# Threat Modelling Technique for GDPR Compliance based on Logical Reasoning

Naila

Submitted in fulfilment of the requirements for the Degree of Doctor of Philosophy

School of Computing Science
College of Science and Engineering
University of Glasgow

July 2024

# Abstract

Data-driven applications and services are increasingly being deployed across various sectors, where they collect, aggregate, and process vast amounts of personal data from diverse sources on centralized servers. Consequently, safeguarding the privacy and security of this data is crucial. Since May 2018, the EU/UK's General Data Protection Regulation (GDPR) has necessitated sophisticated compliance models. Current threat modeling techniques, however, do not adequately address GDPR compliance, particularly in complex systems where personal data is collected, processed, manipulated, and shared with third parties. This thesis proposes a comprehensive solution to develop a threat modeling technique that addresses and mitigates non-compliance threats by integrating GDPR requirements with existing security and privacy modeling techniques, namely STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege) and LINDDUN (Linking, Identifying, Non-repudiation, Detecting, Data Disclosure, Unawareness, and Non-compliance). The proposed technique in this thesis introduces a new data flow diagram aligned with GDPR principles, develops a knowledge base for non-compliance threats, and employs an inference engine to reason about these threats using the developed knowledge base. Additionally, this thesis presents a practical solution for modeling GDPR compliance using Defeasible Logic Programming (DeLP), enhancing the robustness and reasoning capabilities of compliance models in real-world scenarios. To address the challenges of undecided outputs in logical reasoning, this work incorporates explicit priorities for conflicting rules and suggests related knowledge for queries in an incomplete knowledge base. Furthermore, the technique includes a threat mitigation mechanism that identifies reasons for non-compliance threats and recommends actions to mitigate them. This approach is demonstrated through case studies on *Telehealth* Services and *Fitbit* (i.e., health tracking devices), focusing on addressing non-compliance threats and resolving UNDECIDED query results. Finally, the complexity of the defeasible reasoning mechanism is analyzed, and its performance is compared across different query outcomes, namely "YES/NO/UNDECIDED," based on vertical and horizontal complexities. The findings indicate that DeLP offers a flexible and dynamic framework suitable for implementing GDPR in real-world settings, making a significant contribution to the fields of legal reasoning and compliance modeling. Additionally, our findings show that the inference engine efficiently identifies non-compliance threats, handles UNDECIDED query results, and suggests appropriate threat mitigation measures.

# Contents

# List of Tables

# List of Figures

# List of Abbreviations

| | |
|---|---|
| **GDPR** | General Data Protection Regulation |
| **STRIDE** | Spoofing Tampering Repudiation Information Disclosure Denial of Service Elevation of Privilege |
| **LINDDUN** | Linkability, Identifiability, Non-Repudiation, Detectability, Disclosure of information, Unawareness, and Non-Compliance. Linkability |
| **PASTA** | Process for Attack Simulation and Threat Analysis |
| **hTMM** | Hybrid Threat modelling Method |
| **PnG** | Persona Non-Grata |
| **QTMM** | Common Vulnerability Scoring System |
| **CVSS** | Audio-Visual |
| **VAST** | Visual, Agile, and Simple Threat |
| **OCTAVE** | Operationally Critical Threat, Asset, and Vul- nerability Evaluation |
| **SDLC** | Software Development Life Cycle |
| **DC** | Data Controller |
| **DS** | Data Subject |
| **DP** | Data Processor |
| **ACS** | Autonomous Car Systems |
| **IOIs** | items of interest |
| **AC** | Autonomous Car |
| **RSU** | Road Side Unit |
| **TA** | Trusted Authority |

| | |
|---|---|
| **CAN** | Connected and Autonomous Network |
| **V2X** | Vehicle-to-everything |
| **V2V** | Vehicle to Vehicle |
| **DDoS** | distributed denial-of-service |
| **V2X** | Vehicle-to-everything |
| **LTE** | long-term evolution |
| **WiFi** | wireless fidelity |
| **GPS** | Global Positioning System |
| **PEC** | Privacy Enhancing Techniques |
| **DL** | Defeasible Logic |
| **DeLP** | Defeasible Logic Programming |
| **SWRL** | Semantic Web Rule Language |
| **WOL** | Web Ontology Language |
| **RuleML** | Rule Markup Language |
| **MSTMT** | Microsoft Threat Modelling Tool |
| **TSS** | Telehealth Services System |
| **SA** | Supervisory Authority |
| **GDS** | Generic Data Store |
| **OTS** | Organ Transplant Service |

# List of Publications

During my PhD. tenure, I have been involved in 5 publications, all as the first author (5 papers) which are all directly related to my thesis topic. Below is a list of all my publications in this period.

A. *Journals*

(1) **Azam, N.**, Michala, L., Ansari, S., Truong, N. B. (2022). Data Privacy Threat Modelling for Autonomous Systems: A Survey from the GDPR's Perspective. IEEE Transactions on Big Data..

(2) **Azam, N.**, Lito Michala, A., Ansari, S., & Truong, N. B. (2024). A Defeasible Logic-based Modelling Solution for GDPR Compliance. IEEE Transactions on Dependable and Secure Computing, January 2024 (under review)

(3) **Azam, N.**, Chak, A., Lito Michala, A., Ansari, S., & Truong, N. B. (2024). A Practical Solution for Modelling GDPR Compliance based on Defeasible Logic, Expert Systems with Applications (under review).

B. *Conference Proceedings*

(1) **Azam, N.**, Michala, L., Ansari, S., & Truong, N. B. Modelling Technique for GDPR-compliance: Toward a Comprehensive Solution. IEEE Global Communications Conference (GLOBECOM 2023).

(2) **Azam, N.**, Chak, A., Michala, L., Ansari, S., & Truong, N. B. Modelling GDPR-compliance based on Defeasible Logic Reasoning: Insights from Complexity Perspective. International Workshop on AI-Driven Trust, Security and Privacy in Computer Networks (AI-Driven TSP 2024) (Accepted for publication).

# Acknowledgements

Throughout the course of my doctoral research, I was honored to benefit from the invaluable support and mentorship of numerous esteemed individuals, to whom I express my deepest gratitude.

I would like to seize this moment to extend my heartfelt gratitude to my esteemed academic supervisor, Dr. Nguyen Truong, for his unwavering support and invaluable guidance throughout my PhD journey. His profound insights and thoughtful mentorship have shaped my academic pursuits. His dedication and commitment to excellence have profoundly impacted my growth and development, both as a scholar and an individual.

I would like to thank Dr. Shuja Ansari and Dr. Lito Michala for their continued support and invaluable input throughout my PhD.

I am also extremely thankful to the University of Glasgow for supporting me in my hardship time.

Last but certainly not least, my heartfelt thanks go to my parents and my husband for their unwavering support and encouragement throughout my PhD journey. Their belief in my endeavors has been a constant source of motivation.

# Declaration

Appendix 2.4

**University of Glasgow**
***College Identity***

**Statement of Originality to Accompany Thesis Submission**

**Name:** Naila

**Registration Number:** xxxxxxxx.

I certify that the thesis presented here for examination for [a/an MPhil/PhD] degree of the University of Glasgow is solely my own work other than where I have clearly indicated that it is the work of others (in which case the extent of any work carried out jointly by me and any other person is clearly identified in it) and that the thesis has not been edited by a third party beyond what is permitted by the University's PGR Code of Practice.

The copyright of this thesis rests with the author. No quotation from it is permitted without full acknowledgement.

I declare that the thesis does not include work forming part of a thesis presented successfully for another degree [unless explicitly identified and as noted below].

I declare that this thesis has been produced in accordance with the University of Glasgow's Code of Good Practice in Research.

I acknowledge that if any issues are raised regarding good research practice based on review of the thesis, the examination may be postponed pending the outcome of any investigation of the issues.

Signature: ______

Date: 30/07/2024.

**This completed statement must be bound into the submitted copies of the soft-bound thesis.**

# Statement of Copyright

# Chapter 1

# Introduction

## 1.1 Background and Motivation

In recent years, data-driven applications are increasingly being deployed in all aspects of life including smart homes, smart cities, healthcare, and medical services [1]. In such applications, Artificial Intelligence (AI) incorporating various algorithms is employed, where personal data is collected and aggregated from heterogeneous sources before being processed using "black-box" algorithms in opaque centralised servers [2, 3, 4, 5]. As a consequence, preserving the data privacy and security of these applications is of paramount importance [6]. In this respect, a modelling technique for detecting potential threats and specifying countermeasures to mitigate the vulnerabilities plays a significant role in securing personal data from a variety of data breaches and privacy attacks.

Numerous threat modelling techniques have been proposed in the literature such as STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege), LINDDUN (Linking, Identifying, Non-repudiation, Detecting, Data Disclosure, Unawareness, and Non-compliance), and PASTA (Process for Attack Simulation and Threat Analysis). However, none of these techniques is sufficient to model the non-compliance threats of data-driven applications, due to several limitations:

1. **Limited Scope on non-compliance Threats:** Most of the existing threat modelling techniques primarily focus on software security threats rather than regulatory compliance (i.e., non-compliance threats).

2. **Reliance on Predefined Assumptions:** Techniques like LINDDUN, while aimed at data privacy, rely on predefined assumptions [7] (e.g., implementation and security assumptions [8, 9]). As a consequence, these techniques are restricted to addressing only predefined threat scenarios [8].

3. **Lack of Adaptability in Dynamic, Complex Environments:** Existing methods lack the adaptability required for new and evolving types of privacy attacks in complex systems

consisting of interconnected and dynamic components, such as IoT devices like smartphones, traffic cameras, and roadside units (RSUs).

4. **Absence of Legal Focus for GDPR Compliance:** Traditional methods also lack an integrated legal reasoning component for GDPR compliance, which is essential to cover nuanced legal requirements like data minimization and purpose limitation.

To address these limitations, we propose a novel threat modelling technique based on logical reasoning (i.e., defeasible logic), as it provides a flexible and adaptable approach to reason through dynamic and evolving compliance requirements. Defeasible logic stands out by allowing for rules that can be overridden when specific exceptions arise or when new information is available. This is crucial for GDPR compliance, as regulations may evolve and new compliance threats may emerge in complex systems. Defeasible logic also enables handling conflicts effectively (e.g., when rules about data protection conflict with emergency processing needs), by prioritizing the more specific or higher-priority rules, thus making the system resilient to conflicting requirements and adaptable to real-world conditions.

For instance, while LINDDUN is limited to specific privacy threats based on pre-set assumptions, defeasible logic allows for a dynamic model that can adapt as new threats or compliance requirements emerge. By employing defeasible logic, our model provides a comprehensive view of the threat landscape, dynamically adapting to changes and covering a broader range of non-compliance threats, thereby aligning with GDPR's stringent compliance standards.

Furthermore, since May 2018, the new data protection legislation in EU member states and the UK, namely the General Data Protection Regulations (GDPR)[1], has emphasized the need for applications processing personal data to employ tools that can model and analyze compliance with sophisticated GDPR requirements [10]. This highlights the necessity of developing a threat modelling technique that is both adaptable and legally informed which defeasible logic uniquely supports.

Therefore, the use of defeasible logic allows for a robust, adaptable, and legally focused approach to GDPR-compliant threat modelling, overcoming the limitations of existing methods. The advantages of GDPR, which our approach aligns with, are outlined as follows:

1. **Enhanced Data Protection:** GDPR mandates strict guidelines for processing and storing personal data, ensuring that individuals' privacy is safeguarded. This comprehensive framework requires applications and services to implement robust data protection measures, significantly reducing the risk of data breaches and unauthorized access.

2. **Improved Accountability and Transparency:** GDPR promotes greater accountability among organizations managing personal data by imposing strict documentation and reporting requirements. To improve transparency and foster customer trust, organizations

[1]https://gdpr-info.eu/

must clearly outline their data processing activities and ensure compliance with GDPR regulations.

3. **Empowered Individuals with Control Over Their Data:** GDPR grants individuals enhanced rights over their personal data, such as the right to access, rectify, and delete their information. This empowers users to have greater control over their data, ensuring that their personal information is handled according to their preferences and enhancing their overall data privacy rights.

## 1.2 Problem Statement

Data-driven applications are being widely integrated into many aspects of daily life. With our growing dependency on these applications, personal data is being processed more rapidly. It is noteworthy that existing threat modelling techniques are insufficient for modelling data privacy threats (i.e., non-compliance threats) due to their disadvantages, which are listed below.

1. **Limited Scope:** Existing techniques primarily concentrate on system security threats, neglecting the broader spectrum of data privacy threats. This narrow focus leads to incomplete threat models, leaving critical vulnerabilities unaddressed, especially those related to regulatory compliance.

2. **Lack of Legal Focus:** Most current modelling techniques do not incorporate legal requirements, such as GDPR, into their frameworks, which limits their ability to identify and mitigate compliance-related risks in applications handling sensitive personal data, such as *Telehealth Service Systems* and *Fitbit* devices.

3. **Insufficient Adaptability:** The evolving nature of data privacy regulations and emerging threats necessitates modelling techniques that can quickly adapt. However, current techniques lack this flexibility, reducing their effectiveness in dynamic, real-world environments.

## 1.3 Aims and Objectives

The purpose of this thesis is to develop a novel GDPR-compliance threat modelling technique based on Defeasible Logic Programming (DeLP). This technique aims to overcome the challenges faced by existing methods and effectively address compliance risks in data-driven applications. Our solution is a first-of-a-kind technique for practically and sufficiently modelling the non-compliance threats of GDPR; the proposed solution also provides the ability to model any systems with incomplete and conflicting knowledge which is indispensable for representing real-world scenarios.

1. **Develop a Novel Modeling Technique:** Create a first-of-its-kind technique for practically and adequately modelling non-compliance threats under GDPR, ensuring that the technique is scalable and adaptable to various data-driven application contexts.

2. **Overcome Challenges and Enhance Adaptability to Real-world Scenarios**: Address the limitations of current threat modelling techniques that are restricted in modelling non-compliance threats and provide the ability to model any systems with incomplete and conflicting knowledge which are indispensable for representing real-world scenarios.

3. **Enhance Compliance Risk Management:** Provide a robust solution for identifying and mitigating GDPR compliance risks in data-driven applications, ensuring that the developed technique can handle dynamically evolving regulatory requirements and emerging data privacy threats.

4. **Support Practical Applications:** Demonstrate the practical applicability of the proposed solution in various fields, particularly those dealing with sensitive personal data, and validate the effectiveness of the technique in modelling complex systems with incomplete and conflicting knowledge, which are common in real-world situations.

## 1.4 Contributions

This thesis introduces an innovative threat modelling technique for GDPR compliance based on logical reasoning. The solution operates as an expert system equipped with an inference engine that leverages rule-based and defeasible programming languages. It integrates GDPR-compliance-related knowledge, encompassing established security and privacy modelling methods (STRIDE and LINDDUN) along with core GDPR principles on roles, responsibilities, and data protection. The proposed model effectively manages contradictions by prioritizing conflicting rules, enabling comprehensive analysis to identify and mitigate non-compliance threats and thus achieve GDPR compliance. Additionally, the system's reasoning abilities are enhanced to handle scenarios with incomplete knowledge, where it suggests relevant information for system modellers to provide more accurate compliance outcomes. The primary contributions of this thesis are summarized below:

1. **Construction of a GDPR Compliance Knowledge Base:** We developed a comprehensive knowledge base using rule-based and defeasible logic programming languages, establishing a foundational framework for GDPR compliance analysis.

2. **Implementation of a Robust Reasoning Mechanism:** The thesis implements a reasoning mechanism for rule-based and DeLP-based knowledge bases, effectively identifying non-compliance threats and integrating mitigation steps, demonstrating the efficacy of the inference engine.

3. **Integration of Conflicting and Incomplete Information Handlers:** The model incorporates "UNDECIDED" results handlers within the DeLP reasoning mechanism to resolve contradictions and manage incomplete knowledge, demonstrating applicability in real-world scenarios.

4. **Analysis of Reasoning Complexity:** We conducted a complexity analysis of the reasoning mechanism, introducing novel concepts of vertical and horizontal complexity. This analysis provides critical insights into system performance across different query outputs ("YES/NO/UNDECIDED").

## 1.5 Thesis Organisation

This thesis is organised into the following chapters:

**Chapter 2** provides the extensive background and literature related to the GDBR, existing notable threat modelling techniques, Rule-based knowledge representation, and Defeasible reasoning.

**Chapter 3** presents the methodology of our proposed solution, illustrating the high-level system architecture and its main components: the knowledge base and inference engine. A comprehensive GDPR knowledge base is constructed by combining system-default and system-specific knowledge bases. The chapter also explains how the inference engine is used to identify non-compliance threats within the knowledge base.

