles in the expression. It uses a built-in function *find_cycle* from *Networkx* [19] to find the cycle from graph, which returns a list of edges of the cycle along with direction, *forward* or *backward*. It is an ordered list of edges where last edge of the list should meet the first edge to complete the cycle. Last edge of the list is removed to make the sequence of clauses acyclic. It does the same for each iteration of control loop until all cycles are removed and return an acyclic Horn expression,

```python
def check_cycle(F):

    fig, ax = plt.subplots(figsize=(6, 6))
    G = nx.DiGraph()
    for clause in F:
        G.add_node(clause.args[1])
        G.add_node(clause.args[0])
        G.add_edge(clause.args[0],clause.args[1])

    #draw graphs to check cycles
    options = {"edgecolors": "tab:gray", "node_size": 2550, "alpha": 0.9,"font_size":
    ↪  13}
    pos = nx.shell_layout(G)
    nx.draw(G, pos, **options, with_labels = True,
    ↪  node_color="tab:pink",arrowsize=20, edge_color = 'green')
    plt.axis("off")
    plt.show()

    #detect cycle in graph
    if nx.is_directed_acyclic_graph(G):
        print('\n---------This expression has no cycles---------\n')
        return F

    else:
        while not nx.is_directed_acyclic_graph(G):
            cycle = nx.find_cycle(G, orientation="original")
            print('This expression has cycle: ', cycle)
            tuple_clause = cycle[-1][:-1]
            print(tuple_clause)
            clause_to_remove = Implies(tuple_clause[0], tuple_clause[1])
            expr_without_cycle = [x for x in F if x != clause_to_remove]

        print('\n-----------------------------------------------------')
        print('\nHorn Expression after removing cycle is: ')
        return expr_without_cycle

```

Listing 4.4: Structure for *Cycles* in Horn expression

We use an example to show how its implementation works.

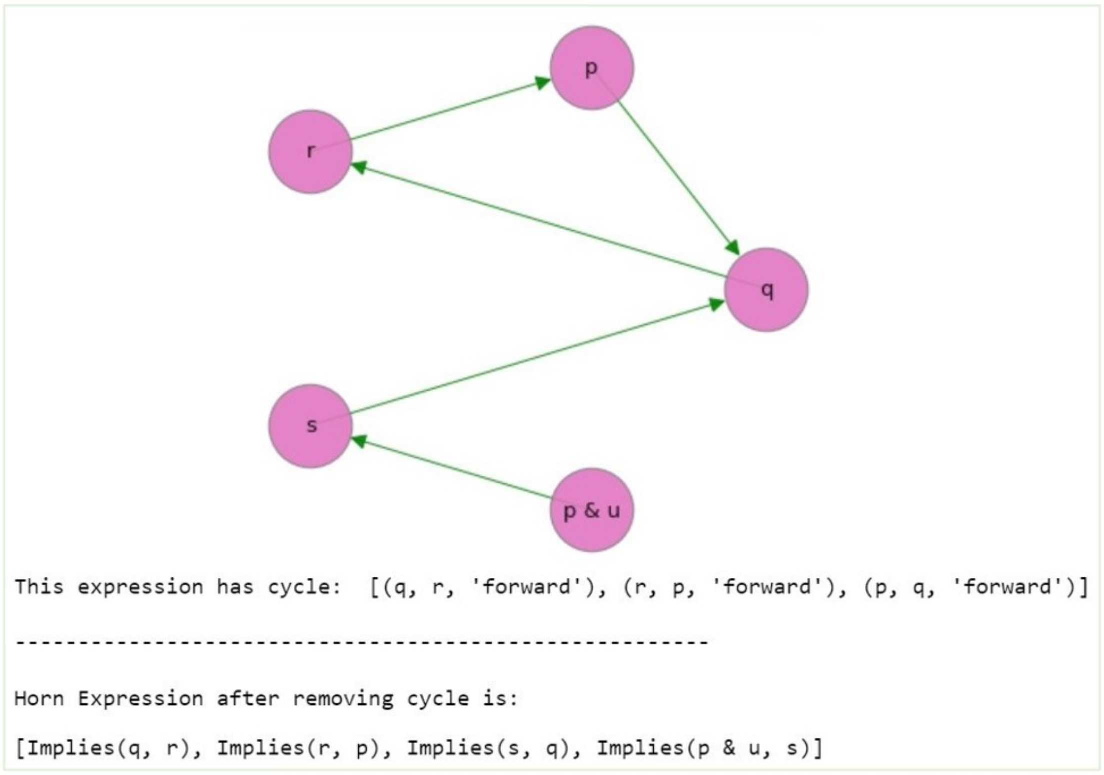
Example 4.2.1. Consider Horn Expression $\mathcal{F} = \bigcup_{i=1}^{5} \phi_i$, and clauses ϕ_i are defined as:

$$\begin{aligned}
\phi_1 &= \{p \rightarrow q\},\\
\phi_2 &= \{(p \land u) \rightarrow s\},\\
\phi_3 &= \{q \rightarrow r\},\\
\phi_4 &= \{r \rightarrow p\}\\
\phi_5 &= \{s \rightarrow q\}.
\end{aligned}$$

It has a sequence of clauses as $(\phi_1, \phi_3, \phi_4, \phi_1)$ which makes a cycle.

$$p \to q\ (\phi_1),\ q \to r\ (\phi_3),\ r \to p\ (\phi_4).$$

Algorithm removes the clause of the sequence that is completing a cycle. It might be ϕ_4 or ϕ_1 depending on the order in which edges are occurring. In the code, the function takes this Horn expression, draws a directed graph, checks for cycles, and removes those. The result of implementation appears as:



```
This expression has cycle:  [(q, r, 'forward'), (r, p, 'forward'), (p, q, 'forward')]
----------------------------------------------------
Horn Expression after removing cycle is:
[Implies(q, r), Implies(r, p), Implies(s, q), Implies(p & u, s)]
```

Figure 4.1: Output for cyclic Horn expression

This shows that the function has removed the cycle by removing clause $\phi_1 :\ p \to q$.

Consider a Horn expression with no cycle Clauses are defined as:

$$\phi_1 = \{p \to q\},\quad \phi_2 = \{(p \wedge u) \to s\},\ \phi_3 = \{q \to r\},\ \phi_4 = \{s \to q\}$$

Similarly function takes this Horn expression, draws a directed graph, checks for cycles, and returns the same expression if there is no cycle. The result of implementation appears as:

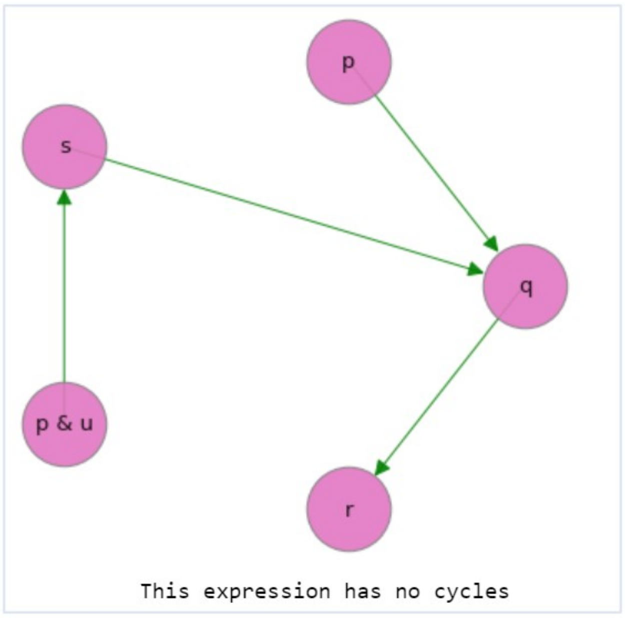


Figure 4.2: Output for acyclic Horn expression

As we know, the Horn expression should be non-redundant to find a common ground. Algorithm 2 remove redundancy if exists in the Horn expression. We remind the definition to check redundancy.

A Horn expression $\mathcal{F}$ is non-redundant if, for all $\phi \in \mathcal{F}$, it is not the case that

$$\mathcal{F} \setminus \{\phi\} \models \phi$$

Implementation of redundancy part has the following steps.

- A function called *non_redundant_expr* is defined. It takes $\mathcal{F}$ as input and checks if expression is redundant or not using above definition.(Appendix A.5).
- A second function name *check_redundancy* is defined. It keeps checking for redundant clauses using control loop. Meanwhile, it keeps removing the redundant clauses and updates the expression $\mathcal{F}$ until it becomes non-redundant. If the given expression is already non-redundant, it is returned as it is.

```python
def check_redundancy(F):
    if non_redundant_expr(F):
        print('-------This Horn Expression is already Non-Redundant-------')
        return T
    else:
        for clause in F:
            expr_without_clause = [x for x in F if x != clause]
            if entails(clause, expr_without_clause):
                F = expr_without_clause
                while non_redundant_expr(F):
                    print('-------This Horn Expression is Non-Redundant now------')
                    return T
```

Listing 4.5: Structure for *Redundancy*

4.3 Conflict

Definition of conflict is recalled, to implement conflict.

Let $\mathcal{F}$ be a definite Horn expression. We say that $\mathcal{F}$ is **in conflict** if we can find any pair of clauses $(\phi, \psi) \in \mathcal{F}$ such that

- $\phi \Rightarrow_{\mathcal{F}} \psi$ and $con(\phi) \in \overline{con(\psi)}$ (i.e., $\mathcal{F}$ is incoherent); and
- there is no $r \in ant(\phi) \setminus ant(\psi)$ with $q \in \overline{r}$ s.t. ψ^{+q} is coherent with $\mathcal{F} \setminus \{\psi\}$.

There are two conditions in a conflict, *incoherence* and *conflict condition*. Following steps are followed for implementation:

- A function named *check_conflict_clauses* is defined. It checks *conflict condition* only for a pair of clauses. It takes a pair of clauses, Horn expression $\mathcal{F}$, and background knowledge $\mathcal{B}$ as input. It extracts antecedents of both clauses. It checks if there is any atom in differences of both clauses' antecedents, where exlcudents are coherent with $\mathcal{F} \setminus \{\psi\}$. If not then these clauses are not in conflict. The complete code can be seen here in Appendix A.6
- second function named $check\_conflict\_expr$ is defined to check conflict of whole horn expression $\mathcal{F}$. It checks both conditions *coherence* and *conflict condition* for all possible pair combinations of the clauses. If condition is true for any of the pairs, $\mathcal{F}$ would assume to be in conflict.

```python
def check_conflict_expr(F, background_know):

    count = 0
    combinations = generate_combinations(F)

    for clause in combinations:
        con_clause1 = next(iter(extract_ant_con(clause[0])[1]))
        con_clause2 = next(iter(extract_ant_con(clause[1])[1]))

        if check_conflict_clauses(F, clause[0], clause[1], background_know) and
        ↪  clauses_derivation(clause[0], clause[1], F) and con_clause1 in
        ↪  get_excludents(background_know, con_clause2):
            count +=1

    if count > 0:
        return True
    else:
        return False
```

Listing 4.6: Structure of *Conflict* in Horn expression

4.3.1 Resolve Conflict

As we know that $\mathcal{F}$ should not be in conflict to find a common ground. Algorithm 2 resolves the conflict, where implementation contains the following steps.