**Chapter 4** introduces a comprehensive solution for developing a threat modeling technique using Rule-based programming language. This technique addresses GDPR non-compliance threats by integrating GDPR requirements with existing methods like STRIDE and LINDDUN. This chapter illustrates the proposed new data flow diagram that incorporates GDPR principles and demonstrates its application in the *TSS* use case to identify non-compliance threats. Finally, this chapter presents the results and discussions to showcase the feasibility and effectiveness of our proposed solution.

**Chapter 5** elaborates on the suitability of DeLP for modelling GDPR non-compliance threats. It details converting a Rule-based knowledge base into a DeLP knowledge base while maintaining data integrity. The chapter explains the system design and management of logic formulas within the DeLP framework and introduces vertical and horizontal complexity concepts to analyze the Reasoner's complexities. This comprehensive analysis aims to optimize the DeLP reasoner's efficiency in handling dynamic GDPR regulations, providing robust solutions for legal reasoning and compliance checking.

**Chapter 6** details the GDPR-compliance threat modelling tool, focusing on handling "UNDECIDED" query results caused by conflicting rules or missing information. It emphasizes the importance of classifying systems as either compliant ("YES") or non-compliant ("NO") with GDPR. This further elaborates on the implementation of the "UNDECIDED" result handlers. Finally, this chapter discusses the integration of threat mitigation into the defeasible reasoning mechanism to enhance system robustness.

**Chapter 7** demonstrates the identification and handling of non-compliance threats for the *Telehealth Services* use case. Additionally, it describes the experiments conducted to address conflicting rules, missing information, and threat mitigation for the data collection and processing use case in *Fitbit* wearable devices.

**Chapter 8** presents results from the developed technique applied to the *TSS* and *Fitbit* use cases, providing detailed analysis and discussion. Initially, it highlights the outcomes and insights for various query results obtained through DeLP threat modelling for GDPR compliance. Subsequently, it illustrates the performance and analysis of the "UNDECIDED" result handlers, evaluating the proposed system's effectiveness based on horizontal and vertical complexities.

**Chapter 9** concludes the thesis and details future work to be considered for expanding the work discussed throughout the thesis.

# Chapter 2

# Literature Review

This chapter explores the significance of the General Data Protection Regulation (GDPR), the standard for data privacy regulations in Europe. The GDPR, with its 173 recitals and 99 articles across 11 chapters, applies primarily to European organizations but also to non-European entities providing goods, services, or monitoring individuals in Europe [11] [12]. Its core purpose is to safeguard individuals' fundamental rights and freedoms during personal data processing.

However, non-compliance with the GDPR can lead to significant threats. This chapter also describes how existing threat modelling techniques are insufficient to model GDPR non-compliance threats adequately. This chapter analyzes the pros and cons of these modelling techniques. Additionally, it explores the characteristics of defeasible logic to demonstrate its suitability for modelling non-compliance threats in the context of GDPR. Defeasible logic is non-monotonic, allowing conclusions to be revised when new evidence emerges. This feature is crucial for modelling scenarios where regulations and compliance requirements may change or be reinterpreted.

The chapter also elaborates on related work in the field, highlighting potential gaps and limitations. It provides an overview of various existing threat modelling techniques and discusses their limitations in Section 2.1. Section 2.2 covers the importance of GDPR and its various principles. Section 2.3 provides details about data privacy threat modelling for the use-case ACS. A review of knowledge representation and defeasible logic is presented in Section 2.4. Section 2.5 covers the application of priority logic for conflicting information. Section 2.6 discusses related work and its limitations. Finally, the findings of this chapter are summarized in Section 2.7.

## 2.1 Overview of Threat Modelling

Threat modelling is a procedure that is used to (i) determine the security requirements of a system, (ii) identify threats and vulnerabilities, (iii) evaluate the criticality of the detected threats and vulnerabilities, and (iv) prioritize the mitigation methods. Threat modeling relies on traditional security methods, like attack trees and STRIDE. These methods were developed in the

1990s [13]. The modelling of threats requires comprehending the system's complexity and recognizing all possible dangers to the system [14]. It is essential to identify the threats that can occur in a system before claiming that it is secured [14]. Furthermore, the security of the system is defined using a systematic engineering approach [15]. This approach includes the identification of security risks, security requirements, and recovery strategies. It would require less time and effort to address the security issues if security engineering [16] is incorporated in the system design process [17] from the initial architecture specification.

The identification of threats helps in the formulation of realistic and relevant security requirements. This is important because if the security criteria [18] are inaccurate, the system's concept of security is incorrect, and the system cannot be secured. A proper threat assessment [19] reduces the capacity of attackers to misuse the system. The most succinct and basic descriptions of the threat modelling approach have been provided which include four key phases, namely decomposing the system, eliciting the threats, determining the countermeasure and mitigation, and prioritizing the threats. Therefore, a threat modelling technique is typically developed based on a four-step framework in accordance with the four phases, as illustrated in Figure 2.1 [20].

In step 1, a model of the system is created. Data flow diagrams and attack trees, for example, are good ways to illustrate system modelling. In step 2, the threat model/approach is used to find threats. Threat modelling approaches such as STRIDE [20], Attack trees [21], PASTA [22] etc., can be used to find threats in a system. In step 3, these approaches are used to outline mitigation strategies for the threats. Finally, in step 4, the model is validated for completeness and effectiveness (i.e., the system is secured from potential threats).

## 2.1.1 Notable Threat Modelling Techniques

A variety of threat modelling techniques have been proposed and are already used in real-world scenarios, coming with both pros and cons.

### STRIDE

STRIDE is a model-based threat modelling technique developed by Microsoft [23]. It has been effectively applied to cyber-physical systems (i.e., grid systems, robotics systems etc.)[24]. This is a two-way method. In the first phase, a data flow diagram is created to check the flow of data. In the second phase, the STRIDE technique is utilized to identify and model the threats as defined by its name (i.e., Spoofing, Tempering, Repudiation, Information Disclosure, and

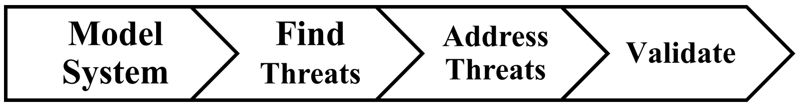


Figure 2.1: The four-step framework for threat modelling techniques

Table 2.1: Threat Categories of STRIDE

| Threat | Security Requirement | Description |
|---|---|---|
| Spoofing | Authentication | Pretending to be something or someone other than yourself. |
| Tampering | Integrity | Try to add/modify something in resources(disk, network, memory etc.). |
| Repudiation | Non-Repudiation | Claiming you were not responsible or did not do something. |
| Information Disclosure | Confidentiality | The information is provided to the one who is not authorized. |
| Denial of Service | Availability | Restrict the resources that are required to deliver. |
| Elevation of privilege | Authorization | Permitting someone to perform a task for which they are not authorised. |

Elevation of Privileges) [25]. The threats identified by STRIDE are listed in Table 2.1, along with their corresponding definitions.

Before the physical installation of systems (i.e., IoT devices, autonomous systems etc.), STRIDE [26] is utilized in the design phase to identify cyber-attacks(i.e., phishing attacks). After that, threat mitigation strategies are employed to stop the identified threats [27, 28, 29]. In addition, The main issue of modelling with the STRIDE is that as the system's complexity grows, so does the number of threats. Another drawback of STRIDE is that it cannot guarantee to model the system's data privacy threats [30].

### LINDDUN

LINDDUN stands for Linkability, Identifiability, Non-Repudiation, Detectability, Disclosure of information, Unawareness, and Non-Compliance. Linkability allows the attacker to connect two or more Items of Interest (IoIs) to establish a link to a specific system. The term ”identifiability" means that the attacker will be able to locate the object of interest. Non-repudiation is another threat in which an adversary attempts to attack a target, but difficult-to-counter evidence. Detectability refers to whether an enemy can identify a target of interest. Furthermore, information disclosure is a security risk that exposes information that should not be exposed [31].

Moreover, Unawareness is a threat that occurs when a user does not know the effects of sharing information. The non-compliance threat shows that the system is not compliant with the regulations and legislation. Table 2.2 illustrates the threat categories of LINDDUN. Further-

Table 2.2: Threat Categories of LINDDUN

| **Threats** | **Properties** |
|---|---|
| Linkability | Unlinkability |
| Identifiability | Anonymity & pseudonymity |
| Non-repudiation | Plausible deniability |
| Detectability | Undectecbility & unobservability |
| Disclosure of information | Confidentiality |
| Content unawareness | Content awareness |
| Policy and content non-compliance | Policy and content compliance |

more, LINDDUN uses the iterative process to discover dangers in a system and then build threat trees [32, 33, 34]. The strong point of LINDDUN is that it has rich privacy documentation. On the other hand, it is a lengthy procedure.

Another deficiency of LINDDUN is that it is based on some pre-defined assumptions and lacks flexibility in complex scenarios where different components interplay with each other. The assumptions are defined in the LINDDUN tutorial [32] as "direct or indirect choices to trust the system components (i.e., data store or data flow) to behave as expected". The LINDDUN threat template (which is included in the supporting materials) allows assumptions to be entered in the 'Remarks' section. However, the study [7] found that most of the assumptions are based on the DFD notation's limitation of expressiveness. Some assumptions directly refer to concepts such as trust and attacker capabilities that are not typically modelled in a system architecture, which raises the question of whether these aspects should be modelled directly as a part of the system [7].

### PASTA

Process for Attack Simulation and Threat Analysis (PASTA) is a risk-centric technique that consists of seven stages [35]. It has several functions which are performed at various phases. Stage 1, defines the objectives; stage 2, defines the technical purpose; stage 3, implements the application decomposition; stage 4, conducts threat analysis; stage 5, conducts vulnerability and weaknesses analysis; stage 6, conducts attack modelling; and stage 7 conducts risk and impact analysis [22], [36]. This technique can be used to meet both business and technical goals [37]. PASTA has rich documentation to assist with its laborious and extensive process [38]. However, this technique is insufficient to deal with data privacy threats.

### Persona non-Grata

The Persona Non-Grata (PnG) modelling approach focuses on attackers, their motivations, and their ability to attack a system. It allows the threat modellers to identify the threats from the counter side. The technical experts try to identify vulnerabilities that are caused by the potential adversary [39]. This technique [40] identifies misuse cases with a target, possible attack scenarios, and adversarial personas [41].

Furthermore, this technique is simple to implement, yet it is underutilized in research. It has a low rate of false positives and a good level of consistency, although it may not be able to detect all threat types [40]. This technique can be used with an agile approach that includes personas.

#### Security Cards

This is an informal technique based on brainstorming to identify novel and difficult attacks. The analysts utilize play cards to answer questions about potential attacks in various scenarios. For instance, why is the system under attack? Who is responsible for this? What kind of assets can be harmed and how can they be harmed? [40].

To identify threats, the Security Cards modelling technique uses a deck of 42 cards such as human impact (9 cards), adversary's motivations (13 cards), adversary resources (11 cards), and adversary's methods (9 cards). This approach can be used to discover almost any form of threat, but it produces a lot of false positives and can not be employed in non-standard scenarios [40]. In industry, the Security Card technique is hardly used [40].

#### hTMM

The Hybrid Threat Modelling Method (hTMM) is made up of SQUARE (Security Quality Requirements Engineering Method), Security Cards, and PnG activities [41], which was developed by the Software Engineering Institute in 2018. The main characteristic of the technique is to provide a consistent result with no false positives(i.e., an object, that has been classified as harmful despite the fact that it isn't a threat), and no overlooked threats [41]. The main steps of hTMM are to highlight the system to be threat-modelled, apply for the security cards, remove unlikely PnG (i.e., there are no realistic attack vectors), use the tool support for finalizing the results, and finally continue to process for risk assessment. The flaw in htMM is that it does not provide mitigation for the threats that have been identified, and it requires a lot of effort to model complex systems.

#### CVSS

The Common Vulnerability Scoring System (CVSS) modelling approach identifies vulnerability attributes and assigns a numerical score to their severity. This establishes a consistent grading system for a variety of cyber-physical systems [42, 43]. CVSS has three metric categories (Base, Temporal, and Environmental), each with a set of measurements.

The scoring algorithms in CVSS can be complex to implement due to the layered nature of these metrics. For example, calculating a Base score involves assessing multiple interdependent factors such as attack vector, attack complexity, and privileges required. This scoring requires users to accurately interpret each metric's parameters, and any misinterpretation can lead to inconsistencies in scoring across similar vulnerabilities. Despite these challenges, CVSS is widely adopted, with its complexity often mitigated by using complementary threat modeling techniques for a more holistic view.

### Attack Trees

This is one of the oldest techniques, and it has been widely used in conjunction with other threat modelling techniques like STRIDE, CVSS, and PASTA [44, 45]. The attacker's aim is put at the root of the tree, while the strategies to achieve the goal are put at the leaf nodes. By travelling through the leaves, AND and OR nodes are used for various aims. Attack trees are used to make security decisions and determine whether the system is vulnerable to attack. The use of the attack tree modelling technique was proposed [45] to develop the threat model of buildings and home automation systems to model the security flaws in their development and implementation. This strategy is simple to understand and only beneficial when security considerations are properly comprehended [45].

### Quantitative Threat Modelling Method (QTMM)

This technique [46] consists of STRIDE, CVSS and Attack Trees. With this technique [47], a few pressing issues could be solved for cyber-physical systems. Another aim of QTMM technique is to generate attack ports for individual components. These attack ports then forward the risk to the connected components. The system risk assessment is done by score- card (i.e., (1) Insignificant, (2) Minor, (3) Moderate, (4) Major, and (5) Catastrophic). If the component root nodes have a high-risk score, the attacked port has a high-risk score as well and is thus more likely to be executed. This is a time-consuming technique and requires high effort to achieve consistent results.

### Trike

Trike is a risk-driven threat modelling method that emphasizes defensive measures by analyzing system security requirements rather than solely identifying attacker capabilities[48]. The process begins with defining the system and creating an actor-asset-action matrix, which maps each actor's potential interactions with assets, identifying any actions that may breach security requirements. By focusing on risk management, Trike prioritizes actions based on risk tolerance, helping organizations align system functions with security objectives. However, Trike's detailed, requirement-centric approach can be complex and time-consuming, often lacking comprehensive documentation, which may limit its scalability in large, dynamic systems.

### VAST

The Visual, Agile, and Simple Threat (VAST) modelling approach is an automated threat modelling approach. Because of its scalability and applicability, this strategy is employed in large organizations to offer actionable and dependable findings for a variety of stakeholders [49, 50].

Two models are developed in this technique: an application threat model that uses data flow diagrams and an operational threat model that is based on attacker mindset DFDs. As a result, VAST can be integrated into the development and DevOps life-cycle of an organization [49]. This technique is used to model security and privacy in intelligent autonomous vehicles [51]. However, this technique is time-consuming and requires extensive effort in modelling a system.

### OCTAVE

OCTAVE (Operationally Critical Threat, Asset, and Vulnerability Evaluation) is a comprehensive, risk-based threat modeling technique focused on identifying and managing organizational risks rather than technical or system-specific vulnerabilities [52, 53]. OCTAVE involves three key phases: building an asset-based threat profile, identifying infrastructure vulnerabilities, and developing a security strategy based on identified risks. A significant advantage of OCTAVE is its flexibility and scalability, as it can be adapted to suit large organizations as well as smaller entities through OCTAVE-S. However, its extensive, in-depth process and limited guidance on specific technical threats make it time-intensive and challenging to document effectively in complex settings.

Table 2.3 illustrates the strengths and weaknesses of threat modelling techniques [37]. These techniques are evaluated based on various parameters such as maturity, focus, time/effort, mitigation etc. The 'maturity' is determined by how effectively each technique is specified, how frequently it has been utilized in case studies, and how frequently it has been coupled with other techniques. The 'focus' shows the point of view or the perspective based on which the technique was designed. The 'Time/Effort' indicator indicates how time-consuming and labour-intensive the procedure is. The term 'mitigation' refers to whether the procedure includes any mitigation strategies. The term 'consistent results' refers to whether or not a technique provides consistent results when repeated. The 'Easy to use/learn' term represents how easily these techniques are adopted. The term 'automation' shows the ability of the technique to be examined in an automated way. Finally, the term 'tool' shows its integration with the Software Development Life Cycle (SDLC).

In this thesis, we prioritize STRIDE and LINDDUN due to their distinct advantages in covering security and privacy threats, which are crucial for determining GDPR non-compliance threats. STRIDE's structured approach to identifying diverse security threats and its adaptability across complex systems make it a robust choice, especially compared to narrower techniques like VAST or Attack Trees. For example, VAST is scalable and integrates well with DevOps, it is less focused on security or privacy threats from a regulatory compliance perspective, which was

Table 2.3: Strengths and Weaknesses of some notable Threat Modelling Techniques [37]

| Threat Modelling Techniques | Maturity | Focus | Time/Effort | Mitigation | Consistent results | Portability | Easy to use/learn | Automation | Tool |
|---|---|---|---|---|---|---|---|---|---|
| STRIDE [25] | High | Defender | High | Yes | No | Yes | Medium | Yes | Yes |
| LINDDUN [32] | High | Assets /Data | High | Yes | No | Yes | No | No | No |
| PASTA [35] | High | Risk | High | Yes | No | Yes | No | No | No |
| CVSS [54] | High | Scoring | High | No | Yes | No | No | Yes | No |
| Attack Trees [45] | High | Attacker | High | No | Yes | Yes | Yes | No | No |
| PnG [40] | Medium | Attacker | Medium | No | Yes | Yes | Yes | No | No |
| Security Cards [40] | Medium | Attacker | Medium | No | No | Yes | Yes | No | No |
| hTMM [41] | Low | Attacker /Defender | High | No | Yes | Yes | Medium | No | No |
| Quantitative TMM [46] | Low | Attacker /Defender | High | No | Yes | Yes | No | No | No |
| Trike [48] | Low | Risk | High | Yes | No | Yes | Medium | No | No |
| VAST [49] | High | Attacker | High | Yes | Yes | Yes | Medium | Yes | Yes |
| OCTAVE [52, 53] | Medium | Risk /Organization | High | Yes | Yes | Yes | No | No | No |

a core requirement of this research. LINDDUN, uniquely tailored for privacy threats, directly aligns with GDPR's compliance requirements, addressing privacy concerns that broader risk-centric methods like PASTA or OCTAVE often overlook. Together, STRIDE and LINDDUN enable a comprehensive threat modelling framework that meets both security and privacy standards, offering the rigour, flexibility, and alignment with evolving compliance needs that other techniques lack. This thesis demonstrates how these security and privacy threats can lead to non-compliance threats, illustrating their interrelationship and how addressing them within our framework supports overall GDPR compliance. In the following section, we will explore the GDPR regulation in detail and illustrate how it can serve as a baseline when using the STRIDE and LINDDUN modeling techniques.

## 2.2 Understanding GDPR

GDPR is a regulation on data protection and privacy [55]. This was enacted in May 2018 in the countries of the European Union [56]. This is an up-gradation of privacy principles proposed in 1995 [57]. GDPR is developed to preserve the privacy of personal data by complying with its principle under strict conditions [58]. Each organization in the EU is obliged to comply with GDPR. If the organization avoids complying with the GDPR then it would be liable to pay a heavy amount of fine[59, 60]. The GDPR is detailed in more than 95 articles that cover all of the technical and administrative principles that govern how corporate and government organizations process personal data [61].

European legislators aimed to harmonize privacy law and enforcement with GDPR [62]. They intended to enhance individual privacy protection while preserving the benefits of data processing [63]. Each EU member state is supposed to have a supervisory authority that is responsible for monitoring GDPR compliance [64, 12].

The organization should comply with Global Privacy Principles [1] such as being clear and transparent; being accountable and keeping personal data secure; taking responsibility and valuing privacy; processing personal data ethically and respecting individual preferences. The international data protection privacy laws are followed and guided by five global privacy principles [65] that include Notice; Choice and Consent; Access and Participation; Integrity and Security; and Enforcement. The principle of ‘Notice’ means that the user should notice and know about the rules available to protect personal information. The ’Choice and Consent’ principle is meant to give individual choices and consent about the use, collection and management of personal information and storage. The ’Access and Participation’ principle states that information should only be utilized and accessed by those who are authorized and have the appropriate security protocols in place. The ’Integrity and Security concept’ is intended to ensure that data is accessed in a secure and authorized manner. Finally, the term ’enforcement’ refers to the process of enforcing compliance with any regulatory model. Therefore, the GDPR is based on international data protection rules, which are an extension of privacy principles.

GDPR is open for interpretation because compliance requirements are abstract. It is made up of seven main principles [66, 67] such as Lawfulness, Fairness, and Transparency; Data minimization; Purpose limitation; Storage limitation; Accuracy; Integrity and Confidentiality; and Accountability. Based on these principles, GDPR is aimed to meet the privacy requirement of personal data.

GDPR defines three main entities [68] such as Data Controller [69] (DC), Data Processor (DP) [70], and Data Subject (DS) [71] play important roles while preserving data privacy. The enterprise must be a data controller and a third-party provider can be a data processor who performs on behalf of the enterprise. There is also a difference between the data controller and the data owner. For example, an accountant can be considered a data controller due to independent judgment which is done to perform professional duties [70]. The various scenarios in which acting as a data controller by enterprise and the third party are discussed in [70].

GDPR gives DS more control over its data by allowing it to exercise various rights such as the right to be informed; right of access; right to rectification; right to erasure; right to restrict processing; right to object; right to data portability; and right to automated decision-making. The DC and DP are responsible for providing access to various rights to DS and fulfilling the request of DS. Moreover, there are six lawful bases of data processing, for example, consent, legitimate interest, contract, legal obligation, vital interest, and public interest. For processing the data, one of these six lawful bases of processing is taken to ensure compliance. For example, the DC should take the consent of the DS before processing its personal data; Without obtaining the consent of the DS, its data can not be processed.

The system can only be considered compliant when the principles identified by GDPR are adopted and the defined duties of DP and DC are performed for preserving the privacy of the in-

[1]`https://globaldma.com`

dividual [12]. Furthermore, to comply with the GDPR, organizations are required to implement appropriate controls and statistical disclosure-limitation strategies. One of the challenges for the implementation is the considerable conceptual gap between legal statements and mathematical formulation around data privacy [72]. The authors explained the concept of "*Predicate Singling Out*" (PSO), which is a privacy attack type that endeavours to capture the notion of singling out occurring in the GDPR. If an attacker identifies a predicate p matching exactly one row in *x* with a probability substantially higher than a statistical baseline, it isolates a dataset *x* using the output of a data-release mechanism *M(x)*. This further demonstrates that PSO security implied differential privacy [73] which is a mathematical concept with legal outcomes. The PSO security of differential privacy and *k*-anonymity are investigated in [74]. Furthermore, the study in [72] depicted that differentiated privacy necessitates PSO security through a relationship to statistical generalisation.

It is worth noting that while the GDPR predominantly applies to organizations within the EU/UK, it extends its obligations to organizations outside the regions that process EU/UK citizens' data [12]. A summary of principles and user rights, along with their related Articles, is provided in Table 2.4 [75].

The next section of this chapter goes into further detail on GDPR principles and how they relate to threats to data privacy.

### 2.2.1 GDPR Principles and the Reciprocity to Data Privacy Threats

GDPR is aimed to provide data protection and privacy to individuals [55]. In today's modern age, preserving the privacy of individuals is not trivial. This sub-section discusses the reciprocity between GDPR principles and data privacy and security by scrutinizing the underlying threats which may occur in case of non-compliance. The relationship between non-complying with the GDPR principles and potential privacy threats will be thoroughly discussed.

#### Lawfulness, Fairness and Transparency

The first principle of GDPR is Lawfulness/Fairness & Transparency. The lawful basis for processing personal data must be considered for the processing to be lawful. There are six lawful bases (i.e., consent, legitimate interest, contract, legal obligation, vital interest, and public interest). If none of the legal bases apply, there will be a violation of this principle, resulting in the unlawful processing of personal data.

Fairness is applied when the data is handled reasonably. This covers how data is collected. The data controller violates the principle of fairness if they have misled someone to collect their data.

According to the principle of transparency, individuals must know which data is obtained, for what purpose, for whom, and for how long it will be kept. This information should be written

Table 2.4: Some important Requirements and Obligations defined in the GDPR. [75]

| **GDPR Article** | **Name** | **Description** |
|---|---|---|
| 5 | Principles relating to Processing of Personal Data | Personal data controllers are obliged to follow principles encompassing lawfulness, fairness, transparency, purpose limitation, accuracy, storage limitation, integrity, and confidentiality. The data controller is mandated to guarantee and demonstrate compliance with the principle of Accountability. |
| 6 | Lawfulness of Processing | Data subjects are required to grant explicit consent for the processing of their personal data for specific purposes, and such processing must adhere to the legal obligations imposed on the controller. |
| 7 | Conditions for Consent | Controllers must be capable of showcasing that explicit consent has been granted by the data subject for the processing of their personal data, with the provision that such consent can be revoked at any given moment. |
| 12 | Transparent Information | Controllers are required to implement suitable measures to furnish the data subject with clear, transparent, understandable, and easily accessible information regarding the processing activities. |
| 13 | Right to be Informed | Controllers are obligated to supply the data subject with particular information at the time of collecting their personal data. |
| 15 | Right of Access | Individuals possess the entitlement to retrieve any personal data maintained by a company. |
| 16 | Right to Rectification | Guarantees the ability to correct any inaccurate data held by controllers. |
| 17 | Right to Erasure | Ensures the capability to delete any data held by controllers upon the subject's request. |
| 18 | Right to Restriction of Processing | Guarantees that any collected data will be utilized solely for the purposes for which consent was granted. |
| 19 | Notification Obligation | Controllers are obligated to inform every recipient to whom the personal data has been disclosed about rectification, erasure of personal data, or restriction of processing. |
| 20 | Right to Data Portability | Empower individuals to carry their data when they depart from the organization. |
| 21 | Right to Object | Enables individuals to raise objections to the processing of their data. |

as clearly as possible in an easily understandable way.

If the processing of data is unlawful, unfair, and non-transparent, the processing of personal data would lead to *data abuse*, and *data exploitation* . For example, it is noted that Amazon processes the data of its users unlawfully without informing the DS which is the transparency requirement (i.e., the right to be informed). It was, therefore, recently fined a large sum of money (i.e. \$877 million) due to the way it collects and shares personal data via cookie consent on its website [2]. Furthermore, the violation of principles of lawfulness, fairness & transparency would lead to privacy leakage with various privacy attacks (i.e., property inference, reconstruction, membership inference, and model extraction etc.) [76].

### Purpose Limitation

This principle states that the processing of the data should be limited to legitimate, explicit, and specific 'purposes' clearly defined in the legal basis (e.g., consent) before the data collection. The processing of data should not be used or transferred beyond the initial purposes for which it has been collected or stored. Generally, processing personal data for new purposes outside of the originally stated purposes is considered unlawful; unless the Data Controller performs and passes a 'Compatibility' test for a new purpose to ensure that the data is still processed on the same 'lawful basis. There are also exceptions including further processing based on EU or member state law and further processing for public interest purposes. Purpose limitation is designed to ensure the confidentiality, reliability, and accuracy of personal data being collected and processed [77]. Preserving purpose limitation is of interest to Data Subject, ensuring the confidentiality of personal data, as well as safeguarding the balance of powers between Data Subjects and Data Controllers [3].

Violation of this purpose limitation principle neglects personal privacy and might lead to various data privacy threats including *data misuse*, *data exploitation*, and *data breaches*. The fundamental purpose of this principle is to protect the Data Subject's privacy from data misuse and data exploitation. For instance, Google has been found to unlawfully feed personal data to advertisers in violation of the purpose limitation principles and unclear data consent policies by the French data regulator [4]. This could go further than just a targeted advertisement, and the damage could be tremendous. Personal data could be processed for numerous illegitimate purposes including (e.g., by using inference attacks [78, 79]) to political campaigns [5].

### Data Minimization

According to the principle of Data Minimization, only the required detail of personal data that is necessary for a specific purpose should be processed by the data controller. The data breaches would result in a violation of the data minimization principle. As H&M was fined (i.e., \$41.4 million) for data breaches that occur due to violating the principle of data minimization [6].

[2]https://www.tessian.com/blog/biggest-gdpr-fines-2020/

[3]https://migrationpolicycentre.eu/point-no-return-migration-and-crime/

[4]https://www.bbc.co.uk/news/technology-46944696

[5]https://publications.parliament.uk/pa/cm201719/cmselect/cmcumeds/1791/179110.htm

[6]https://www.bankinfosecurity.com/clothing-retailer-hm-told-to-wear-41-million-gdpr-fine-a-15111

If the data provided for processing is not minimized sufficiently, there would be the privacy threats of Linkability and Identifiability [7]. Because the excessive availability of data [80] would let the attacker easily find and identify the two items of interest (IOI)s for the specific target. The violation of data minimization would also lead to data abuse [80] and inference privacy attacks (i.e., location privacy attacks, property inference etc.) [81].

### Accuracy

According to the principle of Accuracy, the data provided for processing should be accurate and up to date. Organizations should ensure that the given data is correct and provide the option of erasure and rectification to DS for updating their personal data.

Failing to comply with the 'Accuracy' principle would lead to the processing of data with inaccurate and erroneous data. This would also mean not providing the right to erasure and rectification to the data subject. Therefore, the processing of data with inaccurate data would lead to data abuse and data exploitation [82, 83].

### Storage Limitation

According to the principle of Storage Limitation, organizations should keep personal data until the purpose of processing is achieved. The personal data should be erased after the required processing. Thus, erasure from the storage is needed after the processing. The violation of the 'storage limitation' principle would lead to the linkability and identifiability [8] at the data store because data stored even after the purpose of processing is completed would let the attacker easily identify the two Items of Interests (IOI)s and link it to the targeted object (i.e, AC). The violation of storage limitation would lead to data breach incidents and undesired inference of data [84]. For instance, the data breaches [85] caused by privacy attacks include similarity attack, skewness attack, differential privacy attacks, homogeneity attack and background attack etc. [86, 87].

### Integrity and Confidentiality

According to the principle of Integrity and Confidentiality, DC should ensure the secrecy and confidentiality of personal data. For integrity, the controller should maintain the 'accuracy and validity (consistency)' of the data. There should be the 'trustworthiness' of the data. For confidentiality, data should be protected from unauthorized access, theft, or disclosure of information.

The violation of the '*integrity and confidentiality*' principle would lead to the privacy threats of disclosure of information [9] and data theft because if the data is not secured any unauthorized user can get access to personal data which would lead to the disclosure of information. The

[7]https://www.linddun.org/linddun-threat-catalog
[8]https://www.linddun.org/
[9]https://www.linddun.org/disclosure-of-information

violation of the principle of integrity and confidentiality would also lead to data privacy threats [88] of tempering and unauthorized alteration and destruction of data [89, 90].

**Accountability**

The Accountability principle asserts taking responsibility for whatever you do with the data of the DS. It also enforces showing how you comply with the other principles. Therefore, there should be appropriate measures and records to present compliance with the GDPR.

The violation of the principle of accountability would lead to data breaches [91] and the privacy threat of non-repudiation [91], for which the subject would be held accountable if it is not able to repudiate a claim or action.

GDPR principles provide the compliance requirements that need to be adopted to reduce privacy threats [10]. The relation of data privacy threats with GDPR principles is illustrated in Table 2.5.

Table 2.5: Data privacy threats and related GDPR principles

| **Data Privacy threats** | **Description** | **Consequences** | **Related GDPR Principles** |
|---|---|---|---|
| Linkability | Being capable of identifying whether two items of interests (IOI)s are linked or not. | It can cause identifiability and inference about the particular subject. | Data Minimization |
| Identifiability | The subject can be identified easily within the available set of subjects. | It causes severe privacy violations (when the subject is assumed anonymous). | Data Minimization |
| Non-repudiation | Unable to deny a claim or an action. | If a subject is not able to repudiate a claim/action, it can be held accountable. | Accountability |
| Detectability | The ability to distinguish if an item of interests (IOI)s exists or not. | Inference of a subject can be caused by the detection of an IOI. | - |
| Disclosure of information | This is referred to information disclosure of the subject. | This can lead the disclosure of personal information of subject. | Confidentiality |
| Unawareness | Not aware of impacts and consequences of sharing information. | This can lead to linkability and identifiability. | - |
| Consent non-compliance | The system/organization is compliant if it adheres to the regulatory principle of transparency and takes the user's consent. | This can make consent inconsistent. | Transparency |

The following section provides a comprehensive review of existing modeling techniques used to address data privacy threats concerning the aforementioned GDPR principles and requirements, focusing on a specific use case of autonomous systems.

## 2.3 Data Privacy Threat Modelling for ACS

This section discusses how threat modelling techniques, specifically STRIDE and LINDDUN, can be used to model data privacy threats in a specific autonomous system, namely Autonomous Car Systems (ACS). It examines the challenges and gaps encountered when using these techniques to identify data privacy threats, leveraging GDPR as the baseline.

[10] `https://www.linddun.org/`

### 2.3.1 Use-cases: Data Privacy Threats in ACS

#### Overview of ACS

ACS is one of the milestone inventions in autonomous technology [92] including automotive capabilities based on LIDAR, Radar [93, 94] and machine learning algorithms. The successful implementation of ACS depends on both safety parameters (i.e., the effectiveness of the self-driving mechanisms, cyber-security and data privacy) and human trust [95]. ACS can only be practically deployed in the real world if it is trustworthy [96, 97, 98]. Along with technologies to ensure safety, there should be approaches to educate and enhance users' confidence and trust in ACS [95, 99].