- A function *resolve_conflict* is defined. It takes Horn expression $\mathcal{F}$, both background knowledge $\mathcal{B}$, $\overline{\mathcal{B}}$ and collection of symbols $\mathcal{C}$ as input.
- It uses a control loop to look for conflict in the Horn expression. In each iteration, it checks conflicts in all possible pair combinations of clauses in $\mathcal{F}$. It resolves the conflict by weakening second clause of the pair using generalized symbols from $\overline{\mathcal{B}}$. It uses specific symbols where clauses with generalized symbols are still in conflict. Meanwhile, it keeps updating the expression (using generated copy of expression) by weakened clauses and checking the expression for conflict. At the end, it returns a Horn expression with distinct clauses.

```python
def resolve_conflict(F, background_know, col_symbols, background_know_2):
    copy_expr = copy.deepcopy(F)

    while check_conflict_expr(copy_expr, background_know):
        dict_of_clauses = {i : copy_expr[i] for i in range(0, len(copy_expr))}
        for i in generate_combinations(dict_of_clauses):
            clause1 = dict_of_clauses[i[0]], clause2 = dict_of_clauses[i[1]]
            con_clause1 = next(iter(extract_ant_con(clause1)[1]))
            con_clause2 = next(iter(extract_ant_con(clause2)[1]))

            if check_conflict_clauses(copy_expr, clause1, clause2, background_know)
            ↪  and clauses_derivation(clause1, clause2, copy_expr) and con_clause1
            ↪  in get_excludents(background_know, con_clause2):

                for region in background_know_2:
                    if clause2 in F:
                        if col_symbols[i[1]] in region.args:
                            update_ant = clause2.replace(clause2.args[0],
                            ↪  And(clause2.args[0], region.args[1]))
                    else:
                        if
                        ↪  clause1.args[0].atoms().intersection(region.args[1].atoms())
                        ↪  ==
                        ↪  clause2.args[0].atoms().intersection(region.args[1].atoms()):
                            clause1 = clause1.subs(region.args[1], col_symbols[i[0]])
                            clause2 = clause2.subs(region.args[1], col_symbols[i[1]])
                        copy_expr[i[0]] = clause1
                        copy_expr[i[1]] = clause2
    return [*set(copy_expr)]
```

Listing 4.7: Structure for *Resolving Conflict* in Horn expression

4.4 Resolve Incoherence

A Horn expression should be fully coherent for common ground to exist. If there is incoherence then it can be resolved according to Algorithm 1. A *safe pair* is searched and weaken version of a clause of pair is generated for resolving incoherence. Its implementation contains the following steps.

- As any pair of clauses is safe if it has no parent in the dependency graph of the Horn expression $\mathcal{F}$. From lemma 1, a safe pair is guaranteed to exist in case of incoherence. These steps are followed to search safe pairs:
    - ✳ A function named *get_safe_pairs* is defined. It returns safe pairs if $\mathcal{F}$ is incoherent and returns *False* if there are no safe pairs (means $\mathcal{F}$ is coherent).

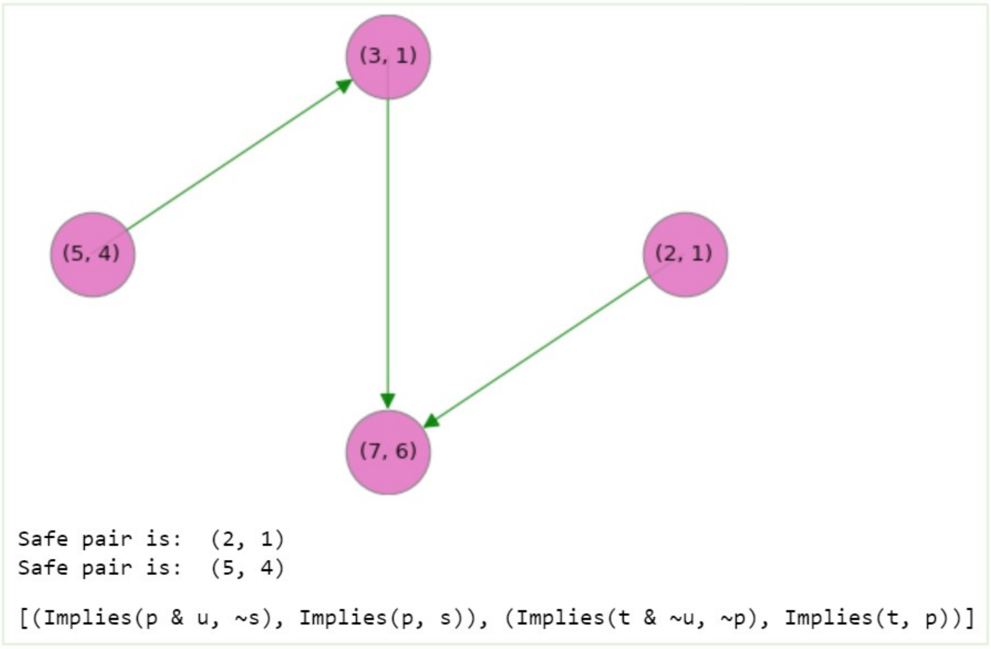
Networkx is used to sketch dependency graphs from the Horn expression. According to definition of dependency graphs 6, nodes are the pairs of clauses (ψ, ϕ) such that $\psi \Rightarrow_{\mathcal{F}} \phi$ and $con(\psi) \in \overline{con(\phi)}$ while edges are a set of all pairs of clauses $((\psi', \phi'), (\psi, \phi))$ such that $\phi' \neq \phi$ and ϕ' occurs in a derivation of ϕ w.r.t. ψ and $\mathcal{F}$.

✳ A built-in function *predecessors* [20] from *Networkx* is used in drawing the dependency graph. It provides the predecessor nodes in a directed graph. Then implementation searches for the nodes with no predecessors. These are the safe pairs. The complete code can be seen in Appendix A.7.

Example 4.4.1. Consider $\mathcal{F} = \bigcup_{i=1}^{7} \mathcal{F}_i$ is not in conflict, not redundant, acyclic, but incoherent.

$$\begin{aligned} &\mathcal{F}_1 = \{p \to s\}, && \mathcal{F}_2 = \{(p \wedge u) \to \overline{s}\}, \\ &\mathcal{F}_3 = \{(t \wedge q) \to \overline{s}\}, && \mathcal{F}_4 = \{t \to p\}, \\ &\mathcal{F}_5 = \{(t \wedge \overline{u}) \to \overline{p}\}, && \mathcal{F}_6 = \{s \to q\}, \\ &\mathcal{F}_7 = \{(p \wedge t) \to \overline{q}\}. \end{aligned}$$

This Horn expression is used in our implementation. It draws a dependency graph, check predecessors in the graph and gives nodes that have no parent. The implementation result from this Horn expression looks like this:



```
Safe pair is:  (2, 1)
Safe pair is:  (5, 4)

[(Implies(p & u, ~s), Implies(p, s)), (Implies(t & ~u, ~p), Implies(t, p))]
```

Figure 4.3: Output of *Safe pairs* function for Horn expression

Dependency Graph shows nodes (5,4) and (2,1) which have no predecessor. Both $(\mathcal{F}_5, \mathcal{F}_5)$ and $(\mathcal{F}_2, \mathcal{F}_1)$ are safe pairs.

- It generates a weak version of a clause from safe pairs of clauses after finding safe pairs. It has the following steps:

  ✳ A function named *weaker_version_clause* is defined. It takes $\mathcal{F}$ and clauses of safe pairs as input. Following part of the algorithm is used to generate a weaker version:

  > If we have $\psi, \phi \in \mathcal{F}$ such that (ψ, ϕ) is *safe*. Replace ϕ by all $\phi' \in \{\phi^{+p} \mid p \in \bar{l}, l \in ant(\psi) \setminus ant(\phi)\}$ coherent with $\mathcal{F} \setminus \{\phi\}$.

  The function takes the second clause and modifies it by adding an atom to the antecedents of the clause such that it becomes coherent. The complete code can be seen in Appendix A.8.

- All features of Algorithm 2 has been implemented. Now we implement the final structure of the algorithm to find a common ground for a given Horn expression $\mathcal{F}$.

4.5 Finding Common Ground Algorithm

To implement the algorithm 2, a function named *find_common_ground* is defined. It takes a Horn expression $\mathcal{F}$, both background knowledge $\mathcal{B}$, $\overline{\mathcal{B}}$ and collection of symbols $\mathcal{C}$ as input and returns a common ground for $\mathcal{F}$. This function contains the following steps.

- Firstly, it checks for cycles and removes cycles using the implemented function *check_cycle* that returned acyclic expression.
- Secondly, it gives a non-redundant expression while removing redundancies using *check_redundancy* function.
- Thirdly, it checks for conflict in $\mathcal{F}$ and resolve it using implemented functions *check_conflict_expr* and *resolve_conflict*.
- In the end, a while loop is used for coherence of $\mathcal{F}$. It checks for the coherence of Horn expression using implemented functions *HornExpression_coherence*. If it is incoherent then it searches for the safe pair. It uses *get_safe_pairs* and generates

a weaker version of the clause using *weaker_version_clause*. Horn expression is updated according to updated clauses (weakened clauses). That is a common ground for $\mathcal{F}$.

```python
def find_common_ground(F, background_know, background_know_2, col_symbols):

    F = check_cycle(F)
    F = check_redundancy(F)

    if not check_conflict_expr(F, background_know):
        print('\n-------------- F is not in conflict--------------\n')
    else:
        F = resolve_conflict(F, background_know, col_symbols, background_know_2)
        print('\n-------------- Conflict is resolved --------------\n', F)

    while not HornExpression_coherence(F, background_know):
        print('\n-------------- Resolving Incoherence --------------\n')

        for safe_pair in get_safe_pairs(F):
            clause1 = safe_pair[0], clause2 = safe_pair[1]
            weaker_version, indices = weaker_version_clause(F, clause1, clause2)
            F = [weaker_version if F.index(x) in indices else x for x in F]

    print('\n-------------- Expression is coherent now --------------\n')
    return F
```

Listing 4.8: Structure for *Finding Common ground Algorithm* from Horn expression

Chapter 5

Experiment

This chapter begins by presenting the Moral Machine experiment. It follows by analyzing the moral machine dataset and applying common ground algorithm 2 on this dataset. A study is performed on dataset along with preprocessing to make the dataset suitable for the algorithm. In preprocessing, rules are extracted from dataset in the form of definite horn clauses that makes a horn expression. To prepare data, both background knowledge and collection of symbols are defined accordingly. Finally, the results of the algorithm's application on the dataset are presented.

5.1 Moral Machine Experiment

The use of self-driving cars on the roads, will soon become a reality, but questions like responsibility and priorities are still unanswered. Awad's research group set up "The Moral Machine Experiment", an online platform. This is an online experimental platform designed to explore the moral dilemmas faced by autonomous vehicles. This platform gathered 40 million decisions in ten languages from millions of people in 233 countries and territories. Its objective is to quantify societal expectations about ethical principles and to use those as guidance for machine behavior [21].

The study deals with the expectations that people have from the ethical behavior of AI. The goal of this experiment is to measure the moral preferences which adopted in case of accidents with self-driving cars. Car manufacturers and policymakers are currently struggling with setting a benchmark for these moral dilemmas due to the complex nature. It is challenging to solve it by a simple normative ethical set of principles.

We imagine at some point in future, shall be driving down the highway in a self-driving vehicle, boxed in on all sides by other vehicles. Inevitably, we might find ourselves stuck in a life-threatening situation where our car would not stop with in time to avoid a collision. It has a choice, either collide with one of the fellow vehicles, endangering other passenger or its own. What should a self-driving vehicle do in such context? If we are driving a traditional non-autonomous vehicle, whichever way we choose, it would be considered a reaction to the situation as opposed to a deliberate decision—an instinctual, potentially panicked reaction with no forethought or malice. A programmed, self-driving vehicle would, at some point, take a life to save another. Which subject needs to be saved, whereas morality dictates saving both lives? The moral machine experiment is all about finding answers to such morally grim questions [22].