Figure 2.2 [100], illustrates the main components of an ACS in which data acquisition is done by the radars, sensors, cameras, communication devices, and Light Detection and Ranging (LIDAR). Data collected by these devices are manipulated and processed by a central system of the Autonomous Car (AC) and then passed to a decision-support system which lets the system perform a set of required tasks. To travel from point A to point B, AC perceives and gets awareness of the external surroundings, plans an appropriate route, navigates, and makes controlled movements.

Moreover, Figure 2.3 [100] is a simple illustration of the ACS in which AC communicates with other communicating nodes that include the Road Side Unit (RSU), Trusted Authority (TA)(i.e., registration and management authorities), and other connected AC for its fully implemented. Notably, AC communicates with RSU, other connected vehicles, and TA through VANET by LTE, WiFi, visible light communication etc. In ACS a vehicle (e.g., AC) interacts with another vehicle (V2V) [101, 102], and infrastructure (V2I) [103, 104] such as RSU and TA for sharing information (i.e., traffic information, safety warnings etc.).

Furthermore, one of the most serious issues in the automotive industry is the threat to security and privacy [100]. The researchers in [105] and [106] have examined numerous cyber threats in autonomous vehicles. There are a variety of conventional security vulnerabilities in ACS such as the injection of malicious code into various sensors and telematics units [107, 108]; hacking into an in-car network [109, 110]; external spoofing while communication [111]; packet fuzzing [112]; and jamming [113, 114]. Researchers have also demonstrated how an automobile may be readily hacked using a bus of Connected and Autonomous Networks (CAN) [115]. Furthermore, a car communicates with the other car through the CarSpeak mechanism for sharing sensory information [116] which should be protected from privacy concerns. In ACS, personal data is shared with infrastructure and other connected cars for multiple purposes (e.g., safety and value-added services), thus it is crucial to preserve the privacy and security of such data.

**Data Privacy Threats in ACS**

There are a large number of data privacy threats in ACS, as the system collects and processes heterogeneous personal and sensitive information from different sources such as RSUs, central base stations, and other ACs. In [117], the researcher presented the main challenges to safety and security in ACS by identifying various attacks. For example, Sensor Attack [118] occurs when an attacker attempts to disable the GPS by hacking the sensor installed on the car. An attack on VANET [119] is done when a hacker employs brute force to get access to a vehicle's confidential data (i.e., passwords or keys). The V2X attack [120] is held when the attacker attacks any gadget (i.e., smartphone) through which a vehicle communicates to an external network by WiFi, Global System for Mobile or Bluetooth. The V2V attack [121] in which a distributed denial-of-service (DDoS) occurs by overpowering and manipulating the V2V communication. Moreover, GPS spoofing attacks occur when the attacker pretends to be the legitimate terminal

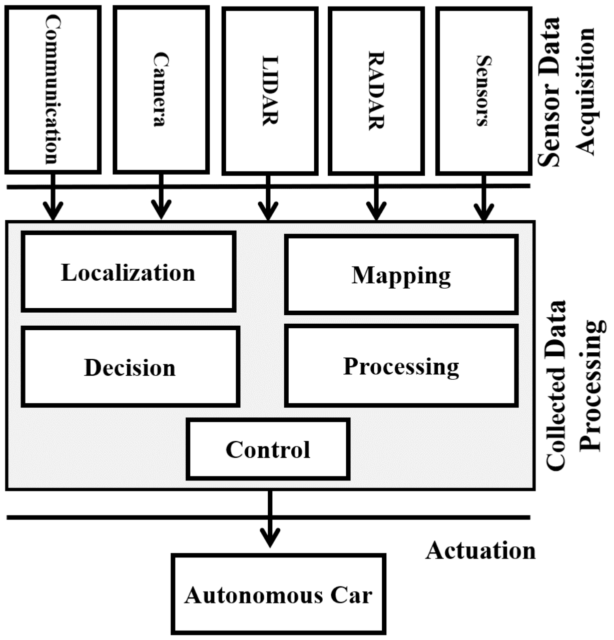


Figure 2.2: AC Architecture

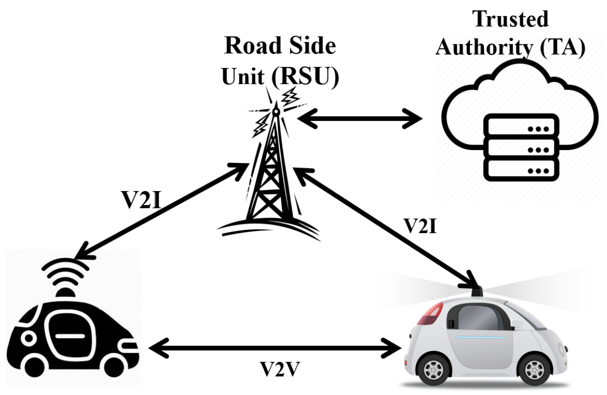


Figure 2.3: Simple illustration of ACS

in the GPS network and tries to access confidential data and pose significant damage to the network. This would let the attacker navigate the AC by spoofing the GPS and taking control of the car.

The AC is more computerized in generating a large amount of data. This system is more vulnerable to privacy concerns[122], since the autonomous industry pays less attention to monitoring and analyzing how data is collected and created by the AC. Third parties and hackers now have more opportunities to abuse the vehicle's data. A hacker can easily access the driver's personal information, the vehicle's location, the information of others in the car (such as passengers), or someone in the vicinity of the automobile.

As demonstrated in Figure 2.2, in AC, the obtained data from the sensors [118] can be used by organizations and third parties for location tracking. In self-driving cars, location data is primarily collected and used for route planning [123]. A data collection that correlates location and travel information (e.g., current area, goal, speed, course, date, and time) may reveal sensitive information about users. These concerns about personal safety exist on both a personal and societal level (Data Protection Report [11]).

Moreover, autonomous vehicles are ideal for acquiring information about different drivers' driving habits, goals, and other information without their consent. Additional issues could arise as a result of the vehicle's use of symbolism, such as ownership questions and potential intrusion of protection claims, depending on the situations in which the images are captured [123]. Similarly, an individual's information around AC (i.e., client's locations and on-street behaviour) may be useful to third parties such as the government and private sector entities, law enforcement, the news media, private specialists, and insurance companies. In the following section, we will elaborate on the detailed challenges in modelling data privacy threats in ACS.

### 2.3.2 Detailed Challenges in Modelling Data Privacy Threats in ACS

As GDPR Principles provide the finest foundation for assuring data protection and privacy in a system, we will utilize them as a baseline to analyze STRIDE and LINDDUN concerning their capabilities in modelling data privacy in ACS.

#### Comparison between threat modelling techniques for ACS

Table 2.6 presents our comparison of the threat model using STRIDE and LINDDUN in accordance with the GDPR principles, individual rights, and other requirements. We use the GDPR as the baseline for comparing STRIDE and LINDDUN, which consists of 7 principles and requirements (e.g. individual rights and international transfer). If one of the requirements of a principle is not covered by a modelling technique, then the main principle is not covered. LINDDUN is

[11]https://www.dataprotectionreport.com/2017/07/the-privacy-implications-of-autonomous-vehicles/

Table 2.6: Modelling threats using LINDDUN & STRIDE in accordance with the GDPR

| **GDPR Principles and Requirements** | **STRIDE** | **LINDDUN** |
|---|---|---|
| **I. GDPR Principles** | | |
| 1. Lawfulness, fairness, and transparency | No | No |
| *1.1 Consent* | - | X |
| *1.2 Legitimate Interests* | - | - |
| *1.3 Contract* | - | - |
| *1.4 Legal Obligation* | - | - |
| *1.5 Vital Interests* | - | - |
| *1.6 Public Interests* | - | - |
| 2. Purpose Limitation | No | No |
| 3. Data Minimization | No | Yes |
| 4. Accuracy | Yes | Yes |
| 5. Storage Limitation | No | Yes |
| 6. Integrity and Confidentiality | Yes | Yes |
| 7. Accountability | No | No |
| **II. Data Subject Rights** | | |
| a. Right to be Informed | No | Yes |
| b. Right of Access | No | Yes |
| c. Right to Rectification | No | No |
| d. Right to Restrict Processing | No | No |
| e. Right to Data Portability | No | No |
| f. Right to Object | No | No |
| g. Right to Automated Decision Making | No | No |
| h. Right to Erasure | No | No |
| **III. International Transfer** | No | No |

37% providing compliance with GDPR as it is mapped with 6 principles. And STRIDE provides 12% GDPR compliance because it is mapped with only 2 principles.

**GDPR Principles**: The comparison of STRIDE/LINDDUN based on GDPR principles [124], and the identified gaps are discussed below:

1. **Lawfulness, Fairness and Transparency:** The privacy requirements of awareness and compliance mentioned in LINDDUN do discuss the consent. But it does not provide any reference about AC users' (i.e. owner/driver and passenger) ability to update/withdraw and view consent; and AC users' consent for sharing data with third parties. On the other hand, STRIDE does not define any threat to processing the data based on lawfulness. Moreover, LINDDUN and STRIDE do not provide any description for processing the AC's users' data on a lawful basis which includes: legitimate interests, contracts, legal obligations, vital interest, and public interest. Similarly, LINDDUN and STRIDE do not provide any reference to the principle of fairness and transparency. Thus, these two modelling approaches do not deal with the compliance threats of un-lawfulness, unfairness, and non-transparency.

2. **Purpose Limitation:** In LINDDUN and STRIDE, we do not find any reference regarding purpose limitation, hence these techniques do not cover this principle. Therefore, STRIDE and LINDDUN do not address the non-compliance threat of 'violating the purpose limitation'.

3. **Data Minimization:** LINDDUN has a reference to data minimization, under the threat tree of Linkability and Identifiability. However, LINDDUN does not include any direct privacy targets/countermeasures or Privacy Enhancing Techniques (PET) to address data minimization, which is regarded as a gap/challenge. STRIDE, on the other hand, shows no evidence of adhering to this principle. As a result, there is a threat of 'non-compliance with data minimization, which must be addressed.

4. **Accuracy:** In LINDDUN, we get references about Accuracy under the threat tree of Unawareness. It also provides a solution for enhancing accuracy by allowing users to delete, update, or review data. In STRIDE, we get a reference for Accuracy/update data under the threat of Tampering, which requires Integrity as a security requirement. Thus, both approaches cover the Accuracy principle.

5. **Integrity and Confidentiality:** LINDDUN and STRIDE define the threat of Disclosure of Information, which has the security requirements of Integrity and Confidentiality. Hence, the principles of Integrity and Confidentiality are covered by these two approaches.

6. **Storage Limitation:** Under the Linkability of a Data Store Threat Tree, LINDDUN displays the potential threats that can arise as a result of storing data for an extended period of time or storing an excessive amount of data. So, we get the reference of storage limitation/retention time in LINDDUN. Furthermore, there is no mention of Storage Limitations in STRIDE. As a result, while modelling using STRIDE, there is a risk of 'non-compliance with the storage limitation' in ACS.

7. **Accountability:** In the use-case of the ACS, the accountability would be held by a Trusted Authority (TA), which would generate revocation/or cancellation of the certificate for the misbehaving AC. LINDDUN does not define any threat related to Accountability. However, it refers to Accountability indirectly, under the Content Unawareness threat tree. Similarly, STRIDE makes no security requirements for Accountability and does not address any threats associated with it. As a result, the threat of 'non-accountability' is not addressed by both modelling techniques.

**Data Subject Rights**: The GDPR requires ACS to implement a variety of Data Subject rights to be compliant with the legislation and to protect Data Subjects from numerous data breaches, data exploitation, and data abuse.

1. **Right to Informed:** LINDDUN refers to the compliance requirement of this principle under the threat tree of Unawareness and Non-compliance. But STRIDE does not provide any reference to cover this right.

2. **Right of Access:** In LINDDUN's Unawareness threat tree, the "unable to review personal information" node refers to individuals' right to access their data without needing physical access to the storage media. Reviewing data can be facilitated through secure methods like remote access, data summaries, or reports, allowing individuals to examine their personal information without directly accessing the physical systems where it's stored. There is also a reference DS to not being able to modify or remove data under the Non-repudiation of the data store threat tree. Moreover, STRIDE does not cover the Right to access, as it does not define any threat related to this right.

3. **Right to Rectification:** Neither LINDDUN nor STRIDE include any references or security/privacy requirements for the Right to Rectification. As a result, both modelling approaches fail to respect the right to rectification.

4. **Right to Erasure:** LINDDUN does not directly define the threat to the right to erasure. It does, however, appear in the Non-repudiation of a Data Store threat tree, where the user is unable to erase their own data. Because this right is not explicitly described in LINDDUN, it is assumed that it is not covered. Similarly, STRIDE does not have any reference related to this right. Thus, both modelling approaches fail to respect the right to erasure.

5. **Right to Restrict Processing:** In both LINDDUN and STRIDE, there is no description/reference of any privacy threat or countermeasure related to the right to restrict processing. Thus, this right is not covered by STRIDE and LINDDUN.

6. **Right to Data Portability:** This right is not covered by STRIDE and LINDDUN, as there is no description or reference related to the right to data portability in these two modelling approaches.

7. **Right to Object:** In both LINDDUN and STRIDE, there is no reference to any privacy threat, related to the right to object. Thus, this right is not covered by STRIDE and LINDDUN.

8. **Right to Automated Decision and Profiling:** This right is not covered by STRIDE and LINDDUN, as there is no description or reference related to the right to an automated decision and profiling in these modelling approaches.

**International Data Transfer:** The compliance requirements of International data transfers in ACS are intended to ensure that the controller/processor complies with the GDPR. However, neither LINNDUN nor STRIDE addresses the threat of 'Non-compliance of international data transfer'.

In the following section, we will elaborate on the challenges in modelling GDPR compliance based on the previously discussed challenges in modelling data privacy threats in ACS.

### 2.3.3 Challenges in Modelling the GDPR-compliance

LINDDUN and STRIDE failed to model non-compliance threats of un-lawfulness, unfairness, and non-transparency, as they do not meet the compliance requirements of lawfulness, fairness, and transparency for processing the AC and its user's personal data. The non-compliance threat of un-lawfulness occurs when the processing of personal data invalidates any lawful basis. Consent non-compliance [12], for example, occurs when trusted authorities fail to obtain the consent of AC's users (i.e., the driver/owner and passenger) before processing and sharing sensitive data with third parties, as well as when users are unable to update, view, or withdraw consent while their data is being processed. Similarly, neither LINDDUN nor STRIDE mentions any compliance requirements for another lawful basis of data processing, such as legitimate interest, contract, legal obligation, vital interest, or public interest. Vital interest is a lawful basis for processing personal data under data protection laws, typically used when the data is necessary to protect someone's life or prevent harm. It applies in emergencies where consent isn't feasible, allowing data processing to safeguard the individual or public from imminent threats. Any lawful basis for data processing can be used to process the data of the ACS. For example, if a malicious AC causes an accident, data processing can be based on 'vital interest'. However, neither LINDDUN nor STRIDE addresses the threat of "not respecting vital interest". As a result, there is a gap in both LINDDUN and STRIDE to cope with the compliance threats of un-lawfulness, unfairness, and non-transparency.

Moreover, STRIDE and LINDDUN modelling techniques lack the privacy/compliance requirements to ensure the principle of 'Purpose limitation'. For example, the data of ACS collected for vehicle management should not be used to share with third parties (i.e., the insurance company - Direct Line Group (DLG)). Likewise, these two modelling approaches do not provide any reference to the compliance requirement of 'data minimization. For example, service providers and data processors (such as RSU, TA, and Uber) should only keep as much data as is required to process it for a specific purpose. LINDDUN and STRIDE also do not meet the 'storage limitation' compliance requirements, which assert holding the data until the purpose of processing is completed. Thus, LINDDUN and STRIDE failed to deal with the non-compliance threats of 'violating of purpose limitation'; 'non-compliance with data minimization'; and 'non-compliance with storage limitation'

Another challenge and gap in LINDDUN and STRIDE modelling techniques are that they do not deal with the compliance threat of 'unaccountability'. In our use-case of the Autonomous Car (AC) system, the principle of 'Accountability' plays a crucial role. If AC misbehaves or has an accident, then AC's users (i.e. driver/owner) should be accountable and explainable for

[12] `https://www.linddun.org/`

this act. For example, in the case of collision and emergency, the AC's owner/driver would be accountable and explain this act [125]. Extensive research is going on the accountability [126, 92, 125] of the autonomous vehicle but none of the existing threat modelling techniques (i.e., LINDDUN/STRIDE) have covered this principle of GDPR.

The compliance requirements for the 'international transfer' of personal data are not guaranteed by LINDDUN and STRIDE. For example, the consent of AC users should be obtained before personal data is transferred for cross-border road safety investigations or commercial purposes (i.e., EU-US privacy shield C/2016/4176) [127].

Furthermore, there is a gap in LINDDUN and STRIDE to meet the compliance requirements of individual rights (Art.12-23). In the ACS, the users of the car should be able to exercise their right to rectification, right to update, right to object, and right to restrict its data processing. For example, it is the right of AC's users to know how their data is gained, stored, shared, and processed (Art. 13, 14 GDPR). The users of the AC may want to access its data from the TA or other data processors where it can exercise its 'right to access' (Art. 15 GDPR). Similarly, these two modelling methods do not respect the 'right to rectification'; 'right to restrict processing '; right to object', 'right to data portability'; and 'right to automated decision making'.

Given the challenges of modelling ACS for GDPR compliance using STRIDE and LINDDUN, we are motivated to explore knowledge representation programming languages and logical reasoning, upon which a novel threat modelling technique can be proposed, as discussed in the following section.

## 2.4 Knowledge Representation and Logical Reasoning

The creation of a suitably precise notation for representing knowledge is the core challenge of Knowledge Representation in which the two significant representation schemes (i.e., Declarative and Procedural schemes) were identified 50 years ago and have been widely used until now [128]. The Declarative schemes are further categorized into logical and semantic network schemes, where logical representation utilizes concepts like constants, variables, functions, predicates, logical connectives, and quantifiers to express facts as logical formulae in various logics (e.g., First or Higher- Order/Multivalued/Modal/Fuzzy, etc.). A knowledge base, from this standpoint, is a compilation of facts and logical rules providing a limited description of the world. Therefore, the fundamental principle of Knowledge Representation is crafting an accurate way to express knowledge, where the use of representation schemes helps us structure and understand complex information about the world effectively [129]. In the following sections, we will discuss different types of knowledge representation programming languages.

### 2.4.1 Rule-based Knowledge Representation

Rule-based knowledge representation is a subset of the logical representation scheme that incorporates the domain knowledge necessary for problem-solving [130]. In this scheme, knowledge is represented in the form of rules, typically structured as `IF(condition)-THEN(conclusion)` statements. This method allows for the efficient organization and retrieval of knowledge by breaking down complex information into manageable rules [131].

To enhance the exploration and utilization of rule-based knowledge bases, hierarchical approaches can be used. These approaches involve clustering rules into hierarchical frameworks, which help in creating coherent and precisely defined clusters of knowledge. By organizing rules hierarchically, it becomes easier to manage large sets of rules, ensuring that related rules are grouped together and can be processed more efficiently.

This method has the important feature of being able to distinguish exceptional rules that differ dramatically from other rules in the same cluster. By isolating these outliers, it becomes possible to address unique or rare scenarios that may require special attention. This capability is crucial for maintaining the accuracy and reliability of the knowledge base [132].

Furthermore, the hierarchical clustering of rules supports the clear and structured representation of domain knowledge. The hierarchical design of this organised approach allows for systematic modifications, which makes it easier to maintain and update the knowledge base.

Overall, rule-based knowledge representation using hierarchical approaches provides an innovative and effective method for managing domain knowledge bases. It guarantees the development of well-organized and precisely defined clusters of rules while concurrently allowing for the identification and handling of exceptional cases, thus enhancing the overall representation and usability of the knowledge base [133].

### 2.4.2 Defeasible Logic Knowledge Representation and Reasoning

Advancements in knowledge representation and reasoning have been significantly influenced by research in non-monotonic reasoning, such as Defeasible Logic (DL), logic programming, and argumentation [134]. The combination of Logic Programming and Defeasible Argumentation has resulted in DeLP so that it is feasible to declaratively define data in the form of weak rules and then employ the argumentation inference mechanism to justify the conclusions drawn from the data [135]. These weak rules are an important component of adding defeasibility that will be used to show a relation between pieces of information that, when all things are taken into consideration, can be defeated.

In DeLP, the language is characterized by three distinct sets: facts, strict rules, and defeasible rules. Facts represent statements that are always true within the logic program. Strict rules represent sound knowledge and are denoted by the symbol "<-" (e.g., Head <- Body). These rules establish a strict connection between the rule's head (consequent) and body (antecedent), meaning that if the body is true, the head must also be true. On the other hand, defeasible rules represent tentative or weak knowledge and are denoted by the symbol "-<" (e.g., Head -< Body). Defeasible rules provide a practical way to represent knowledge that can be used unless valid arguments are presented against it. This type of rule expresses a weaker connection between the rule's head and body, indicating that while reasons to believe in the antecedent (Body) provide reasons to believe in the consequent (Head), this connection can be challenged or overridden. Strong negation in the head of program rules can indicate contradicting knowledge. A query ($q$) in DeLP seeks to determine whether a specific conclusion is justified based on the provided facts, strict rules, defeasible rules, and defeaters through dialectical analysis. A query $q$ succeeds in DeLP when there is a supporting argument ($A$) for it. An argument in DeLP is a logical sequence deriving a conclusion from premises using facts and rules, while a supporting argument specifically justifies a conclusion based on these premises. Moreover, dialectical analysis in DeLP is the process of evaluating arguments and counterarguments to determine which conclusions are justified by constructing and analyzing a dialectical tree, where nodes represent arguments and edges represent defeat relations. Thus, DeLP's structured approach, integrating facts, strict rules, and defeasible rules, enhances its ability to navigate and represent complex knowledge, facilitating the reasoning process.

DeLP uses defeasible inference and dialectical analysis to effectively handle contradictions, warranting queries when their supporting arguments are undefeated, showing its advanced reasoning and data representation capabilities. Defeasible inference in DeLP can be blocked or defeated through weak rules, defining the space where blocking may occur. If a supporting argument for a query is not defeated, the query succeeds. The warrant technique, utilizing dialectical analysis, ensures that if an argument $A$ supporting a query $q$ cannot be refuted, the query is warranted. Certain restrictions are imposed during dialectical analysis to prevent undesirable outcomes, such as an infinite series of defeaters. DeLP adeptly manages contradicting programs and facilitates DL reasoning, allowing the representation of both defeasible and non-defeasible information.

For example, a DeLP knowledge base is used to check GDPR compliance for a wearable device (like Fitbit) processing health data. Facts define that user consent is only given for internal processing $(ConsentGivenForInternalProcessing(user))$, while GDPR requires explicit consent for any external sharing of personal data. Strict rules state that any purpose involving personal data needs consent$(ConsentGiven(Y,X) < -SpecificPurpose(X,Y))$, and defeasible rules infer potential non-compliance if required consent is absent (∼ConsentCompliance(Y,X) -< ∼ConsentGiven(Y,X)).

When queried about GDPR compliance for external sharing, the DeLP Reasoner checks if explicit consent is provided by identifying supporting argument and dialectical tree. Since consent only covers internal use, the argument for non-compliance is undefeated, meaning a GDPR non-compliance threat is inferred due to the lack of consent for external sharing. This demonstrates how DeLP can model GDPR requirements, identifying compliance risks effectively.

### Characteristics of Defeasible Logic

The characteristics of DeLP play a crucial role in enhancing its applicability in legal reasoning, particularly in modelling threats related to GDPR non-compliance. One noteworthy attribute is its defeasibility, acknowledging the potential for conclusions to be overridden or revised, aligning well with the dynamic nature of real-world reasoning processes [136]. Previous research on defeasible reasoning in natural language has explored methods to adjust the probability of a conclusion when new, concrete information is introduced [137, 138]. This flexibility enables the model to adapt to changing circumstances and evolving information, a critical aspect in legal scenarios where contextual shifts are frequent.

Another notable feature of DeLP is the prioritization of rules, enabling the system to consider the significance of certain rules over others [135]. This feature allows one to put preferences on conflicting rules and draw the outcome based on the specified priorities. Hence, the embedded conflict resolution mechanisms by prioritization of rules within DeLP address situations of conflicting rules.

DeLP's proficiency in handling inference with defaults is significant and enables the system to conclude even in the absence of complete information. This capability mirrors the wide and evolving nature of legal arguments and interpretations. Moreover, the non-monotonic nature of DeLP provides a crucial analytical advantage by allowing conclusions to be retracted or adjusted based on new information added to fill the gaps in a knowledge base, reflecting the cautious and adaptable nature of legal analysis [139].

In conclusion, the combination of these distinctive features establishes DeLP as a robust analytical logic for legal reasoning, notably highlighted in its effectiveness in modelling threats related to GDPR non-compliance. DeLP proves valuable by tackling the dynamic, uncertain, and complex aspects inherent in legal scenarios, providing a comprehensive and precise foundation for legal analysis. This contribution significantly enhances the field of legal decision-making. Therefore, the attributes of DeLP and its use in legal reasoning highlight its pivotal role in modelling GDPR non-compliance threats, with priority logic being suitable for resolving conflicts among defeasible rules.

## 2.5 Priority Logic

In the context of logical frameworks for knowledge representation and reasoning, priority is commonly acknowledged as an essential element for characterising an individual's knowledge and its intended application. This prioritization is vital in instances where competing claims or rules must be evaluated against each other to derive meaningful conclusions. For example, the claim that Quakers are generally pacifists is more persuasive than the claim that Republicans are not generally pacifists [140]. Thus, the inclusion of priority within logical frameworks enhances the accuracy and applicability of knowledge representation and reasoning systems, ensuring that they are capable of managing and interpreting conflicting information in a manner that aligns with human reasoning and real-world scenarios.

The authors in [135] have defined DeLP and a preference relation ’>’ among its defeasible rules. When two argument structures where `A` defined as argument and `h` defined as literal, `<A1, h1>` and `<A2, h2>` are compared; If (1) there is at least one rule in `A1` and one rule in `A2` with *ra* > *rb*, and (2) there isn’t any rule in `r'b ∈ A2` and `r'a ∈ A1` with `r'b > r'a`, then `<A1, h1>` is considered superior to `<A2, h2>`. Please note that *ra* and *rb* are the rules in $A1$ and $A2$, respectively.

Defeasible reasoning, a classic challenge, can be adapted to manage conflicting information through a preference-based approach [141, 142, 143]. Defeasible reasoning relies on the prioritisation of rules, favouring the rule with higher priority in conflict situations. Therefore, this prioritization mechanism, ensures that when contradictory information arises, the system can selectively apply the most relevant rule to maintain logical consistency and accuracy.

The conflicting rules are resolved by prioritising the sources of information and choosing data from the source that is most desired in the context of a contradiction [144]. This approach makes use of preferences, which might change based on the situation, to solve reasoning problems. For example, when talking about general statements like ’birds fly’ may be replaced by particular exceptions like ’penguins are birds that do not fly’ [145]. In short, conflicting rules are addressed by assigning priority to the desired sources of information. The following section elaborates on the related work and its limitations.

## 2.6 Related Work

Regulations like the GDPR have specific properties that make DeLP a suitable choice for interpreting its knowledge base. Regulations may complement, overlap, contradict, and change over time [146]. DeLP is well-suited for this due to its intrinsic defeasibility, which enables conclusions to adapt with the introduction of defeaters or arguments. The logical alignment of DeLP in modelling regulations becomes clear when considering the common occurrence of overlapping and contradictory legal texts across various government levels. Thus, the practical application of DeLP in daily legal practice [147] emerges as a notable benefit for system modellers, supporting decision assistance and legal reasoning [148].

Defeasibility is implemented through mechanisms from default logic, prioritization [149], or norm’s partial ordering [150]. An extension of DL, known as Modal Description Logic (Modal DL), incorporates modality to describe preferences among different legal interpretations [151]. This approach involves three types of rules: monotonic rules, non-monotonic (*defeasible*) rules, and *defeaters*, which block conclusions instead of generating new ones [152]. While the direct handling of conflicting information in DL is computationally straightforward, it has found extensive application in modelling various aspects of legal reasoning [153, 154], regulations [155, 156], business rules, contracts [157], negotiation [158], and business process monitoring of compliance [159].

The legal domain has experienced growing interest in applying argumentation and defeasible reasoning. In a recent publication by [160], a comprehensive overview of various logic-based approaches to defeasible reasoning is provided, including Defeasible Logic (DL), Answer Set Programming, *ABA+*, *ASPIC+*, and DeLP. The authors' comparative analysis assesses these approaches from three distinct perspectives: the logical model (related to knowledge representation), the method (encompassing computational mechanisms), and the technology (considering available software resources). However, it is important to note that the paper does not address the specific relevance of GDPR compliance in the legal field, nor does it examine the logical reasoning associated with GDPR compliance.

Similarly, the study by [161] demonstrates how the inherent characteristics of DL, such as expressiveness, contribute to the development of explainable, transparent, and justified intelligent systems for GDPR compliance. The paper specifically focuses on the application of argumentation theory in Medical Informatics, offering an overview of existing approaches documented in the literature. However, it is crucial to highlight that the study does not present empirical results to demonstrate the practical application of DL with conflicting rules for the stated objective. Moreover, the writer [162] argues that defeasible reasoning, valued for its efficiency and simplicity, is well-suited for use as a modelling language in practical applications, such as the representation of regulations and business rules. However, the paper does not provide concrete examples of real-world applications, including how they might be applied to GDPR compliance or how legal rules are prioritized. In a separate study, Pandit et al. [163] discuss the utilization of open and shared technologies, specifically designed for GDPR compliance, to develop knowledge-based systems. Their methodology utilized semantic web technologies (i.e., RDF triples), chosen for their openness and adaptability in describing concepts and relationships. However, the study's scope is limited by the challenges of documenting exceptions within the developed knowledge bases for this regulation.

In light of data protection compliance threat analysis, the authors in [164] have proposed a framework for modelling compliance in which a System Security Modeller (SSM) tool has been developed. This tool is expected to enable the automated detection of compliance issues and end-to-end security concerns during system layout. However, the modeler cannot handle conflicting and missing information, nor can it address emerging threats effectively. In [165], the authors have provided a modelling framework motivated by the *privacy-by-design* concept for designing systems that are GDPR-compliant. Semantic web technologies are also leveraged to represent and query provenance data pertaining to GDPR compliance requirements [166]. But the modeler lacks the capability to manage conflicting and missing information in real time and is ineffective at addressing emerging threats. Additionally, the study fails to offer mitigation strategies in cases of non-compliance. The authors have developed a provenance ontology called GDPRov[13] to express provenance data on consent and data lifecycles. A linked data version of the GDPR text and an ontology defining its many terminology and concepts are both provided by GDPRtEXT in the same work [167]. The semantic web-based approach enables the creation of meaningful knowledge in terms of concepts and relationships with the flexibility to be developed and connected in accordance with requirements. For instance, an interactive ontology for GDPR has been developed by Irem Besik[14]. Nevertheless, these ontologies have never been translated into DeLP, which allows for the incorporation of new information and is well-suited for modeling GDPR compliance threats.

Legal ontology, such as PrOnto [168, 169], aims to model GDPR requirements through DeLP. However, it does not demonstrate how the formal model addresses the resolution of conflicting legal information. Similarly, Sovrano et al. [170] present the scenario where the GDPR *Art. 8* is overridden by corresponding regulation in Italy, achievable through modelling legal rules in LegalRuleML using DL. Nevertheless, this study does not thoroughly explore the complexities involved in converting between LegalRuleML and DL, nor does it address the issue of missing information in the knowledge base. Similarly, the authors in [170] illustrated the scenario where the GDPR *Art. 8* is overridden by corresponding regulation in Italy, achievable through modelling legal rules in LegalRuleML using DL. However, the study did not delve into the intricacies and complexities involved in converting facts and rules from *LegalRuleML* to DL. Another study by [171] proposed a formal modelling approach for GDPR compliance, employing interactive theorem proving, temporal logic, and Kripke structures. But this approach operates within a "completed closed world", requiring all information related to GDPR requirements, security, and privacy to be explicitly formulated, which is not practical in real-world scenarios.

The above research works have tried to enhance the structure and meaning of logical programming and disjunctive databases to accurately depict incomplete information when multiple outcomes are possible. However, most of them fall short of presenting a practical approach for addressing this incomplete information in real-life scenarios including reasoning for GDPR compliance. Many current benchmarks for reasoning are designed with scenarios where some essential information is not provided and must be inferred by the model itself [172, 173, 174, 175]. There are also cases where datasets are structured so that none of the necessary rules is directly given [176, 177, 178, 179]. In the topic of modelling GDPR compliance, our proposed solution should deal with incomplete information scenarios; and the proposed reasoner would generate a list of possible missing information (i.e., facts) in the knowledge base, and then inquire from the user to explicitly supply the facts that need to be re-evaluated.

[13]`https://harshp.com/GDPRov/`
[14]`https://github.com/irembesik/gdpr-ontology`

The distinctiveness of our work lies in five key aspects: (i) introducing an innovative solution using DeLP to model GDPR compliance by developing a comprehensive knowledge base that includes GDPR requirements *(principles and user rights)* (ii) Utilizing an inference engine based on DeLP to infer non-compliance threats over this knowledge base, (iii) handling the "UNDECIDED" query result by resolving the issue of conflicting rules and missing information, (iv) the integration of threat mitigation into reasoning mechanism, (v) A *Fitbit* use-case demonstration focusing on non-compliance threats and dealing with "UNDECIDED" query results.