In the moral machine experiment, participants are asked to form judgments about variations of the well-known trolley problem. It is a thought experiment in ethics about a fictional scenario. An onlooker has the choice to save five persons in danger of being hit by a trolley, by diverting the trolley to kill just one person.

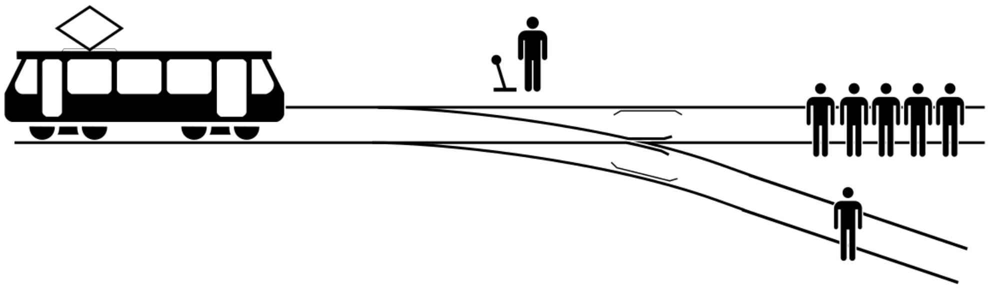


Figure 5.1: Trolley Problem

The variations replace trolleys with autonomous vehicles, and people on tracks with passengers and pedestrians. The Moral Machine takes the idea to test nine different comparisons shown to polarize people: should a self-driving vehicle prioritize humans over pets or passengers over pedestrians, etc.

In the main interface of the Moral Machine, users are shown unavoidable accident scenarios with two possible outcomes. These depend on whether the automated vehicles swerves or stays on course. Users can click on the outcome that they find preferable. A sample scenario given below depicts what the Moral Machine presents to get human perspective.

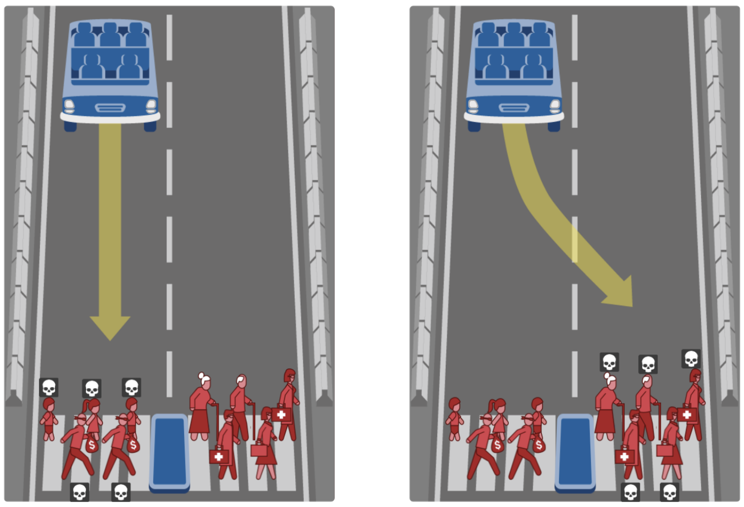


Figure 5.2: Sample scenario of Moral Machine experiment

In the left case of 5.2, self-driving car in case of brake failure continues ahead and drive through a pedestrian crossing ahead. This results in

Dead: **2 boys 1 girl 2 criminals**

In the right case of 5.2, self-driving car with brake failure swerves to other lane and drives through a pedestrian crossing. This results in:

Dead: **1 elderly woman1 elderly man 1 female doctor 1 male doctor 1 female executive**

Each dilemma presents two potential negative outcomes. Each results in loss of lives. The number, gender, and age, along with other factors of surrounding characters and environment in each outcome, are variable at each occurrence. For each scenario, a choice for the preferred outcome has to be made. At the end of the experiment, a summary of decisions exercised is presented. It accompanies by a comparison and an optional survey.

Moral Machine follows an exploration strategy to generate accident scenarios. It focuses on nine factors which are staying on course (vs. swerving) and saving humans (vs. pets), passengers (vs. pedestrians), more lives (vs. fewer lives), men (vs. women), young (vs. the old), pedestrians who cross legally (vs. jaywalk), healthy (vs. sick), high social status (vs. low social status). Additional characters are included in some scenarios (e.g., criminals, pregnant women, doctors) These are not linked to any of the above mentioned nine factors. These characters mostly served to make scenarios less repetitive for the users. Participants can complete a survey after completing a thirteenth accident session. Survey also collects, among other variables, demographic information such as gender, age, income, and education, religious and political attitudes. Participants are geolocated. Their coordinates can be used in a clustering analysis. It helps to identify groups of countries or territories with homogeneous vectors of moral preferences [5].

The researchers found that countries' preferences differ widely, but these also correlate highly with culture and economics. For example, participants from collectivist cultures like China and Japan are less likely to spare the young over the old. Researchers hypothesized, the high respect given to elders. [23].

5.2 Findings of Moral Machine Survey

The Moral Machine compiled nearly 40 million individual responses from around the world. The researchers analyzed the data as a set. They break out participants into subgroups defined by age, education, gender, income, and political and religious views. The team finds few significant moral differences based on said characteristics.

Figure 5.3 represents a world map highlighting the locations of Moral Machine visitors. Each point represents a location from which at least one visitor made at least one decision [5]. The number of visitors or decisions from each location is not represented.

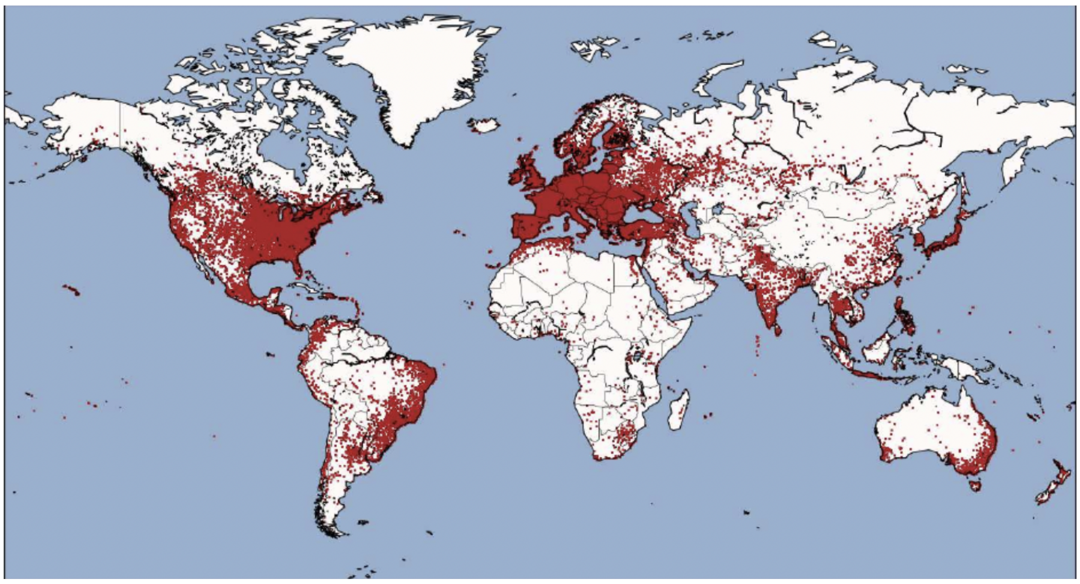


Figure 5.3: Moral Machine Map

However, research finds clusters of preferences based on cultural and geographic affiliations. Geolocation allowed us to identify the country of residence of Moral Machine respondents. It aids to seek clusters of countries exhibiting homogeneous vectors of moral preferences. It divides the world into three dominant clusters as *Western*, *Southern*, and *Eastern* as shown in Figure 5.4 [5].

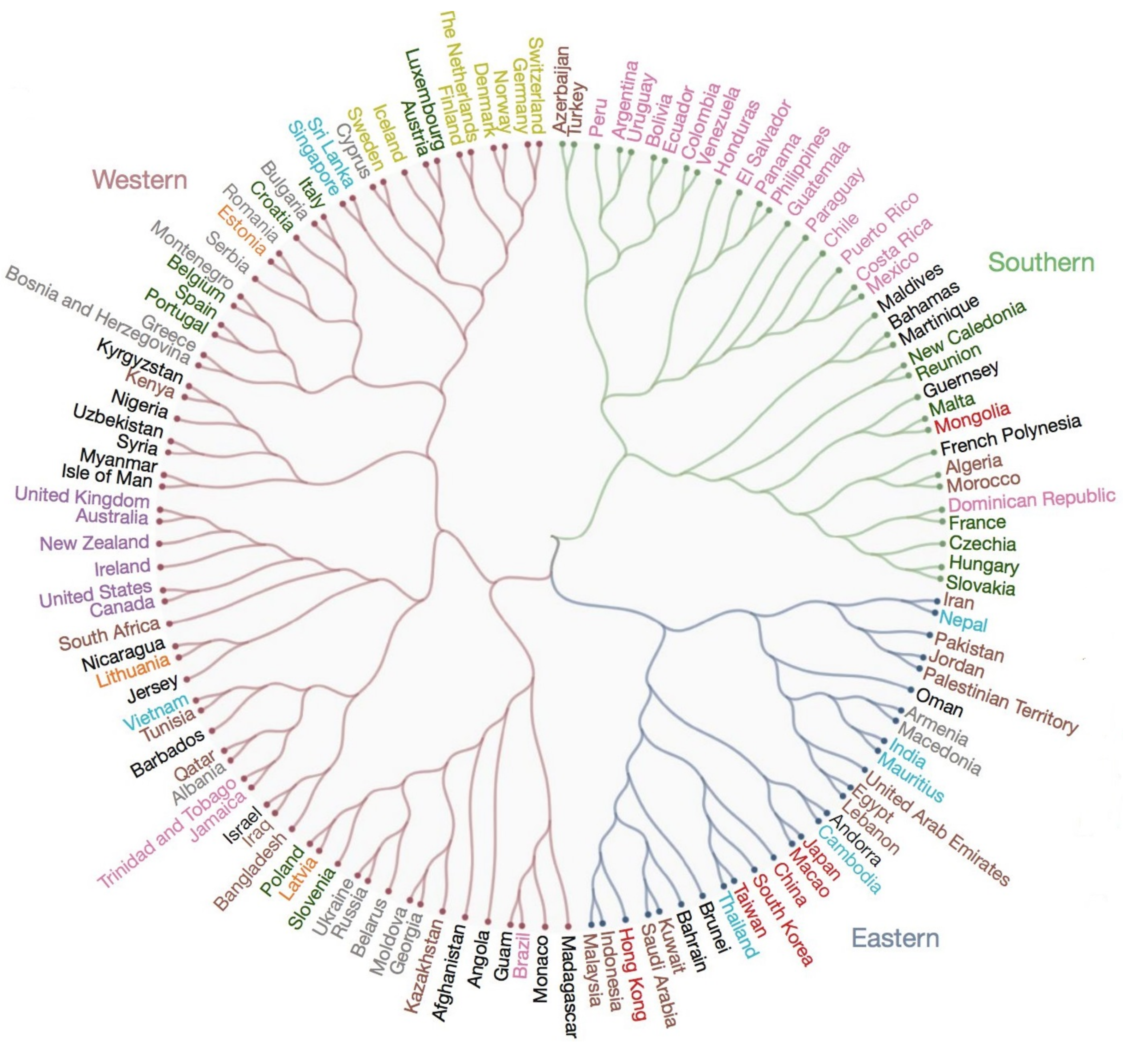


Figure 5.4: Countries clusters of Moral Machine

The first cluster, labeled as Western cluster, contains North America and many European countries of Protestant, Catholic, and Orthodox Christian cultural groups. The internal structure within this cluster also exhibits notable face validity, with a sub-cluster containing Protestant/Scandinavian countries. It has a sub-cluster containing Commonwealth/English-speaking countries.

The second cluster called Eastern, contains many far eastern countries. Such as

Japan and Taiwan, belongs to the Confucianism cultural group, and Islamic countries like Indonesia, Pakistan, and Saudi Arabia.

The third cluster (a broadly Southern cluster) consists of the Latin American countries of Central and South America. It also has countries that are characterized in part by French influence e.g., metropolitan France, French overseas territories, and territories that were at some point under French leadership. Latin American countries are cleanly separated in their own sub-cluster within the Southern cluster.

The research analysts used the following Spiderweb chart to interpret the survey data:

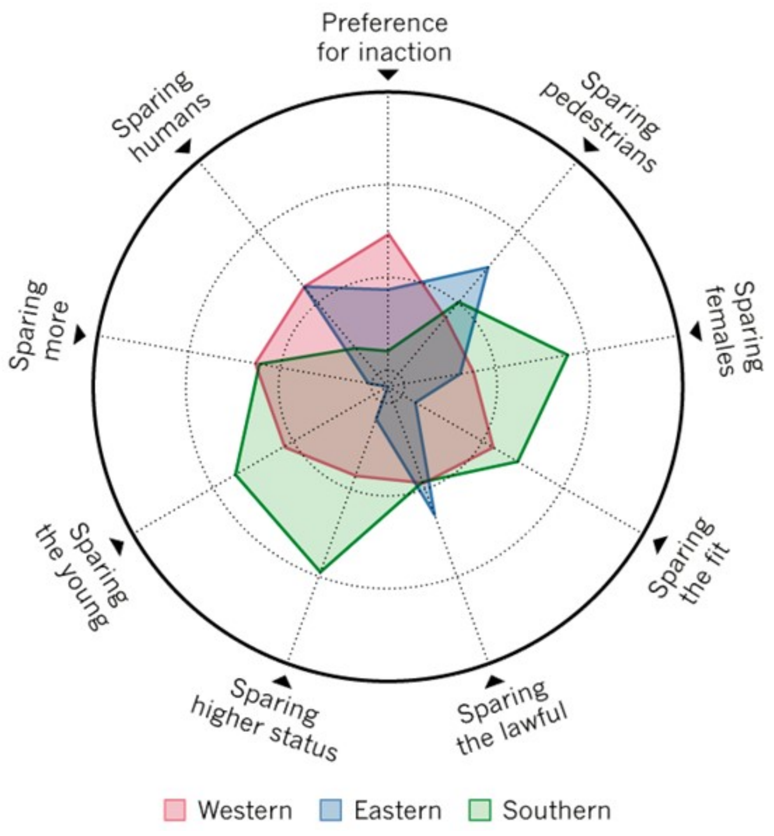


Figure 5.5: Clusters' Tendencies of Moral Machine

Overall, the researchers found three elements that people most agreed on. Majority of people believe in saving the lives of *humans* over *animals*, *many* rather than *few* and *young* rather than *old*. It is not simple. The degree of respondents' agreement or disagreement to above said principles, varies by different groups and countries. Research finds Asia and the Middle East countries like China, Japan, and Saudi Arabia, prefer to spare younger rather than older character. People from these countries also care relatively less about sparing high net-worth individuals compared to people who answered from Europe and North America. The respondents in southern countries have a relatively stronger preference for saving young people over the old and shielding higher-status characters.

Similarly, countries in the Southern cluster exhibit a much weaker preference for saving humans over pets, compared to the other two clusters. Only the (weak) preference for sparing pedestrians over passengers and the (moderate) preference for sparing the lawful over the unlawful appear to be shared to the same extent in all clusters [5]. There are also some striking peculiarities, like the strong preference for sparing women and the strong preference for sparing fit characters in the Southern cluster.

Based on the initial Moral Machine findings, morality differs, depending on several factors. These may be race, culture, religion, country, economy, and so on. This is for sure that morality debate will continue along with the prosperity in automation (self-driving vehicles and consumer devices). Whether the Moral Machine settles the ethical debate to some extent, but it began an enhancement in ethical discussions for the world to consider.

5.3 Moral Machine Dataset

This includes a discussion about the dataset of Moral Machine. It is going to be used in our Common ground Algorithm. The data is collected from the respondents of experiments from all over the world. We do a brief look at a random row of the data and explain what each column shows. The data file contains forty-one columns. The last twenty columns represent the number of characters of each type in each outcome. For example, the columns “Man” and “Woman” represent the number of men and Women characters in each outcome (both are zero in this example).

ResponseID	2224g4ytARX4QT5rB
ExtendedSessionID	213978760_9992828917431898.0
ScenarioOrder	7
Intervention	0
PedPed	0
Barrier	1
CrossingSignal	0
AttributeLevel	Less
ScenarioType	Utilitarian
DefaultChoice	More
NonDefaultChoice	Less
DefaultChoiceIsOmission	0
NumberOfCharacters	4
Saved	1
UserCountry3	USA
Man	0
Woman	0
Pregnant	0
Stroller	1
OldMan	0
OldWoman	0
Boy	0
Girl	0
Homeless	0
LargeWoman	0
LargeMan	0
Criminal	0
MaleExecutive	0
FemaleExecutive	0
FemaleAthlete	1
MaleAthlete	0
FemaleDoctor	1
MaleDoctor	0
Dog	0
Cat	1

Table 5.1: Random row of Moral Machine Dataset

A quick look at this example, shows the row contains a baby stroller, a female athlete, a female doctor, and a cat as characters involved in the scenario. Each type has only one character, but each of these cells could take a value between 0 and 5. It has crucial restriction of total number of characters in each outcome (cell) is between 1 and 5. This number is captured in the column “NumberOfCharacters”. Further details about the remaining columns is given below:

- ***ResponseID***: a unique, random set of characters that represents an identifier of the scenario.
- ***ExtendedSessionID***: a unique, random set of characters that represents an identifier of the session.
- ***ScenarioOrder***: it takes a value between 1 and 13, representing the order in which the scenario was presented in the session.
- ***Intervention***: represents the decision of the automated vehicles (STAY or SWERVE) that would lead to this outcome [0: the character would die if the automated vehicles stays, 1: the character would die if automated vehicles swerves]. This is not the actual decision taken by the user, but rather a part of the structural characterization of the scenario.
- ***PedPed***: every scenario has either pedestrians vs. pedestrians or pedestrians vs. passengers (or passengers vs. pedestrians). This column provides information about not just this outcome, but about the combination of both outcomes in the scenario; whether the scenario pits pedestrians against each other or not [1: pedestrians vs. pedestrians, 0: pedestrians vs. passengers (or vice versa)]
- ***Barrier***: a structural column that describes whether the potential casualties in this outcome are passengers or pedestrians [1: passengers, 0: pedestrians]. This column was used to calculate PedPed.
- ***CrossingSignal***: a structural column which represents whether there is a traffic light in this outcome, and light color if yes [0: no legality involved, 1: green or legally crossing, 2: red or illegally crossing]
- ***Saved***: this resembles the actual decision made by the user [1: user decided to save the characters in this outcome, 0: user decided to kill the characters in this outcome].
- ***NumberOfCharacters***: takes a value between 1 and 5, the total number of characters in this outcome. This is the sum of numbers in the last 20 columns (character columns).
- ***UserCountry3***: This column consists of alpha-3 ISO code of the country from

which the user accessed the website

- ***ScenarioType***: This column has 7 values, corresponding to 7 types of scenarios (6 attributes + random). These are “Utilitarian”, “Gender”, “Fitness”, “Age”, “Social Status”, “Species”, and “Random”.
- ***AttributeLevel***: is dependent on the scenario type. Each scenario type (except random) has two levels:
  - ✳ -*Gender*: [Males: characters are males, Females: characters are females]
  - ✳ -*Age*: [Young: characters in this outcome are younger (Boy/Girl + Man/-Woman) than in the other outcome, Old: characters in this outcome are older (Elderly Man/Woman and Man/Woman)].
  - ✳ -*Fitness*: [Fit: characters in this outcome are more fit (Male/Female Athlete and Man/Woman), Fat: characters in this outcome are less fit (Large Man/-Woman and Man/Woman)].
  - ✳ -*Social Status*: [High: characters in this outcome have higher social status (Male/Female Executives, Pregnant women), Low: characters have a lower social status (Homeless, Criminal)]
  - ✳ -*Species*: [Humans: characters in this outcome are humans, Pets: characters in this side are pets (Dog/Cat)]
  - ✳ -*Utilitarian*: [More: there are more characters in this outcome, Less: there are fewer people in this outcome]. In fact, the characters on the ”More” side are the same characters on the ”Less” side, in addition to at least one more character.
  - ✳ -*Random*: it has one value [”Rand”: characters in both outcomes are randomly generated].
- ***DefaultChoice, NonDefaultChoice***: The default Choice depends on the Scenario Type. For the following Scenario Types: [“Gender”, “Fitness”, “Age”, “Social Status”, “Species”, “Utilitarian”], the default choice is [“Male”, “Fit”, “Young”, “High”, “Humans”, “More”], while the non-default choice is [“Female”, “Fat”, “Old”, “Low”, “Pets”, “Less”].
- ***DefaultChoiceIsOmission***: Omission here means no intervention (Intervention

= 0), and the default choice is as described in the "DefaultChoice" column. When DefaultChoiceIsOmission = 1 it means that based on the scenario type, characters that hold the default choice (that is, males for gender, fit for fitness, young for age,. . . etc.) will be the ones killed if the automated vehicles does nothing (omission or no intervention). On the other hand, if DefaultChoiceIsOmission = 0, then the characters that hold the non-default choice will be the ones killed if the automated vehicles does nothing (i.e., the characters holding the default choice will be the ones killed if the car swerves (i.e., intervenes)).

After explaining the columns, we get back to the example scenario discussed above. It represents the dilemma in where four (4) passengers are in the automated vehicle and five (5) pedestrians who are crossing legally. There is a barrier in front of the AV. If the automated vehicle stays, then it will hit the barrier and kill the four (4) passengers. If it swerves, then it will save the passengers. The user chooses automated vehicle to SWERVE and thus has decided to spare the four passengers.

5.4 Preprocessing of Data

Now we have explained the dataset and in order to use this dataset for our algorithm we need to prepare the dataset for the algorithm. This means we extract rules (preferences from participants) from the dataset in the form of definite horn clauses that we can use as a Horn expression.

Our preprocessing consists of the following steps:

- After loading the data file, we drop rows where the column '*ScenarioType*' has value 'random' and rows where there is a column with a '*none*' value.
- We convert the value of *UserCountry3* from the alpha-3 ISO code to the full of the name country. For example, **FRA** would appear as **france**.
- Dataset contains 233 countries. It is not possible to use the data for all countries. We filter the dataset to analyze specific countries and focus on some scenarios types for readability of the rules created for our experiment for finding common grounds. The filtration is as follows:
    - ✳ We define a function that accesses the dataset and list of countries as input. It uses this list to select only the data from these countries.

- We use the data where only pedestrians are involved. That means we ignore the scenarios where the combination of passengers and pedestrians is used.