This chapter provided the related work in the field that outlines its limitations and gaps. The work of this thesis seeks to fill this gap in the literature. The next chapter describes the system design and the proposed methodology of our problem statement.

## 2.7 Summary

This chapter have provided the background regarding existing threat modelling techniques with their pros and cons. The limitation of these existing modelling techniques shows that they are insufficient to model the data privacy threats and more specifically non-compliance threats. This chapter have discussed the GDPR and its various principles in relation to data privacy threats. This chapter has also highlighted knowledge representation and defeasible logic with its various characteristics and described how it is an appropriate tool to model the legal norms and regulations (i.e., GDPR). Furthermore, it provided related work in the field that outlines its limitations and gaps.

The work of this thesis seeks to fill this gap in the literature. The next chapter describes the system design and the proposed methodology of the problem statement.

# Chapter 3

# GDPR-compliance Threat Model: A Holistic Solution Approach

This chapter outlines the methodology of our proposed solution, focusing on the system's high-level architecture, which includes the knowledge base and inference engine. We describe the construction of a comprehensive GDPR knowledge base using various knowledge representation programming languages, such as Rule-based and DeLP-based programming languages. This involves integrating the system's default knowledge base with a system-specific knowledge base to build a comprehensive model for GDPR compliance.

We also detail the utilization of the inference engine in identifying potential non-compliance threats within the knowledge base. Specifically, we employ a Rule-based inference engine for the Rule-based knowledge base and a DeLP-based inference engine to perform reasoning over the DeLP-based knowledge base.

Overall, this chapter provides a thorough overview of the design and functionality of our GDPR compliance modeling system, demonstrating its ability to meet the stringent requirements of modern data protection laws.

## 3.1 Overview of the Solution Approach

This section provides an overview of the development of the proposed GDPR-compliance threat modelling technique based on logical reasoning. Technically, the proposed modelling solution aligns with developing an expert system [180] tailored for GDPR compliance. Thus, this development, as illustrated in Fig. 3.1, the process involves three key steps: constructing a GDPR-compliant knowledge base that stores relevant rules and policies, designing a user interface that allows modelers to specify system details (e.g., adding specific facts and rules into the knowledge base to represent the system accurately), and developing an inference engine to perform logical deductions on the knowledge base. This structure enables the system to dynamically verify and enforce GDPR compliance through consistent, rule-based decision-making.

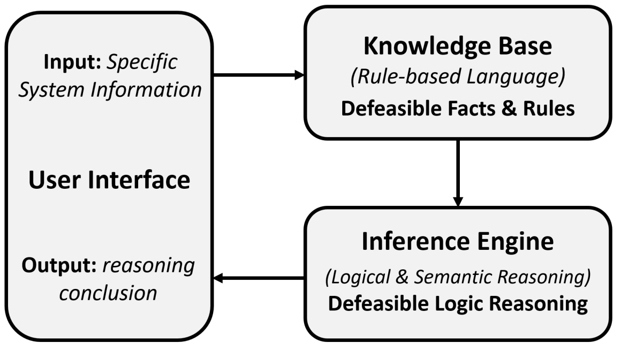

Input: Specific System Information
User Interface
Output: reasoning conclusion
Knowledge Base
(Rule-based Language)
Defeasible Facts & Rules
Inference Engine
(Logical & Semantic Reasoning)
Defeasible Logic Reasoning

Figure 3.1: A high-level system architecture of an expert system for modelling GDPR compliance based on logical reasoning[180]

In the expert system example for GDPR-compliance, the knowledge base contains GDPR rules and policies specific to data access permissions and role-based authorizations, such as which roles are allowed to access certain patient data under the law. The reasoner processes each data access request, comparing it against the rules in the knowledge base to determine if the requester has the necessary permissions. If a non-compliance threat, like unauthorized access, is detected, the reasoner denies access. The user interface allows healthcare personnel to submit data access requests and receive feedback, either granting or denying access based on GDPR compliance. This setup ensures real-time adherence to GDPR by verifying requests before any sensitive data is accessed.

In the next section, we will discuss how we constructed the knowledge base, a crucial component of the expert system.

## 3.2 Knowledge Base Construction

We have constructed a knowledge base for GDPR compliance by interpreting the principles, obligations, and requirements (e.g., depicted in Table 2.4) into knowledge representation languages. For example, for our modeling tool, we use two important types of knowledge representation languages: Rule-based and DeLP-based Programming. Initially, we tried Rule-based programming language to develop the knowledge base [181, 180] and then converted it to DeLP to better represent the legal requirements of the GDPR. This transformation enables the knowledge to be expressed through a combination of related facts, strict rules, and defeasible rules, allowing for a comprehensive representation of real-world scenarios. Constructing a complete knowledge base covering all aspects of the GDPR is a substantial effort. Therefore, in this chapter, we focus on specific GDPR requirements, such as "Consent as the legal basis for data processing", as a proof-of-concept and for demonstration purposes.

Generally, to model a system for GDPR compliance, a knowledge base consists of two parts: (1) general information applicable to all systems (i.e., system-default knowledge base) and (2) specific knowledge for the system being modeled (i.e., system-specific knowledge base). Further explanations of our Rule-based knowledge base and DeLP-based knowledge base, including their respective parts, are discussed in the following sections.

### 3.2.1 Building a Rule-based Knowledge Base

The knowledge base is represented using Rule-based/policy-based language such as RuleML and Semantic Web Rule Language (SWRL). Every rule specifies a relation, recommendation, directive, strategy or heuristic and has the $IF(condition)\ THEN(action)$ structure. As illustrated in Fig. 3.2, the knowledge base for the GDPR non-compliance threat modelling consists of three main areas: (i) STRIDE knowledge base: Rule-based threat library obtained from STRIDE, LINDDUN knowledge base: additional threat trees described in LINDDUN specification and converted to Rule-based language; and (iii) the GDPR knowledge base: obtained from Ontology and expert knowledge using an SWRL as a combination of the Web Ontology Language (OWL) and the Rule Markup Language (RuleML)[1].

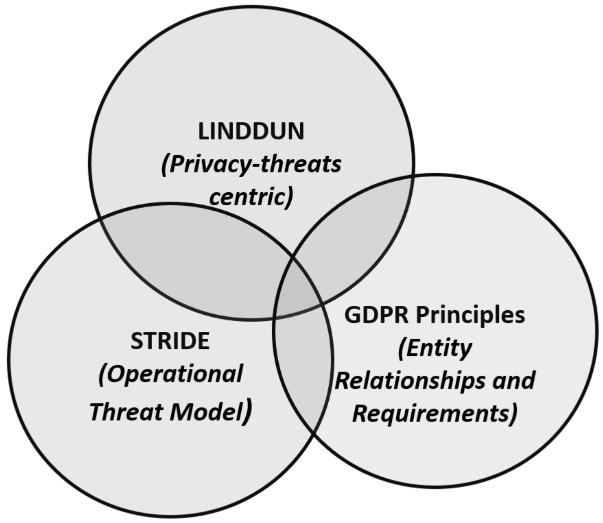

LINDDUN
(Privacy-threats centric)
STRIDE
(Operational Threat Model)
GDPR Principles
(Entity Relationships and Requirements)

Figure 3.2: GDPR-Compliance Modelling Catalyst

The three aspects, i.e., security, data privacy, and GDPR-compliance sets of knowledge, interplay with each other to make sure whether a service provider is compliant with the GDPR. When developing the modeller, we employ STRIDE-the operational threat model with DFD along with data security threats, LINDDUN-privacy threats tree, and the GDPR-compliance baseline including the legal basis, accountability and governance, DS rights, and DS, DC, and DP relationships. The Rule-based knowledge base is comprised of two parts: a default knowledge base and a system-specific knowledge base as follows.

[1]https://www.w3.org/Submission/2004/SUBM-SWRL-20040521/

#### Rule-based Default Knowledge Base

The default knowledge base for the GDPR non-compliance threat modelling consists of three overlapped areas: (i) the STRIDE knowledge base converted to a Rule-based threat library, (ii) the LINDDUN knowledge base which is additional threat trees converted to Rule-based knowledge, and (iii) the GDPR knowledge base. Again, all of this knowledge is represented under Rule-based/Policy-based language such as RuleML [182] and Semantic Web Rule Language (SWRL)[183].

The first two areas are extracted from the existing threat modelling tools whereas the GDPR-compliance knowledge base is developed by our team. For instance, we have taken consent as the legal basis for processing personal data and used *GConsent* the *OWL2 ontology* to represent consent for GDPR compliance. The ontology is based on an analysis of modelling metadata requirements related to the consent lifecycle for GDPR compliance. For example, the consent complaint requirements in our knowledge base are used as expressions such as "*ConsentProvided*" and "*ConsentRequestFormProvided*".

#### Rule-based System-specific Knowledge Base

The second part of our knowledge base is the specific knowledge of a particular system to be modelled. This is the duty of the modeller, who understands the system and will use an existing tool to provide such information to the knowledge base. For example, for our use-case, the *Telehealth Services System (TSS)*, the modeller will add system-specific information to the knowledge base using a provided tool (e.g., the Microsoft Threat Modelling Tool (MSTMT)) with the novel format of the DFD.

In the following section, we present how we developed the knowledge base based on defeasible logic.

### 3.2.2 Building a DeLP-based Knowledge Base

Similar to the Rule-based knowledge base, constructing the DeLP-based knowledge base follows our GDPR-compliance modeling framework. This framework determines whether a service provider is GDPR compliant based on three factors: (i) system security threats, (ii) data privacy threats, and (iii) the GDPR principles and requirements [180]. In this respect, we create our modelling methodology which harnesses STRIDE and LINDDUN and integrates with GDPR principles, as illustrated in Fig. 3.3. Our approach involves utilising STRIDE, an operational threat model to address system security threats, alongside LINDDUN, which presents a privacy threat hierarchy with a primary emphasis on DS privacy threats. Additionally, we integrate the six GDPR principles encompassing legal foundations, DS rights, accountability, governance, and the interplay between DS, DC, and DP. For this purpose, the Rule-based knowledge for GDPR developed in this project [181, 180] is reused and converted into DeLP so that the knowledge is expressed as a collection of facts, strict rules, and defeasible rules to expressively present real-world scenarios. This knowledge base encompasses both the general knowledge shared across all systems (i.e., default knowledge base) and the specific knowledge unique to a particular system that is being modelled (i.e., system-specific knowledge base).

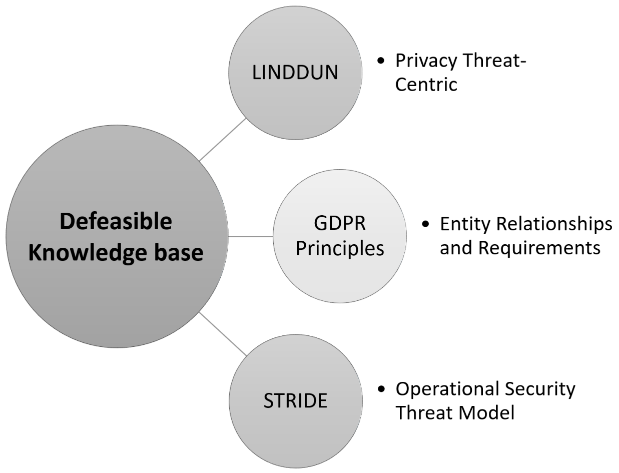

LINDDUN
• Privacy Threat-Centric
Defeasible Knowledge base
GDPR Principles
• Entity Relationships and Requirements
STRIDE
• Operational Security Threat Model

Figure 3.3: Defeasible Knowledge base for GDPR-Compliance consisting of STRIDE Rule-based security threats, LINDDUN privacy threat trees, and GDPR requirements.

### DeLP-based Default Knowledge Base

The default knowledge base encompasses three core domains: (1) STRIDE knowledge base: A Rule-based knowledge base of threats converted into DeLP, (2) LINDDUN knowledge base: Integration of the LINDDUN *Threat Tree* into DeLP knowledge base, and (3) GDPR knowledge base converted from the RuleML and Semantic Web Rule Language (SWRL) knowledge extracted from the project [181, 180] into DeLP.

It is a substantial effort to build a complete GDPR knowledge base, and we focus on requirements of the GDPR for proof-of-concept and demonstration. For instance, we construct a knowledge base for *Consent* as the legal basis for processing personal data. We employ *GConsent* which is an *OWL2* ontology to present GDPR-compliant consent. This ontology is designed after inspecting metadata requirements linked to the GDPR-consent lifecycle. For instance, the specifications for consent compliance in this knowledge base are denoted through terms like "*ConsentGiven*" and "*RevokeConsent*". We also leverage *PrOnto*, which is a Legal Ontology (MeLOn) for modelling GDPR conceptual concepts[168], for our knowledge base.

### DeLP-based System-specific Knowledge Base

As mentioned before, this part of the knowledge base is to depict a system being modelled which involves system-specific insights. The modeller who possesses a comprehensive understanding of the system would employ an existing tool to incorporate these particulars into the knowledge base. For our *TSS* use case, for instance, the modeller will directly infuse system-specific details into the knowledge base in the form of defeasible facts and rules, which supplement the default knowledge base.

In the previous sections, we discussed the construction of the knowledge base, a critical component of our system design. Now, we will elaborate on the inference engine, another essential component, which performs reasoning over the knowledge base. The knowledge base serves as the foundation for the inference engine, with the rules and facts it contains directly informing the engine's decision-making processes.

## 3.3 Reasoning over Knowledge Base

As illustrated in Fig. 3.1, an inference engine is performed over the knowledge base to reason about potential compliance threats in a system. To perform reasoning over different types of knowledge bases, we use various types of inference engines, which are elaborated in the following sections.

### 3.3.1 Rule-based Inference Engine

For Rule-based knowledge base, a Rule-based inference engine is utilised for reasoning, for instance, a semantic reasoner with either *backward-chaining* or *forward-chaining* algorithms. This reasoner takes the knowledge base as its input and iteratively infers new knowledge until a goal has been reached (i.e., finding a specific non-compliance threat) or no rules can be matched (i.e., finding all potential non-compliance threats). For instance, in the demonstration in chapter 4, we use the MSTMT's inference engine which follows the following defined syntax to infer potential threats.

```
Include: IF w is A, and x is B or y is C
Exclude: IF t is D, or u is E and v is F
THEN z = pw + qx * ry
```

The terms *Include* and *Exclude* work as $IF(condition) - THEN(conclusion)$ rules. If the condition is satisfied as defined in *Include*, then whatever condition is written in *Exclude* the inference engine should exclude it for the defined threats in the knowledge base. where A, B, C, D, E, and F are sets in the antecedent and p, q, and r are constants, the variables w, x, y, t, u, and v represent elements (or values) that belong to the respective sets.

### 3.3.2 DeLP-based Inference Engine

We leverage the defeasible reasoning algorithm proposed by [135] over our DeLP-based knowledge base to determine whether a system is compliant with the GDPR. It plays a pivotal role in evaluating the truth value of a query in the form of logical formulas. The mechanism is structured in sequential steps including (1) Receiving Query, (2) Grounding the DeLP Knowledge Base, (3) Generating Arguments, (4) Validating the Arguments, (5) Constructing Dialectical Tree, (6) Marking and Evaluating Nodes, (7) Making Decision (Warranted?), and (8) Returning Query Result as illustrated in Figure 3.4. The implementation of the standard reasoning algorithm proposed in [135] is also available as a part of the Tweety project proposed in [184]. The open source code related to this algorithm can be found here [2].

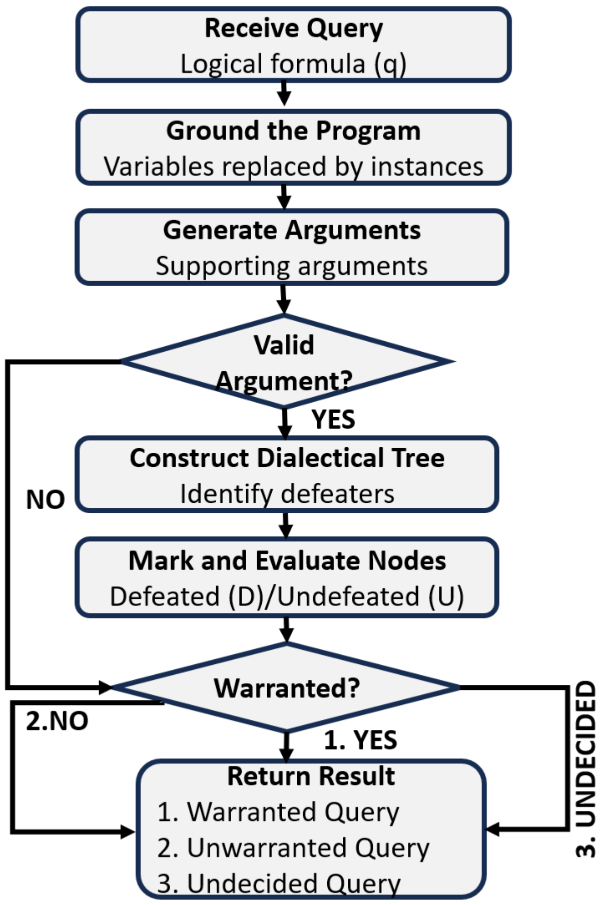


Figure 3.4: The defeasible reasoning route comprises eight sequential steps, spanning from receiving a query to returning the query result.