- We use the data, where either the traffic light is green or red. That means we ignore the scenarios where no lights are involved. Here 'red' means that the traffic signal has been violated and 'green' means it has been respected. In other words, subjects are law-abiding.

- We ignore columns of IDs, and columns representing the number of characters of each type in each outcome. Then we are left with the columns *'CrossingSignal', 'ScenarioType', 'AttributeLevel'* and *'UserCountry3'*. We group the dataset according to these columns to check the frequency of each scenario as saved or not saved (by the *'saved'* column). We ignore duplicate rows.

  For example, in the dataset, we have rows where there is "utilitarian" as *ScenarioType*, "less" as *AttributeLevel*, "green light" as *CrossingSignal*, and USA as *UserCountry3* and 1 or 0 as *saved* or *Not saved*. The goal is to see for a specific scenario how many participants believe that the characters involved would be saved or sacrificed. In this way, each scenario would be associated with 2 rows, one saved as 0 and one with saved as 1, with the respective frequencies. We keep scenarios with a higher frequency. Table 5.2 contains the information of the scenarios after grouping.

CrossingSignal	1	1
ScenarioType	Utilitarian	Utilitarian
AttributeLevel	Less	Less
UserCountry3	USA	USA
Saved	1	0
Frequency	200	300

Table 5.2: Example of grouping scenarios

  In Table 5.2, 200 people are in favor to save the people involved in this scenario, while 300 people are willing to sacrifice. By frequency, we ignore the data of lower frequency. After dropping lower-frequency rows, is the filtered data for our algorithm. The complete code can be seen here in Appendix A.9.

- The next step is to extract rules from the filtered dataset. For this, a function is defined that treats each row of the dataset as a rule. Columns *CrossingSignal, ScenarioType, AttributeLevel* and *UserCountry3* are used as antecedents of rules while *Saved* column is consequent (as value *Saved* and *not_Saved*). The collection of rules from a country form a Horn expression that represents the moral recommendations of a country regarding these attributes. For each row, the associated country name from column *UserCountry3* is served as a distinct symbol in the collection of symbols $\mathcal{C}$. The complete code of extracting rules can be seen here in Appendix A.10. The resulting rules would appear as:

$$\phi_1 = \{(\mathsf{green} \land \mathsf{gender} \land \mathsf{male}) \rightarrow \mathsf{saved}\},$$
$$\phi_2 = \{(\mathsf{green} \land \mathsf{species} \land \mathsf{pets}) \rightarrow \mathsf{not\_saved}\},$$
$$\phi_3 = \{(\mathsf{red} \land \mathsf{fitness} \land \mathsf{fat}) \rightarrow \mathsf{not\_saved}\},$$

For each country, the associated country name is a distinct symbol (for each rule) representing the view of that country. These distinct symbols collectively make $\mathcal{C}$. For ϕ_1, ϕ_2, ϕ_3, we have $\mathcal{C} = [\ \mathsf{france},\ \mathsf{denamrk},\ \mathsf{china}\ ]$

- We define both background knowledge as $\mathcal{B}$ and $\overline{\mathcal{B}}$ according to symbols involved in rules. For, $\overline{\mathcal{B}}$ we need specific and generalized symbols.Here $\mathcal{C}$ is countries' names as specific symbols because for each row in the dataset there is a country name. Generalized symbols are countries clusters involved in the moral machine dataset, i.e., $\mathsf{east}, \mathsf{west}, \mathsf{south}$. For, $\overline{\mathcal{B}}$ we define definite clauses that are implying specific symbols to generalized symbols as the countries lie in a cluster. In $\overline{\mathcal{B}}$, clauses are like:

$$\mathsf{spain} \rightarrow \mathsf{west},$$
$$\mathsf{india} \rightarrow \mathsf{east},$$
$$\mathsf{france} \rightarrow \mathsf{south}$$
$$\mathsf{india} \rightarrow \mathsf{east}$$

Where antecedents are from $\mathcal{C}$. Background knowledge $\mathcal{B}$ is a set of Horn constraints for the symbols involved in $\mathcal{F}$. As both attribute levels of the same scenario cannot occur together, they are excludents of each other. Clauses in $\mathcal{B}$ are like:

$$(\mathsf{young} \land \mathsf{old}) \rightarrow \perp$$
$$(\mathsf{female} \land \mathsf{male}) \rightarrow \perp$$
$$(\mathsf{more} \land \mathsf{less}) \rightarrow \perp$$
$$(\mathsf{green} \land \mathsf{red}) \rightarrow \perp$$

We have prepared our dataset as horn expression $\mathcal{F}$. Both background knowledge $\mathcal{B}$ and $\overline{\mathcal{B}}$ and collection of symbols $\mathcal{C}$ have been generated. Now we can use Common ground Algorithm 2 to find common ground for any collection of countries.

5.5 Results

This section focuses on testing the algorithm on prepared dataset of Moral Machine experiment. More details about the experimental results are provided. For testing, we use different countries to find common ground.

There are rules in the form of clauses for different countries. It is quite possible that there are multiple rules with the same antecedents but different concludents for different countries. For example, according to Moral machine experiment, tendency to save young people is higher in the southern cluster as compared to the eastern. If France is selected from the south and China from the east, then following are the rules for saving young people. Here age is scenario type, and the pedestrian crossing signal is green (that means pedestrians are crossing legally).

$$\phi_1 = (\mathsf{green} \land \mathsf{age} \land \mathsf{young}) \to \mathsf{saved}, \quad c_1 = \mathsf{france}$$
$$\phi_2 = (\mathsf{green} \land \mathsf{age} \land \mathsf{young}) \to \mathsf{not\_saved}, \quad c_2 = \mathsf{china}$$

This case would have a conflict because the same scenario is saved and not_saved in the dataset. First an analysis is performed for case of two largest economies in the world: the United States and China. Then, a group of Scandinavian countries are considered. These countries are highly expected to have similar moral recommendations. Finally, a group with a mix of two European countries and an Asian country is evaluated.

5.5.1 Comparison between the United States and China

There are many countries in the dataset which can be used for our experiment. First, the comparison is performed between two largest economies in the world, the United States and China.

Table 5.3 is the Horn expression that is prepared after preprocessing of the data for the United States and China. In the dataset, there are six (6) scenario types. Each scenario has two attribute levels. There are twelve (12) horn rules for each country. In each scenario, red and green (pedestrians) crossing signals lights are considered. In total, there is a Horn expression of 24 clauses for each country. These are the rules in Horn expression along with symbols as country names. The generalized symbol for the United States is **west** and for china is **east**.

1	green ∧ age ∧ old → not_saved	united states
	green ∧ age ∧ old → not_saved	china
2	green ∧ age ∧ young → saved	united states
	green ∧ age ∧ young → saved	china
3	green ∧ fitness ∧ fat → saved	united states
	green ∧ fitness ∧ fat → saved	china
4	green ∧ fitness ∧ fit → saved	united states
	green ∧ fitness ∧ fit → saved	china
5	green ∧ gender ∧ female → saved	united states
	green ∧ gender ∧ female → saved	china
6	green ∧ gender ∧ male → saved	united states
	green ∧ gender ∧ male → saved	china
7	green ∧ socialStatus ∧ high → saved	united states
	green ∧ socialStatus ∧ high → not_saved	china
8	green ∧ socialStatus ∧ low → not_saved	united states
	green ∧ socialStatus ∧ low → not_saved	china
9	green ∧ species ∧ human → saved	united states
	green ∧ species ∧ human → saved	china
10	green ∧ species ∧ pets → not_saved	united states
	green ∧ species ∧ pets → not_saved	china
11	green ∧ utilitarian ∧ less → not_saved	united states
	green ∧ utilitarian ∧ less → not_saved	china
12	green ∧ utilitarian ∧ more → saved	united states
	green ∧ utilitarian ∧ more → saved	china
13	red ∧ age ∧ old → not_saved	united states
	red ∧ age ∧ old → not_saved	china
14	red ∧ age ∧ young → saved	united states
	red ∧ age ∧ young → not_saved	china
15	red ∧ fitness ∧ fat → not_saved	united states
	red ∧ fitness ∧ fat → not_saved	china
16	red ∧ fitness ∧ fit → not_saved	united states
	red ∧ fitness ∧ fit → not_saved	china
17	red ∧ gender ∧ female → not_saved	united states
	red ∧ gender ∧ female → not_saved	china
18	red ∧ gender ∧ male → not_saved	united states
	red ∧ gender ∧ male → not_saved	china
19	red ∧ socialStatus ∧ high → saved	united states
	red ∧ socialStatus ∧ high → not_saved	china
20	red ∧ socialStatus ∧ low → not_saved	united states
	red ∧ socialStatus ∧ low → not_saved	china
21	red ∧ species ∧ human → saved	united states
	red ∧ species ∧ human → saved	china
22	red ∧ species ∧ pets → not_saved	united states
	red ∧ species ∧ pets → not_saved	china
23	red ∧ utilitarian ∧ less → not_saved	united states
	red ∧ utilitarian ∧ less → not_saved	china
24	red ∧ utilitarian ∧ more → saved	united states
	red ∧ utilitarian ∧ more → saved	china

Table 5.3: Horn Expression for the United States and China

As per findings, United States and China have **three** conflicts. These are highlighted as rule numbers $7, 14$ *and* 19. The application of algorithm results in resolution of all the conflicts and finding common ground by resolving incoherence if exists. First resolving conflicts is discussed. In case of rules 7 and 19, the United States prefers to save people with high social status on both red and green signal. It includes both law-abiding and non-law-abiding pedestrians. This is not practiced in case of China. This is a clear conflict between the United States and china for protecting people with high social status. Similarly, in case of rule 14, the United States prefers to save young people when there is a red signal. Whereas, China does not give young people preference if they cross on the red pedestrian signal. The algorithm would solve the conflict using generalized and specific symbols from background knowledge $\overline{\mathcal{B}}$. After resolution of conflicts, there would be no repetitive rules for both countries. It means if there is no symbol associated with the rule then the rule is true for all the countries involved in the dataset.

1	green ∧ age ∧ old → not_saved
2	green ∧ age ∧ young → saved
3	green ∧ fitness ∧ fat → saved
4	green ∧ fitness ∧ fit → saved
5	green ∧ gender ∧ female → saved
6	green ∧ gender ∧ male → saved
7	green ∧ socialStatus ∧ high → not_saved
	green ∧ socialStatus ∧ high ∧ west → **saved**
8	green ∧ socialStatus ∧ low → not_saved
9	green ∧ species ∧ human → saved
10	green ∧ species ∧ pets → not_saved
11	green ∧ utilitarian ∧ less → not_saved
12	green ∧ utilitarian ∧ more → saved
13	red ∧ age ∧ old → not_saved
14	red ∧ age ∧ young ∧ west → **saved**
	red ∧ age ∧ young → not_saved
15	red ∧ fitness ∧ fat → not_saved
16	red ∧ fitness ∧ fit → not_saved
17	red ∧ gender ∧ female → not_saved
18	red ∧ gender ∧ male → not_saved
19	red ∧ socialStatus ∧ high ∧ west → **saved**
	red ∧ socialStatus ∧ high → not_saved
20	red ∧ socialStatus ∧ low → not_saved
21	red ∧ species ∧ human → saved
22	red ∧ species ∧ pets → not_saved
23	red ∧ utilitarian ∧ less → not_saved
24	red ∧ utilitarian ∧ more → saved

Table 5.4: Horn Expression for the United States and China after resolving conflicts

Now the conflict is resolved, and the algorithm resolves incoherence that was present in rule 7, 14 and 19.