In detail, the reasoning mechanism follows a systematic procedure that includes the 8 subsequent steps as below, which guarantee the logical derivation of a reliable and valid conclusion from the knowledge base (Figure 3.4).

1. Query Input: The reasoner processes a query formulated as a logical formula. This formula represents a statement for which the reasoner determines the truth value, based on the available knowledge base.

[2] `https://github.com/TweetyProjectTeam/TweetyProject/tree/main/org-tweetyproject-arg-delp`

2. Grounding: This step substitutes variables with specific instances to create a concrete representation of the knowledge base, resulting in a finite set of grounded facts and rules for inference [139]. The grounding process makes the knowledge base finite and well-defined, supporting further analysis.

3. Argument Generation: The reasoner identifies potential arguments in the grounded knowledge base that could support or oppose the query's conclusion, using the finite set of facts and rules from the grounding process to ensure a manageable number of potential arguments.

4. Check Valid Argument?: After generating all potential arguments, the reasoner identifies only those that support the formula's conclusion.

5. Dialectical Tree Construction: For each relevant argument, the reasoner constructs a dialectical tree [135], recursively applying defeater rules to uncover potential exceptions. This process, constrained by the knowledge base's structure, prevents infinite tree expansion [162], ensuring the reasoning process remains manageable and efficient.

6. Marking and Evaluation: As dialectical trees are constructed, nodes are marked as Defeated (D) or Undefeated (U). A node is marked with 'D' when exceptions or defeaters negate its supporting argument. Conversely, 'U' denotes the absence of any negating exceptions or defeaters. This mechanism prevents infinite loops in the evaluation process. The marking mechanism ensures the algorithm does not reconsider the same argument paths, thereby facilitating efficient convergence.

7. Decision-Making (Warranted?): The decision-making process evaluates all constructed dialectical trees, enabling the algorithm to identify whether there are valid arguments supporting or opposing the query's conclusion or if there are no valid arguments that exist at all.

8. Return Query Result: Ultimately, given the finite set of potential arguments and the controlled expansion of dialectical trees, the algorithm concludes with a result of "YES", "NO", or "UNDECIDED".

For instance, we want to query: Is the `TSS_Server` allowed to process `patient_P`'s data under Consent legal basis upon a knowledge base? This query can be expressed in the form of the logic formula $q$: `ConsentCompliance(TSS_Server, patient_P)`. The objective is to determine whether this statement holds true in the knowledge base.

In Step (1) and Step (2), the algorithm verifies if $q$ qualifies as a literal statement and if it can be grounded by substituting variables with instances (i.e., `TSS_Server` and `patient_P`). After Step (3), which generates a list of potential arguments, Step (4) involves identifying a supporting argument (argument $A$) for the warranted query. From the given knowledge base, the algorithm finds the supporting argument, which affirms the query, as the rule: `ConsentCompliance(X,Y) -< DataSubject(Y), DataController(X), ConsentGiven(Y, X)`. This rule asserts if `patient_P` has provided consent, and `TSS_Server` is authorised to process the data.

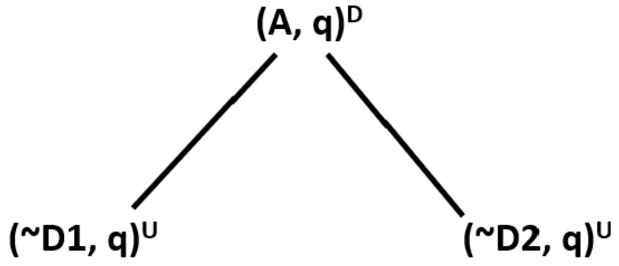


Figure 3.5: A part of a dialectical tree constructed for supporting argument *A*

The algorithm then constructs a dialectical tree for argument *A* as shown in Fig. 3.5 (Step (5)). The root of the tree denoted as $(A,q)$ representing the literals for argument *A* and query *q* (i.e., literal `A=ConsentCompliance(TSS_Server,patient_P)-<DataSubject (patient_P), DataController(TSS_Server), ConsentGiven(patient_P ,TSS_Server)` and `q=ConsentCompliance(TSS_Server,patient_P))`.

In the given knowledge base, argument *A* has two defeaters ~`D1` and ~`D2` that are constructed as two nodes in the tree denoted as `Node1: (`~`D1,q)` and `Node2: (`~`D2,q)`. ~`D1` is a literal obtained from the associated rule `(`~`ConsentCompliance(TSS_Server, patient_P) -< DataSubject(patient_P), DataController(TSS_Server), ConsentExpired(patient_P, TSS_Server)` that affirms the non-compliance as the given consent is expired. Similarly, the defeater ~`D2` affirms `patient_P` has revoked the consent. This concludes that `TSS_Server` is not allowed to process `Patient_P`'s data.

In Step (6), from the dialectical tree traversal, it is concluded that even though `patient_P`'s consent is provided, *A* is defeated due to the existence of some defeaters. Therefore the root of the tree is marked as defeated $'D'$ (i.e., $(A,q)^D$), as represented in Fig. 3.5. `Node1` and `Node2` are marked as undefeated $'U'$ (i.e., $(\sim D1,q)^U$ and $(\sim D2,q)^U$)), implying that no exceptions or defeaters are further applicable in these cases.

In the final step (7), the reasoning algorithm determines that the query *q* is unwarranted and subsequently, in step (8), returns a negative response, `'NO'`. This signifies that, within the framework of defeasible reasoning, the evidence aligns to assert that `TSS_Server` is not authorised to process `patient_P`'s data under the consent legal basis.

## 3.4 Summary

This chapter presents the methodology of the proposed GDPR compliance modelling technique, focusing on the system's high-level architecture, which includes the knowledge base and inference engine. We describe the construction of a comprehensive GDPR knowledge base using rule-based and DeLP languages, integrating both default and system-specific knowledge bases. Furthermore, this chapter explains the utilization of inference engines to identify potential non-compliance threats, employing a rule-based inference engine for the rule-based knowledge base and a DeLP inference engine for the DeLP-based knowledge base.

The next chapter will detail the implementation of our proposed modelling technique for GDPR compliance using a Rule-based programming language, along with its results and limitations.

# Chapter 4

# Rule-based Threat Modelling for GDPR-Compliance

This chapter presents a novel comprehensive solution for developing a threat modelling technique using Rule-based programming languages to address and mitigate threats of non-compliance, taking GDPR requirements as the baseline and combining them with existing security and privacy modelling techniques, such as *STRIDE* and *LINDDUN*. To achieve this, we propose a new data flow diagram integrated with GDPR principles, introducing new entities and their relationships for modelling GDPR compliance. This chapter further illustrates the implementation of the proposed method to infer non-compliance threats using the developed knowledge base for non-compliance threats in the *Telehealth Service System (TSS)* use case. We demonstrate the solution for threats of non-compliance with legal basis and accountability in the *TSS* use case. Finally, this work presents the results, providing analysis and discussions to show the feasibility and effectiveness of the proposed solution.

As mentioned in Chapter 3, the Rule-based knowledge base is constructed by combining the system-default and system-specific knowledge bases, upon which the Rule-based inference engine is utilized to perform the reasoning. The following section presents a novel data flow diagram developed to show how modellers can specify their systems, including GDPR entities and their relationships.

## 4.1 A Novel Data Flow Diagram

GDPR concepts are integrated into the legacy DFD to form a novel DFD. For this reason, GDPR-related roles are introduced to show adherence to the system's regulations. The novel DFD is an idea to complement the existing DFD defined in STRIDE with the new entities and their relationships (Fig. 4.1).

### 4.1.1 New Entities for Modelling GDPR Compliance

The new entities are defined for the GDPR roles such as Data Subject (DS), Data Controller (DC), and Data Processor (DP). Other entities such as Supervisory Authority (SA) (i.e., a government entity to govern compliance with the GDPR) and Reporting Mechanism (RM) (i.e., where DS can lodge a complaint against any data breach, and DC and DP can report any data breach through RM to Supervisory Authority) are also defined. Compliance Trust Boundary Border is also implemented to present that compliance trust boundary would be where a system attains an increased privilege level of compliance. In Fig. 4.1 the circle shape is depicted as processes, the square shape reflects the external entity (e.g., DS), and the entity with a rectangle shape is the traditional Generic Data Store (GDS). Some entities with their attributes are defined below:

```
Element1 Data Controller(DC)
Actions: Provide; Request; Notify; Response;
Accomplish
Properties: ConsentRequestForm; CleanData;
ErasingData; EraseDataWithin28Days; DataBreach

Element2: Data Subject(DS)
Actions: Provide; Request; Complain
Properties: Consent; ErasingData;
DataBreach
```

### 4.1.2 New Relationships for Modelling GDPR Compliance

The interactions among various entities in a complex system result in challenging tasks to model GDPR-compliance [185]. In this project, we define a variety of interactions between the new entities with attributes in order to help the modellers specify their system in detail. In our demonstration, we build new types of relationships between entities in the proposed DFD with attributes based on the GDPR requirements using the MSTMT. For instance, as shown in Fig. 4.1 the DS-DC relationship will have some attributes such as *ConsentProvided*, and *RequestForErasingData* with detailed information on the *Consent* and *Right to Erasure* as follows:

```
Relationship1: DS-DC, DC-DS
Properties: (DS-DC)ConsentProvided,
RequestForErasingData;
(DC-DS)ConsentRequestFormProvided
```

Modellers then can select the relationships with attributes in their systems, providing specific knowledge for the inference engine to accurately determine potential non-compliance threats (i.e., non-consent and non-provided right to erasure).

The chronological order in Fig. 4.1 is represented through the relationships among entities and the directional data flow arrows, which together indicate the sequence in which processes occur within the system. The relationships between entities, such as the DS, DC, DP, RM and

SA, are connected by directional arrows, visually guiding the logical flow of data and establishing the order of operations. The flow begins with the DS, who provides consent (denoted as "Consent Provided") to the DC, marking the initiation of the process. The DC processes this consent and forwards the necessary data to both the GDS for storage and the RM for compliance purposes. The RM subsequently interacts with the SA to ensure GDPR compliance by sending reports. Simultaneously, the DP may receive data from the DC for further processing. The directional arrows connecting the entities clearly define how data flows chronologically through the system, reflecting the progression of GDPR-compliant activities such as data collection, processing, storage, and reporting in a structured and well-defined order.

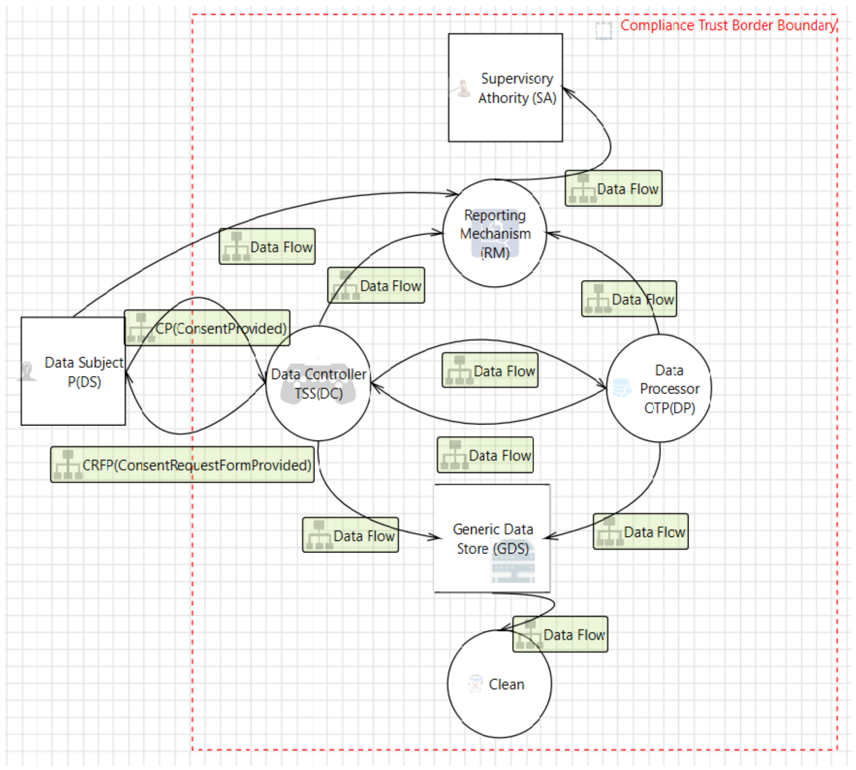


Figure 4.1: The proposed DFD can specify data flow between the new entities (GDPR-related) and the traditional entities (System-related)

In this section, we illustrate the novel DFD for modelling GDPR compliance. In the next section, we will demonstrate how the proposed DFD is mapped onto the roles of telehealth services.

## 4.2 Use-case: Modelling the GDPR Compliance for Tele-health Services System

This section describes how the Rule-based threat modelling technique is employed in *TSS* use case [186].

### 4.2.1 Consent and Right to Erasure in Telehealth Services

Telehealth is the application for the provision of a variety of user-group-specific healthcare services to individuals (e.g., patients, physicians, nurses, etc.) who are located in a diverse range of locations [187]. The TSS has been facing various challenges related to the security and privacy of patients. For example, inadequate security procedures allow for potential data breaches [186], causing patients and healthcare professionals to be vulnerable to security and privacy threats [188]. STRIDE was employed to identify every possible security threat to TSS in [186] and it has shown that the non-compliance threats could result from STRIDE security threats emerging within TSS (e.g., non-consent, non-providing right to erasure, and non-accountability).

In TSS, regardless of any legitimate interest, patients should be requested for consent to process data. The patients might also be requested for consent before their data is processed or shared with third parties by a data processor (such as an Organ Transplant Service). Moreover, a patient may request to erase his/her personal data from the data stores; and the right to erasure may be violated if the DC or DP fails to provide the DS with the required data erasure, posing a non-compliance threat of the non-provided right to erasure.

### 4.2.2 Data Flow Diagram for Telehealth Services

The proposed novel DFD is mapped on the roles of telehealth services as shown in Fig. 4.1. As Patient (P) plays the role of the DS; Telehealth Service Server takes the role of DC; and Organ Transplant Service (OTS) is the DP in the system. Therefore, these entities are playing two separate tasks for the GDPR-related roles and the system-related roles.

As STRIDE does not provide a mechanism for modellers to add GDPR-related information to the DFD, the ultimate purpose of developing the novel DFD is for the modellers to describe their systems in relation to the GDPR legislation. In our proposed modelling tool, modellers can add a variety of entities, relationships, and events with associated characteristics so that it can help infer how the system demonstrates GDPR compliance.

For example, in Fig. 4.1, the relationship between $P(DS)$ and $TSS(DC)$ with the annotations of $CP(ConsentProvided)$ and $CRFP(ConsentRequestFormProvided)$ reflects that the $P(DS)$ provides consent when the consent request form is provided by $TSS(DC)$ for processing the personal data with all of its compliance requirements (i.e., specific, cleared, and direct etc.). The tool automatically determines that consent has been granted and does not add a non-consent threat to the list of threats in the report. I added annotations and directional flow indicators to show that P(DS) provides consent only after receiving the consent request form from TSS(DC). This order is illustrated by labeled arrows, depicting the step-by-step progression of the consent process to make the chronological sequence clear. Additionally, the *clean* process is introduced in the DFD to reflect that the DS would be able to exercise *Right to Erasure* so that personal data is completely erased from the GDS. However, neither the request for *Right to Erasure* from DS is illustrated, nor are its compliance requirements met by DC or DP. As a result, the MSTMT tool equipped with the proposed DFD would generate the threat of a non-provided *Right to Erasure*.