1	green ∧ age ∧ old → not_saved
2	green ∧ age ∧ young → saved
3	green ∧ fitness ∧ fat → saved
4	green ∧ fitness ∧ fit → saved
5	green ∧ gender ∧ female → saved
6	green ∧ gender ∧ male → saved
7	green ∧ socialStatus ∧ high ∧ **not_west** → **not_saved**
	green ∧ socialStatus ∧ high ∧ **west** → **saved**
8	green ∧ socialStatus ∧ low → not_saved
9	green ∧ species ∧ human → saved
10	green ∧ species ∧ pets → not_saved
11	green ∧ utilitarian ∧ less → not_saved
12	green ∧ utilitarian ∧ more → saved
13	red ∧ age ∧ old → not_saved
14	red ∧ age ∧ young ∧ **west** → **saved**
	red ∧ age ∧ young ∧ **not_west** → **not_saved**
15	red ∧ fitness ∧ fat → not_saved
16	red ∧ fitness ∧ fit → not_saved
17	red ∧ gender ∧ female → not_saved
18	red ∧ gender ∧ male → not_saved
19	red ∧ socialStatus ∧ high ∧ **west** → **saved**
	red ∧ socialStatus ∧ high ∧ **not_west** → **not_saved**
20	red ∧ socialStatus ∧ low → not_saved
21	red ∧ species ∧ human → saved
22	red ∧ species ∧ pets → not_saved
23	red ∧ utilitarian ∧ less → not_saved
24	red ∧ utilitarian ∧ more → saved

Table 5.5: Common ground for the United States and China

This is the common ground that the algorithm has found after resolving all incoherence for the United States and China. In table 5.5, blue color shows coherence resolved and red color shows resolution of conflict.

5.5.2 Comparison of Scandinavian countries

Data of three Scandinavian countries are used in the second experiment. Those countries are **Norway**, **Denmark** and **Sweden**. It is expected that these share similar moral preferences countries due to similar cultural trends.

1	green ∧ age ∧ old → not_saved	norway, denmark, sweden
2	green ∧ age ∧ young → saved	norway, denmark, sweden
3	green ∧ fitness ∧ fat → saved	norway, denmark, sweden
4	green ∧ fitness ∧ fit → saved	norway, denmark, sweden
5	green ∧ gender ∧ female → saved	norway, denmark, sweden
6	green ∧ gender ∧ male → saved	norway, denmark, sweden
7	green ∧ socialStatus ∧ high → saved	norway, denmark, sweden
8	green ∧ socialStatus ∧ low → saved	norway
	green ∧ socialStatus ∧ low → saved	denmark
	green ∧ socialStatus ∧ low → not_saved	sweden
9	green ∧ species ∧ human → saved	norway, denmark, sweden
10	green ∧ species ∧ pets → not_saved	norway, denmark, sweden
11	green ∧ utilitarian ∧ less → not_saved	norway, denmark, sweden
12	green ∧ utilitarian ∧ more → saved	norway, denmark, sweden
13	red ∧ age ∧ old → not_saved	norway, denmark, sweden
14	red ∧ age ∧ young → saved	norway, denmark, sweden
15	red ∧ fitness ∧ fat → not_saved	norway, denmark, sweden
16	red ∧ fitness ∧ fit → not_saved	norway, denmark, sweden
17	red ∧ gender ∧ female → not_saved	norway, denmark, sweden
18	red ∧ gender ∧ male → not_saved	norway, denmark, sweden
19	red ∧ socialStatus ∧ high → not_saved	norway, denmark, sweden
20	red ∧ socialStatus ∧ low → not_saved	norway, denmark, sweden
21	red ∧ species ∧ human → saved	norway, denmark, sweden
22	red ∧ species ∧ pets → not_saved	norway, denmark, sweden
23	red ∧ utilitarian ∧ less → not_saved	norway, denmark, sweden
24	red ∧ utilitarian ∧ more → saved	norway, denmark, sweden

Table 5.6: Horn Expression for Scandinavian Countries

Indeed, there is no significant number of conflicts or incoherence in the Horn expressions associated with these countries. There is only one conflict case. Norway and Denmark prefer to save people with low social status when the traffic light is green, while Sweden penalizes people with low social status. In contrast with the United States, where people with high social status are saved even if they cross on the red signal. All the Scandinavian countries penalizes them for such action.

All countries are from the same cluster. Therefore, conflict should be resolved by specific symbols (country name) not by generalized symbol (cluster name). Because rules with generalized symbols would still be in conflict. Table 5.7 depicts rules which are not in conflict, non-redundant, and acyclic but incoherent. This is the case when Algorithm 1 can find a common ground.

8a	green ∧ socialStatus ∧ low ∧ **norway** → **saved**
8b	green ∧ socialStatus ∧ low → saved
8c	green ∧ socialStatus ∧ low ∧ **sweden** → **not_saved**

Table 5.7: Horn Expression for the Scandinavian Countries after resolving Conflicts

Now we have incoherence for 8b and 8c that would be resolved by algorithm. Common ground would be the following rules. All other rules that had no conflict, would remain the same in common ground.

1	green ∧ age ∧ old → not_saved
2	green ∧ age ∧ young → saved
3	green ∧ fitness ∧ fat → saved
4	green ∧ fitness ∧ fit → saved
5	green ∧ gender ∧ female → saved
6	green ∧ gender ∧ male → saved
7	green ∧ socialStatus ∧ high → saved
8a	green ∧ socialStatus ∧ low ∧ **norway** → **saved**
8b	green ∧ socialStatus ∧ low ∧ **not_sweden** → **saved**
8c	green ∧ socialStatus ∧ low ∧ **sweden** → **not_saved**
9	green ∧ species ∧ human → saved
10	green ∧ species ∧ pets → not_saved
11	green ∧ utilitarian ∧ less → not_saved
12	green ∧ utilitarian ∧ more → saved
13	red ∧ age ∧ old → not_saved
14	red ∧ age ∧ young → saved
15	red ∧ fitness ∧ fat → not_saved
16	red ∧ fitness ∧ fit → not_saved
17	red ∧ gender ∧ female → not_saved
18	red ∧ gender ∧ male → not_saved
19	red ∧ socialStatus ∧ high → not_saved
20	red ∧ socialStatus ∧ low → not_saved
21	red ∧ species ∧ human → saved
22	red ∧ species ∧ pets → not_saved
23	red ∧ utilitarian ∧ less → not_saved
24	red ∧ utilitarian ∧ more → saved

Table 5.8: Common Ground for the Scandinavian Countries

5.5.3 Comparison of Clusters

In our last experiment, countries from three different clusters are selected. These clusters divided by the authors of the Moral Machine Experiment. The following example takes India from the eastern cluster, France from the southern cluster, and Norway from the western cluster. Above selected three countries has the following Horn expression:

1	green ∧ age ∧ old → not_saved	norway, india, france
2	green ∧ age ∧ young → saved	norway, india, france
3	green ∧ fitness ∧ fat → saved	norway, india, france
4	green ∧ fitness ∧ fit → saved	norway, india, france
5	green ∧ gender ∧ female → saved	norway, india, france
6	green ∧ gender ∧ male → saved	norway
	green ∧ gender ∧ male → not_saved	india
	green ∧ gender ∧ male → saved	france
7	green ∧ socialStatus ∧ high → saved	norway, india, france
8	green ∧ socialStatus ∧ low → saved	norway, india, france
9	green ∧ species ∧ human → saved	norway, india, france
10	green ∧ species ∧ pets → not_saved	norway, india, france
11	green ∧ utilitarian ∧ less → not_saved	norway, india, france
12	green ∧ utilitarian ∧ more → saved	norway, india, france
13	red ∧ age ∧ old → not_saved	norway, india, france
14	red ∧ age ∧ young → saved	norway
	red ∧ age ∧ young → not_saved	india
	red ∧ age ∧ young → saved	france
15	red ∧ fitness ∧ fat → not_saved	norway, india, france
16	red ∧ fitness ∧ fit → not_saved	norway, india, france
17	red ∧ gender ∧ female → not_saved	norway, india, france
18	red ∧ gender ∧ male → not_saved	norway, india, france
19	red ∧ socialStatus ∧ high → not_saved	norway
	red ∧ socialStatus ∧ high → not_saved	india
	red ∧ socialStatus ∧ high → saved	france
20	red ∧ socialStatus ∧ low → not_saved	norway, india, france
21	red ∧ species ∧ human → saved	norway, india, france
22	red ∧ species ∧ pets → not_saved	norway, india, france
23	red ∧ utilitarian ∧ less → not_saved	norway, india, france
24	red ∧ utilitarian ∧ more → saved	norway, india, france

Table 5.9: Horn Expression for Norway, France and India

These countries have conflicts that are highlighted on rules 6, 14 and 19 in Table 5.9. Basically, Norway and France both have conflicts with India. In particular on line

14, India (and also China), does not prefer to save young people on red signal (crossing illegally) while Norway and France do. There are two conflicts in this case.

On line 6, India penalizes males while France and Norway make no gender distinction when the pedestrian is crossing on the green signal (crossing legally). All the three countries penalize pedestrians of both genders when they cross on the red signal.

The last scenario is on line 19, where these countries deviate. This is the one where pedestrians with high social status cross on the red signal. Like the United States, France saves them, while both India and Norway penalizes pedestrians. In total, there are six conflicts in this Horn expression. The use of algorithm which resolves conflicts and modifies these rules as following, while all other rules come out as these are:

6a	green ∧ gender ∧ male ∧ **west** → **saved**
6b	green ∧ gender ∧ male ∧ **east** → **not_saved**
6c	green ∧ gender ∧ male → saved
14a	red ∧ age ∧ young ∧ **west** → **saved**
14b	red ∧ age ∧ young ∧ **east** → **not_saved**
14c	red ∧ age ∧ young → saved
19a	red ∧ socialStatus ∧ high ∧ **west** → **not_saved**
19b	red ∧ socialStatus ∧ high ∧ **east** → **not_saved**
19c	red ∧ socialStatus ∧ high → saved

Table 5.10: Horn Expression for Norway, France and India after resolving Conflicts

Incoherence and conflicts exist in the same part after resolution of conflicts.

This is showed in table 5.11 where one incoherence is of 6b with 6c rule one. Similarly, for 14b and 14c. For rule 19, there are two incoherences. If any of the incoherence is solved either 19a with 19c or 19b with 19c, then a common ground is established for these three sets of rules.

6a	green ∧ gender ∧ male ∧ **west** → **saved**
6b	green ∧ gender ∧ male ∧ **east** → **not_saved**
6c	green ∧ gender ∧ male ∧ **not_east** → **saved**
14a	red ∧ age ∧ young ∧ **west** → **saved**
14b	red ∧ age ∧ young ∧ **east** → **not_saved**
14c	red ∧ age ∧ young ∧ **not_east** → **saved**
19a	red ∧ socialStatus ∧ high ∧ **west** → **not_saved**
19b	red ∧ socialStatus ∧ high ∧ **east** → **not_saved**
19c	red ∧ socialStatus ∧ high ∧ **not_east** → **saved**

Table 5.11: Common Ground for Norway, France and India

For 19c, incoherence could be resolved by either 19a or 19b, so possible antecedents could be either not_east or not_west.

In summary, this case study confirmed the expectation. It states that countries with more cultural similarities would have more similar moral recommendations. The attribute that caused significant divergences was the social status, followed by age, and then gender.

Table 5.12 presents the number of rules weakened in our experiments.

Countries	**Conflicts**	**Incoherences**
US, China	3	3
Scandinavia	2	1
Clusters	6	3

Table 5.12: Summary of the Study Case

Chapter 6

Related Work

Related work pertaining to conflict resolution among rules has been carried in many different fields. Unlike our work, these fields may not be particularly concerned about the machine ethics contexts. These typically attempt to solve the conflict problem in bigger picture rather than finding practically implementable options, which we propose. Most related work has been highlighted below.

Formal legal reasoning

Resolving of legal conflicts between rules expressed in logic is studied for example in Ju et al. [24]. The legal approach to solving conflicts is to use a hierarchy of more specific rules overriding more general ones, more recent rules overriding older rules and establishing a hierarchy among law issuing institutions. Here we adopt that more specific rules override more general ones. Horty [25, 26] considers a logic framework for formal understanding of common law and discusses when and how rules (which in his framework are also of the form antecedent, conclusion) can be changed in the common law. Such change, in common law, must satisfy two so called Raz/Simpsons conditions: (1) rule modifications can consist of the addition of further restrictions and (2) the modified rule must support the same conclusion as the original rule. Our algorithm supports the Condition 1 by design. A weaker form of Condition 2 is guaranteed by P3.

A logic-based approach to conflict resolution

In the paper [27], Robert Kowalski presents an approach to conflict resolution that unifies logic, goal-reduction and condition-action rules in a cognitive model of the intelligent agent. He investigated two applications to illustrate and test the power and generality of his approach. These applications are known as the prisoner's dilemma and the Agha-Malley proposed solution of the Israeli-Palestinian conflict.

He states simple steps to reconcile a conflict between two inconsistent goals [27]. These steps are:

- Find higher-level goals which do not conflict.
- Find alternative, consistent ways of satisfying higher-level goals.
- Infer positive and negative consequences of alternatives.
- Estimate the utilities of consequences.
- Estimate the probabilities of any unknown and uncertain conditions in the alternatives.
- Choose alternatives which come as close as possible to maximizing the expected utility of the consequences.

This methodology of conflict resolution—by identifying alternative ways of satisfying higher level goals – is like one proposed by van Lamsweerde et al. (1998). Later one use to manage conflicts in the requirements engineering stage of software development.

Multi-agent systems

Norms can be viewed as a powerful means to regulate and influence the behavior of the agents. These are developed by specifying obligations, permissions, or prohibitions in each context. In Vasconcelos et al. [28], numerous techniques are presented that have been proposed to detect and resolve normative conflicts in multi-agent systems. A set of these are specified to use at design time, while others at runtime. In addition, some of these can solve only direct normative conflicts, while others are able to solve indirect conflicts. The strategies used by the analyzed proposals. These can be divided into two kinds: norm prioritization and norm update. In "norm prioritization", one norm overrides another in specific circumstances. "Norm update" results in one of the norms in conflict to be updated.

Institutional facts The work described in [29, 30] deals with normative conflicts between independent institutions that are under the same governance scope. It can resolve direct normative conflicts at design time. Conflicts may occur when an action is permitted in one institution but not in the other at the same time. The strategy adopted to resolve conflicts is based on inductive learning. The method consists of revising a logic program that represents a formal model, containing the rules of a specific normative system. In this approach, a precedence order is established among the institutions. The revision method is applied to the norms of the less important institution of the conflicting pair of institutions. The approach is very similar to *lex superior*. However, in this work the organization with lower precedence is not overridden but changed to be consistent with the organization with higher precedence.

Deontic logic approach The work described in [31] is able to resolve direct and indirect normative conflicts at design time among norms associated with different roles. When there is a normative conflict between two roles, and it is possible that an agent plays these roles simultaneously. Here requires a judgment of priorities between these. It helps to decide which norms to adopt in a given situation. It determines a total order between the roles. For instance, if there is a conflict between roles r_1 and r_2 and the judgment of priorities states that $r_1 > r_2$. In such cases all norms associated with r_1 take precedence over the norms associated with r_2. The judgment of priorities may depend on the individual (especially in moral dilemmas) or may be derived from the hierarchy of roles (sub-roles, sub-ideal-roles). The authors also consider an action is not obligatory if there is no norm that explicitly obligates it.

Normative conflict graph The work presented in [32] is able to resolve direct and indirect normative conflicts at runtime. This research presents a model of normative agents. These agents have a set of norms and a set with a partial ordering over the social contexts. This set determines the importance of complying with norms imposed by different social contexts. Therefore, a norm can be prioritized. The authors argue that when two norms are in conflict, the agent should determine which norm to violate. Its implementation should ensure maximum compliance with the remaining set of norms. To resolve conflicts, agents may prune some edges of the normative conflict graph based on their social context preferences. Agents can also perform the norm pruning considering the norm modality (obligation, permission, or prohibition). For instance, an agent can prefer to comply with obligations rather than prohibitions. Similarly, a normative agent can also have a set with a partial ordering over norm modalities.

Owl-Polar The work described in [33] can resolve direct and indirect normative conflicts at design time. The strategies to solve conflicts use a policy refinement method. It consists of determining an order of norm overruling after determining norm precedence based on standard techniques. The work in [34] states the strategies, such as *lex superior*, *lex posterior* and *lex specialis* may be adopted to resolve conflicts among OWL-POLAR policies. The *lex posterior* strategy can only be adopted when the conflicting norms have been issued by the same authority. The reason is the temporal relationships between different authorities which may be misleading. The *lex specialis* can only be applied when a more specific norm can be considered an exception to other, and the conflicting norms belong to the same organization. The authors present a subsumption reasoning algorithm that verifies if one norm is a specialization of other.

Chapter 7

Conclusion

We discussed the original version of common ground algorithm. It was used by the authors to prove that for non-redundant, acyclic, and not in conflict Horn expressions, a common ground is guaranteed to exist. We improved the algorithm by reducing its common ground's limitation. If there is a cycle or redundancy in the Horn expression, a practical way to deal with the limitations is by taking away the clauses that are causing it. It makes the expression acyclic and non-redundant. The original version of algorithm can resolve incoherence but unable to handle with conflicts. We also proposed a strategy to resolve the conflict if it exists in the Horn expression. For conflict resolution, we introduced constraints like collection of symbols $\mathcal{C}$ associated with clauses in the Horn expression and another background knowledge $\overline{\mathcal{B}}$ besides the background knowledge $\mathcal{B}$ already mentioned in the original version of the algorithm. Both background knowledge contains basic constraints about the world. The main difference is that $\mathcal{B}$ contains horn constraints while $\overline{\mathcal{B}}$ contains definite horn clauses with single antecedent, from the collection of symbols $\mathcal{C}$. We resolve the conflict by making rules weakened, as we did for resolving incoherence. For resolution of conflict, we proposed weakened of clauses by using background knowledge $\overline{\mathcal{B}}$ and symbols $\mathcal{C}$, while incoherence was resolved by using $\mathcal{B}$.

We also implemented the improved version of common ground algorithm in Python programming language. An explanation of the design, adopted for implementation of each feature, is also provided. We explored the phenomena of Moral Machine experiment and analyzed its dataset. During analysis, we performed preprocessing of the dataset by putting defined conditions in order, to prepare it for the algorithm. We generated rules from the dataset. These were used as a Horn expression. We also produced both background knowledge and collection of symbols from the dataset. As each rule in the dataset is expressing a decision from a candidate of Moral Machine experiment. We used countries' names as specific symbols for each row in the dataset and a distinct country name is associated with it.

We conducted an experiment by testing the algorithm on our prepared dataset of the Moral Machine experiment. Three different combinations of countries were used and found common ground. First analyzed the case of two largest economies in the world: United States and China. Three different conflicts were found based on social status and age. Conflict resolution leads to three incoherence. There were resolved and common ground was established by the algorithm. In the second experiment, a group of Scandinavian countries are selected. Similar moral recommendations were expected. Conflict was only found in the scenario of social status. For the third experiment, countries from three different clusters were considered. These clusters are divided by the authors of the Moral Machine experiment. In the third experiment, conflicts were found in scenarios of age and social status for both green and red traffic signal. In summary, our case study confirmed the expectation. Countries with more cultural similarities would have more similar moral recommendations.

Our experiments show that how the ideas that are developed can be used to improve the algorithm on a concrete dataset. As future work, it would be interesting to use the algorithm for other datasets. As our proposed improvements of algorithm has limitations. One of the limitations is to remove cycles from the Horn expression. We removed a clause of a cycle. In this way there is a possibility that we can lose some important agent's recommendation that should have been part of the expression as compared to any other clause of expression. It would be practical to set preferences on removal of clauses. It optimizes our implementation even though the acyclicity requirement can be seen as not a very strong condition. It would also be interesting to explore how preferences can be set over the conditions of a common ground, or to avoid any of the known conditions. If any of the three conditions (acyclicity, non-redundancy, no conflict) is removed then there is no algorithm that is guaranteed to produce a common ground. There are certain challenges in approaches to conflict resolution. Like, it can only deal with datasets where symbols associated with rules as we had countries names as symbols in Moral Machine case. It would be interesting to investigate how to deal with the datasets that do not have corresponding symbols while having conflicts.

Bibliography

[1] Dignum, V. *Responsible Artificial Intelligence*; Springer International Publishing, 2019.

[2] Bjørgen, E. P.; Madsen, S.; Bjørknes, T.; Heimsæter, F. V.; Håvik, R.; Linderud, M.; Longberg, P.; Dennis, L. A.; Slavkovik, M. Cake, Death, and Trolleys: Dilemmas as benchmarks of ethical decision-making. Conference on AI, Ethics, and Society, AIES. 2018; pp 23–29.

[3] Bremner, P.; Dennis, L. A.; Fisher, M.; Winfield, A. F. On Proactive, Transparent, and Verifiable Ethical Reasoning for Robots. *Proceedings of the IEEE* **2019**, *107*, 541–561.

[4] Schwarting, W.; Alonso-Mora, J.; Rus, D. Planning and Decision-Making for Autonomous Vehicles. *Annual Review of Control, Robotics, and Autonomous Systems* **2018**, *1*, 187–210.

[5] Awad, E.; Dsouza, S.; Kim, R.; Schulz, J.; Henrich, J.; Shariff, A.; Bonnefon, J.; Rahwan, I. The Moral Machine Experiment. *Nature* **2018**, *563*.

[6] Ozaki, A.; Rehman, A.; Turk, P.; Slavkovik, M. Finding Common Ground for Incoherent Horn Expressions. 2022; `https://arxiv.org/abs/2209.06455`.

[7] Genesereth, M. Logical Entailment. `https://www.slideserve.com/napoleon/logical-entailment`, 2014.

[8] Russell, S.; Norvig, P. *Artificial Intelligence: A Modern Approach*, 3rd ed.; Prentice Hall, 2010.

[9] Poole, D.; Mackworth, A. *Artificial Intelligence: Foundations of Computational Agents*, 2nd ed.; Cambridge University Press: Cambridge, UK, 2017.

[10] Mayes, G. R. Logical Consistency and Contradiction. `https://www.csus.edu/indiv/m/mayesgr/phl4/handouts/phl4contradiction.htm`.

[11] Baldwin, E. Coherence. https://poemanalysis.com/literary-device/coherence/</a>, 2022.

[12] Dennen, J. Introduction: On Conflict. *The Sociobiology of Conflict. London: Chapman Hall, 1990, pp. 1- 19.* **2005**, Relation: http://www.rug.nl/Rechten/ date_submitted:2000 Rights: University of Groningen. Faculty of Law.

[13] Hare, R. M. *Applications of Moral Philosophy*; Macmillan Education UK: London, 1972; pp 109–115.

[14] Patterson, A. examples-of-redundancy. https://www.writerswrite.co.za/19-examples-of-redundancy/, 2014.

[15] Dependency Graph. https://deepsource.io/glossary/dependency-graph/, Accessed: 2022-10-26.

[16] Source code for sympy.logic.inference. http://man.hubwiz.com/docset/SymPy.docset/Contents/Resources/Documents/_modules/sympy/logic/inference.html#satisfiable, Last updated on Apr 10, 2019.

[17] NetworkX-Developers. https://networkx.org/documentation/stable/reference/classes/digraph.html.

[18] Acyclic directed Graphs. Copyright 2014, NetworkX Developers. Last updated on Jun 21, 2014, Created using Sphinx 5.3.0.

[19] find graph cycles. https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.cycles.find_cycle.html, Copyright 2004-2022, NetworkX Developers. Created using Sphinx 5.3.0.

[20] Networkx graphs predecessors. https://networkx.org/documentation/stable/reference/classes/generated/networkx.DiGraph.predecessors.html, Copyright 2004-2022, NetworkX Developers. Created using Sphinx 5.3.0.

[21] Lassen, I. M. S. The Amorality of the Moral Machine. https://dataethics.eu/the-amoral-of-the-moral-machine/, 2019.

[22] Krishna, S. The moral machine: Who lives, who dies, you decide! https://analyticsindiamag.com/the-moral-machine-who-lives-who-dies-you-decide/, 2022.

[23] Hao, K. Should a self-driving car kill the baby or the grandma? Depends on where you're from. ://www.technologyreview.com/2018/10/24/139313/a-global-ethics-study-aims-to-help-ai-solve-the-self-driving-trolley-problem/, October 24, 2018.

[24] Ju, F.; Nygren, K.; Xu, T. Modeling legal conflict resolution based on dynamic logic. *Journal of Logic and Computation* **2020**, *31*, 1102–1128.

[25] Horty, J. Constraint and Freedom in the Common Law. *Philosophers Imprint* **2015**, *15*, 1–27.

[26] Horty, J. *The Logic of Precedent: Constraint and Freedom in Common Law Reasoning*; Cambridge University Press: USA, 2023; Forthcoming, access at http://www.horty.umiacs.io/articles/2022-7-15-logic-precedent.pdf.

[27] Kowalski, R. A Logic-Based Approach to Conflict Resolution. **2003**,

[28] Vasconcelos, W. W.; Kollingbaum, M. J.; Norman, T. J. Normative conflict resolution in multi-agent systems. *Autonomous Agents and Multi-Agent Systems* **2009**, *19*, 124–152.

[29] Li, T. Normative Conflict Detection and Resolution in Cooperating Institutions. **2014**,

[30] Li, T. Normative Conflict Detection and Resolution in Cooperating Institutions. Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence. 2013; p 3231–3232.

[31] Cholvy, L.; Cuppens, F. Solving Normative Conflicts by Merging Roles. Proceedings of the 5th International Conference on Artificial Intelligence and Law. New York, NY, USA, 1995; p 201–209.

[32] Oren, N.; Luck, M.; Miles, S.; Norman, T. An Argumentation Inspired Heuristic for Resolving Normative Conflict. 2008.

[33] Aphale, M. S.; Norman, T. J.; Şensoy, M. Goal-Directed Policy Conflict Detection and Prioritisation. Coordination, Organizations, Institutions, and Norms in Agent Systems VIII. Berlin, Heidelberg, 2013; pp 87–104.

[34] Sensoy, M.; Norman, T.; Vasconcelos, W.; Sycara, K. OWL-POLAR: A Framework for Semantic Policy Representation and Reasoning. *Web semantics: science, services and agents on the World Wide Web* **2012**, *12-13*, 148–160.

Appendix A

Generated code from Protocol buffers

```python
def extract_ant_con(clause):

    if type(clause.args[0]) == And:
        antacedent = set(clause.args[0].args)
    else:
        antacedent = {clause.args[0]}

    consequent = clause.args[1].atoms()

    return antacedent, consequent
```

Listing A.1: Extract antecedents and consequents of clause

```python
def extract_symbols(F):
    atoms= []
    for clause in F:
        atoms.append(list(clause.args[0].atoms()))

    flat_list = [item for sublist in atoms for item in sublist]
    symbol_list = list(set(flat_list))

    return symbol_list
```

Listing A.2: Extract Symbols from Horn expression

```python
def get_excludents(background_know, atom):

    excludent_atoms = []

    symbol_background_know = extract_symbols(background_know)

    for symbol in symbol_background_know:
        clause = Implies((symbol & atom), False)

        if clause in background_know:
            excludent_atoms.append(symbol)

    return excludent_atoms
```

Listing A.3: Excludents of an atom

```python
def clauses_derivation(clause1, clause2, F):

    count = 0
    ant_clause1 = extract_ant_con(clause1)[0]
    ant_clause2 = extract_ant_con(clause2)[0]
    copy_expr = copy.deepcopy(F)
    copy_expr.extend(ant_clause1)

    for atom in ant_clause2:
        if entails(atom, copy_expr):
            count +=1

    if count == len(ant_clause2):
        return True
    else:
        return False
```

Listing A.4: Structure for *derivation* of two clauses

```python
def non_redundant_expr(F):

    count = 0
    for clause in F:

        #hornexpression without that clause
        expr_without_clause = [x for x in F if x != clause]

        if not entails(clause, expr_without_clause):
            count +=1

    if count == len(F):
        print('\n-------------This expression is Non-Redundant-------------\n')
        return True
    else:
        return False
```

Listing A.5: Structure for *Non-redundancy* of a Horn Expression

```python
def check_conflict_clauses(F, clause1, clause2, background_know):

    ant_clause1 = extract_ant_con(clause1)[0]
    ant_clause2 = extract_ant_con(clause2)[0]

    #remove ant of clause 1 from ant of clause 2
    ant_diff = [x for x in ant_clause1 if x not in ant_clause2]

    if len(ant_diff) != 0:
        for atom in ant_diff:
            excludent_atom = get_excludents(background_know, atom)

            for i in excludent_atom:
                add_ant_to_clause = Implies((clause2.args[0] & i), clause2.args[1])
                F_without_clause = [x for x in F if x != clause2]

                if check_clause_coherence(F_without_clause, add_ant_to_clause):
                    return False
    else:
        return True
```

Listing A.6: Structure for *conflict* in a pair of clauses

```python
def get_safe_pairs(F):
    G = nx.DiGraph()
    clauses = list(itertools.combinations(F, 2))
    inverted_clauses = [t[::-1] for t in clauses]

    indices = list((i+1,j+1) for ((i,_),(j,_)) in
    ↪  itertools.combinations(enumerate(F), 2))

    selected_clauses = []
    options = {"edgecolors": "tab:gray", "node_size": 2550, "alpha": 0.9,"font_size":
    ↪  13, }

    for i in range(len(clauses)):
        if clauses[i][0].args[1] in get_excludents(background_know,
        ↪  clauses[i][1].args[1]) and
        ↪  (clauses_derivation(clauses[i][0],clauses[i][1],F) or
        ↪  clauses_derivation(clauses[i][1],clauses[i][0],F)):
            selected_clauses.append(clauses[i])
            G.add_node((indices[i][1], indices[i][0]))

    inverted_clauses = [t[::-1] for t in selected_clauses]
    s_clauses = generate_combinations(inverted_clauses)

    for clause in s_clauses:
        if next(iter(extract_ant_con(clause[0][1])[1])) in
        ↪  extract_ant_con(clause[1][1])[0] and
        ↪  bool(extract_ant_con(clause[0][0])[0] &
        ↪  extract_ant_con(clause[1][0])[0]):

            G.add_edge((F.index(clause[0][0])+1, F.index(clause[0][1])+1),
            (F.index(clause[1][0])+1, F.index(clause[1][1])+1))

        elif next(iter(extract_ant_con(clause[1][1])[1])) in
        ↪  extract_ant_con(clause[0][1])[0] and
        ↪  bool(extract_ant_con(clause[0][0])[0] &
        ↪  extract_ant_con(clause[1][0])[0]):

            G.add_edge((F.index(clause[1][0])+1, F.index(clause[1][1])+1),
            (F.index(clause[0][0])+1, F.index(clause[0][1])+1))

    pos = nx.shell_layout(G)
    nx.draw(G, pos, with_labels = True, **options,
    ↪  node_color="tab:pink",arrowsize=20, edge_color = 'green')
    plt.axis("off")
    plt.show()
    non_safe_pairs = []
    safe_pairs = []
    for node in G:
        if not list(G.predecessors(node)):
            safe_pairs.append((F[node[0]-1], F[node[1]-1]))
            print('Safe pair is: ',node)
        else:
            non_safe_pairs.append(node)

    if len(non_safe_pairs) == G.number_of_nodes():
        print('There is no safe clause')
        return False
    else:
        return safe_pairs
```

Listing A.7: Structure for *Safe pairs* in Horn expression

```python
def weaker_version_clause(F, clause1, clause2):

    ant_clause1 = extract_ant_con(clause1)[0]
    ant_clause2 = extract_ant_con(clause2)[0]

    ant_diff = [x for x in ant_clause1 if x not in ant_clause2]
    if len(ant_diff) == 0:
        return weaker_version_clause(F, clause2, clause1)
    else:
        excludent_atom = get_excludents(background_know, ant_diff[0])
        for i in excludent_atom:
            weaker_clause = Implies(clause2.args[0] & i, clause2.args[1])
            F_without_clause = [x for x in F if x != clause2]

            if check_clause_coherence(F_without_clause, weaker_clause):
                indices = [i for i, x in enumerate(F) if x == clause2]
                return weaker_clause, indices
            else:
                return False
```

Listing A.8: Structure for *Weaker version* of clause

```python
def get_filtered_data(df, countries_list):

    dataset = df[(df.UserCountry3.isin(countries_list))]

    dataset = dataset[dataset.PedPed == 1]
    dataset = dataset[(dataset.CrossingSignal == 1) | (dataset.CrossingSignal == 2)]

    dataset = dataset.groupby(['CrossingSignal','ScenarioType','AttributeLevel',
    'UserCountry3','Saved']).size().reset_index(name='Count')

    dataset = dataset[dataset.duplicated(subset=['ScenarioType','AttributeLevel'],
    ↪  keep=False)]

    drop =  []
    for i, g in dataset.groupby(dataset.index // 2):
        count = g['Count'].tolist()
        drop.append(g.index[g['Count'] == min(count)].tolist())

    drop_list = [item for sublist in drop for item in sublist]
    filtered_data = dataset.drop(index=drop_list)

    return filtered_data
```

Listing A.9: Structure for filtering data of *Moral Machine dataset*

```python
def get_clauses(data):

    clauses = []
    symbols_col = []

    for index, row in data.iterrows():
        if row["CrossingSignal"] == 1:
            row["CrossingSignal"] = symbols('green')
        else:
            row["CrossingSignal"] = symbols('red')
        row["ScenarioType"] = symbols(row["ScenarioType"])
        row["AttributeLevel"] = symbols(row["AttributeLevel"])
        row["UserCountry3"] = symbols(row["UserCountry3"])
        if row["Saved"] == 1:
            row["Saved"] = symbols('saved')
        else:
            row["Saved"] = symbols('not_saved')
        clauses.append(Implies((row["CrossingSignal"] & row["ScenarioType"] & row["AttributeLevel"]), (row["Saved"])))

        symbols_col.append(row["UserCountry3"])

    return clauses, symbols_col
```

Listing A.10: Structure for extracting rules from *Moral Machine* dataset