### 4.2.3 Non-compliance Threats

The non-compliance threats (i.e., non-consent, and non-provided *Right to Erasure*) that might occur in the TSS use case are presented in detail as follows:

```
Threat type: non-consent
IF DS.Provide{Consent}=NOT AND
DC.Provide{DS.ConsentRequestForm}=NOT
THEN {non-Consent}

Threat type: non-provided right to erasure
IF DS.Request{DC.EraseData} AND
DC.Request{GDS.CleanData}=NOT AND
DC.Request{DP.EraseData}=NOT AND
DP.Request{GDS.CleanData}=NOT OR
GDS.Response.{cleanData}=Not AND
DC.Notify{RecipientAboutErasingData}=NOT AND
DP.Notify{RecipientAboutErasingData}=NOT AND
DC.Accom Request{EraseDataWithin28Days}=NOT AND
DP.Accom Request{EraseDataWithin28Days}=NOT
THEN {non-provided right to erasure}
```

A non-compliance threat might be a consequence of more than one threat in different categories inferred by STRIDE. For instance, the threat of *non-Accountability* can be an aftermath of some types of security threats identified by STRIDE itself. However, our technique shows that even a system without any security threats related to the *non-Accountability* principle, still, if the system does not implement an RM for which DC or DP can notify SA of data breaches then there will be the threat of non-accountability.

In the telehealth use case, the RM process (depicted from the DFD) demonstrates that the $TSS(DC)$ and $OTS(DP)$ are responsible for notifying SA of data breaches. Otherwise, there would be a threat of non-accountability caused by either $TSS(DC)$ or $OTS(DP)$. The rules for identifying non-accountability threat is depicted below:

```
Threat type: non-accountability
IF DS.Complain{RM.DataBreach} AND
DC.Report {RM.DataBreach}=NOT AND
DP.Report{RM.DataBreach}=NOT
THEN {non-accountability}
```

### 4.2.4 Templates for Modelling GDPR-compliance Threats

A new template for our proposed modelling technique is developed in addition to the built-in templates developed by MSTMT. This template for the GDPR-compliance threat modelling is designed to implement all of the new entities we have introduced (i.e., GDPR role-based entities and relationships) along with pre-defined rules associated with the entities. As a result, this template supports modellers to understand more about the GDPR-compliance requirements and easily model their systems using the tool.

To demonstrate the compliance threats for the use case, a new template designed for TSS has been developed (along with a DFD for the use case illustrated in Fig. 4.1). The DFD uses this template to model the TSS system in an effective and convenient manner.

The GDPR threat modeling template is provided on GitHub[1] and can be applied in practice to model various systems for GDPR compliance. For instance, the template has been used to model a telehealth service system, where the relevant entities and their roles were identified and mapped to GDPR-defined entities such as DS, DC, DP, and SA. By establishing the relationships between these entities, the compliance requirements were demonstrated effectively.

The template supports further analysis by allowing the generation of reports to identify any non-compliance threats. These reports highlight potential GDPR violations and provide mitigation suggestions to address the identified risks. Beyond telehealth systems, the template can be applied to any other system by following a similar approach:

1. Identify the entities and their roles in the target system.

2. Map these entities to GDPR-defined roles (e.g., Data Subject, Data Controller).

3. Analyze their relationships to demonstrate the system's compliance requirements. Generate reports for any identified non-compliance threats and suggest mitigating actions.

We have showcased how the proposed Rule-based modelling technique for non-compliance threats can be employed for the TSS. The next section will further discuss and analyse the results.

## 4.3 Results, Analysis and Discussion

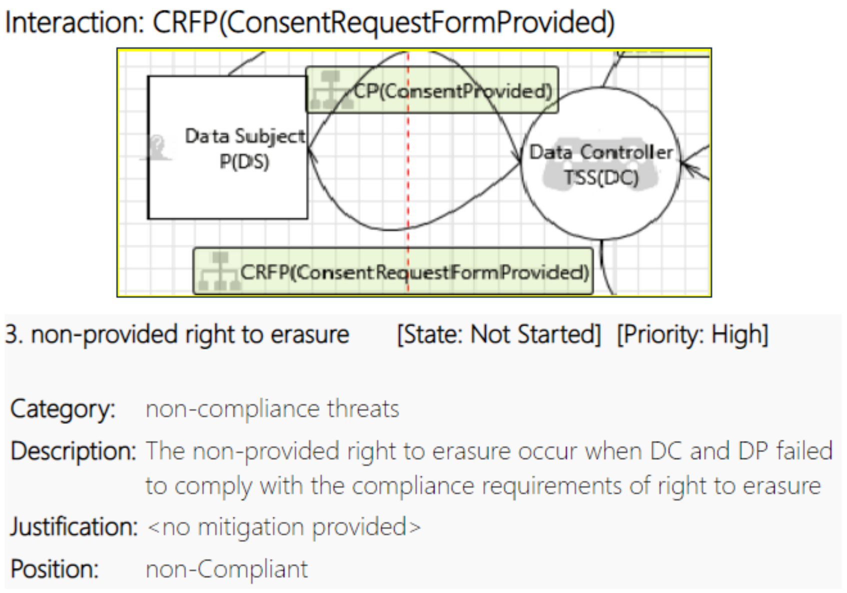
In this section, results from our proposed technique for the TSS use case are presented. An insightful analysis and discussion of the results are also provided.

### 4.3.1 GDPR-compliance Threat Reports

For the demonstration, we use MSTMT equipped with the proposed DFD to identify GDPR-compliance potential threats for the TSS use case. The MSTMT with the novel DFD provides a facility for modellers to describe their system in regard to the GDPR legislation, which can then be further translated into the knowledge base on the back end (i.e., *System-specific Knowledge base*). Combining with the *Default Knowledge base*, the list of potential threats is generated by sparking the MSTMT built-in inference engine over the whole knowledge base (i.e., clicking on

[1]`https://github.com/nailaazam/ModellingGDPRCompliance/blob/main/template/nonCompliant.tb7`

the *'Generate a Report'* button in MSTMT). Based on the output from the inference engine, a report is generated which shows a list of potential non-compliance threats[2]. Fig. 4.2 shows a part of the non-compliance threat report that we have obtained from the tool.


Interaction: CRFP(ConsentRequestFormProvided)

3. non-provided right to erasure [State: Not Started] [Priority: High]

| | |
|---|---|
| Category: | non-compliance threats |
| Description: | The non-provided right to erasure occur when DC and DP failed to comply with the compliance requirements of right to erasure |
| Justification: | <no mitigation provided> |
| Position: | non-Compliant |

Figure 4.2: A part of the GDPR-compliance threats report generated by MSTMT

Threats to non-provided *Rights to Erasure* and *non-Accountability* are identified across the TSS entities and recorded in the report results. Even though the DFD does not provide information to fulfil the compliance requirements of the *Right to Erasure* and *Accountability* principle, the report does not result in a *non-Consent* threat. This is also because in DFD there is an illustration of consent requirements in the form of expressions (i.e., $CP(ConsentProvided)$ and $CRFP(ConsentRequestFormProvided)$).

## 4.3.2 Analysis and Discussion

We have demonstrated the three potential non-compliance threats i.e., non-consent, non-provided right to erasure, and non-accountability for the TSS use case. As shown in Table 4.1, symbol ($\times$) shows the mapping between non-compliance threats and the entities (i.e., the source of the threats) defined in the DFD. As already mentioned the report does not generate the *non-Consent* threat as the system described in the DFD is compliant with the legal base (i.e., Consent) requirement. However, the non-provided *Right to Erasure* and *non-Accountability* threats occur across all of the entities $TSS(DC)$, $OTP(DP)$, $GDS$ except the $P(DS)$. This is because there is not enough information related to the compliance requirements that can be obtained from the DFD for further analysis of potential non-compliance threats.

The results show that if necessary compliance requirements can be obtained from the DFD then the related non-compliance threat will not be presented in the report. It can be understood

[2]https://github.com/nailaazam/ModellingGDPRCompliance

Table 4.1: Mapping between GDPR-compliance Threats and Sources of Threats

| **Compliance Threats** | **TSS(DC)** | OTP(DP) | GDS | P(DS) |
|---|---|---|---|---|
| **Consent** | | | | |
| **Right to Erasure** | × | × | × | |
| **Accountability** | × | × | | |

that the system, as illustrated by the DFD, is compliant with this type of GDPR requirements, except for providing more information to describe the system in more detail. On the other hand, we have shown that a non-compliance threat (i.e., non-accountability) can still occur in a system even though there is no security threat related to it. These results of identifying non-compliance threats from a variety of information sets are evidence of the feasibility and effectiveness of the proposed GDPR-compliance threat modelling technique.

## 4.4 Limitations

Rule-based threat modelling for GDPR compliance has several significant limitations. Firstly, the Rule-based knowledge base strictly follows predefined rules without considering exceptions or changes in context. This rigidity makes it difficult to adapt to new or evolving GDPR requirements and handle intricate scenarios where exceptions might apply or rules might conflict. Additionally, as the number of rules increases, managing and maintaining the Rule-based knowledge base becomes increasingly complex. Scalability is also an issue, as adding new rules or modifying existing ones can negatively impact our system performance.

Moreover, Rule-based systems struggle with ambiguity, which is often present in legal texts and compliance scenarios. They require precise definitions and straightforward scenarios, which are not always available, making it challenging to address uncertain information effectively. The Rule-based knowledge base also lacks contextual understanding, as it operates solely on predefined conditions and actions without considering the broader context, leading to potentially incorrect compliance assessments in complex situations. Furthermore, maintaining and updating the rule base to reflect the latest GDPR amendments and interpretations demands significant manual effort. Continuous monitoring and updates are essential to ensure ongoing compliance, adding to the overall maintenance burden.

## 4.5 Summary

This chapter presents a holistic solution for a threat modeling technique to address and mitigate issues of non-compliance by integrating GDPR legislation with the security and privacy modeling techniques STRIDE and LINDDUN. The proposed technique includes a new DFD for modelers to precisely describe a system with GDPR compliance and to infer non-compliance threats using the developed rule-based knowledge base. For demonstration, we apply this technique to identify non-compliance threats in a telehealth service, focusing on issues such as non-consent, non-compliance with the Right to Erasure, and non-compliance with Accountability. The results demonstrate the feasibility and effectiveness of this modeling technique in addressing such threats.

The next chapter will introduce the use of Defeasible Logic Programming, a non-monotonic formalism with conflict-solving capabilities [135], which supports legal reasoning and compliance checking. We will further illustrate how we implemented the defeasible inference engine to perform reasoning over a constructed GDPR compliance knowledge base.

# Chapter 5

# Defeasible Logic Programming for Modelling GDPR-Compliance

This chapter elaborates on the characteristics of Defeasible Logic Programming (DeLP) that make it more suitable for modelling non-compliance threats to GDPR compliance. It details the process of converting a Rule-based knowledge base discussed in the previous chapter into a DeLP knowledge base, ensuring a seamless transition while maintaining the integrity of the original data and logic.

This chapter provides an in-depth explanation of the system design, outlining the structure of the logic formulas and the implementation of the DeLP reasoner. This includes a step-by-step guide on how the logic formulas are constructed, how rules are defined, and how exceptions are managed within the DeLP framework.

Furthermore, we analyze the complexities involved at each stage of the reasoner's operation. We introduce the concepts of vertical and horizontal complexity to provide a comprehensive understanding of the challenges and performance metrics of the system. Understanding these complexities is crucial for optimizing the reasoner's efficiency and effectiveness in various scenarios. By breaking down the intricate details of the DeLP implementation and its complexities, we aim to provide a clear framework for enhancing the performance and accuracy of GDPR compliance modelling. This comprehensive analysis ensures that the DeLP reasoner can handle the dynamic and evolving nature of GDPR regulations, offering robust solutions for legal reasoning and compliance checking.

## 5.1 DeLP for GDPR-compliance Modelling

GDPR aims to enforce the protection of personal data confidentiality by adhering to its fundamental principles and rigorous standards [189]. To ensure compliance, service providers must navigate through complex legal requirements, exceptions, and contextual considerations, making traditional logic insufficient for capturing the intricacies of GDPR [55]. On the other hand, DeLP allows defeasible reasoning and default inferences, which are essential for modelling the uncertainty and flexibility inherent in GDPR compliance due to its legal dynamics property [190]. With its characteristics, DeLP facilitates the representation of rules and their exceptions, making it suitable for handling situations where certain rules can be overridden by other rules or contextual factors. This capability enables the formalism to accommodate the various exceptions and contextual considerations found in GDPR [11].

In this regard, our research delves into an extensive analysis of constructing a knowledge base using DeLP and conducts DeLP reasoning using DeLP to effectively determine GDPR non-compliance threats.

### 5.1.1 Defeasible Knowledge Base

To lay the foundation for modelling GDPR compliance, we commence by defining relevant facts about entities essential to the regulation, such as DS, DC, and DP. For instance, we may establish a fact denoting $DC(TSS\_Server)$, signifying that $TSS\_Server$ operates as a DC within the system. Subsequently, we develop a set of rules that encapsulate the GDPR principles, legal grounds for data processing, and mechanisms for reporting data breaches. This combination of facts and rules forms a building block for a comprehensive GDPR-compliance knowledge base.

As a requirement of the GDPR, data processing activities can only be carried out once *DC* and/or *DP* obtain *Consent* from *DS*. The GDPR also specifies a DS's right to withdraw consent at any time. When consent is revoked, DC must stop processing data to comply with GDPR. Defeasible reasoning is well-suited to represent this situation. We can define a rule that allows data processing based on consent, and another rule that defeasibly revokes data processing in case of consent withdrawal. This means the initial rule providing consent can be overridden by the defeater rule when the consent is revoked. For example, we can define the following DL rules:

$r1$: `ConsentCompliance(X,Y) -<`
`DS(Y), DC(X), ConsentGiven(Y,X)`

$r2$: `∼ConsentCompliance(X, Y) ←`
`DS(Y), DC(X), ConsentGiven(Y,X), RevokeConsent(Y,X)`

The first rule $r1$ states that if Consent is given by DS(Y) to DC(X), then `ConsentCompliance(X,Y)= YES`, meaning that the DC(X) is compliant with the consent requirement of the GDPR. The second rule $r2$ states that although Consent is given, if it is later revoked, then `ConsentCompliance(X,Y)=NO`. $r2$ is a strict rule (denoted by the symbol $\leftarrow$) and overrides $r1$, which is a defeasible rule (denoted by the symbol $-<$).

DeLP enables us to construct arguments based on the available knowledge base. For instance, given the facts `DS(patient), DC(TSS_Server)` and `ConsentGiven(patient, TSS_Server)`, we can build `ConsentCompliance(TSS_Server,patient)=YES` argument, given a valid consent. However, introducing a new fact `ConsentExpired(patient,` `TSS_Server)` would defeat this argument, resulting in a response of *NO*.

### 5.1.2 Conversion from Rule-based to DeLP-based Knowledge Base

A semantic rule-based knowledge base for GDPR compliance with reasoning capability has recently been investigated and it is still in its infancy [181, 180]. We take a step further to construct a complete knowledge base by integrating the existing related knowledge about the GDPR and integrating with DL which offers non-monotonic reasoning capabilities for the compliance assessment. Fig. 5.1 illustrates the process of the conversion from standard rule-base knowledge to DL as the following steps:

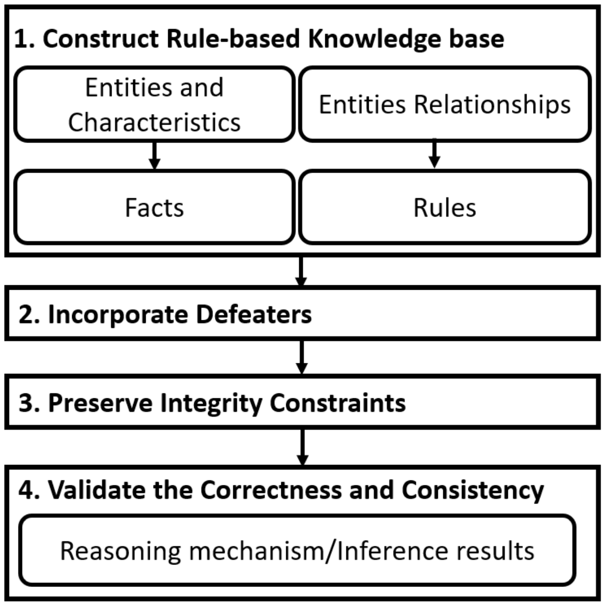

1. Construct Rule-based Knowledge base
Entities and Characteristics
Entities Relationships
Facts
Rules
2. Incorporate Defeaters
3. Preserve Integrity Constraints
4. Validate the Correctness and Consistency
Reasoning mechanism/Inference results

Figure 5.1: Conversion process of transforming rule-based to defeasible knowledge base

1. Construct the Knowledge Base with Insights: To leverage an existing rule-based knowledge base, we begin by analysing its structure and identifying relevant facts, the possibility of strict and defeasible rules, and their relationships. This knowledge is transformed into DL, combining strict rules and facts to describe specific scenarios (e.g., *TSS*), while defeasible rules capture general statements that can be overridden by defeaters.

2. Incorporate Defeaters and Conflict Resolution: Additional information and rules are incorporated into the defeasible knowledge base in the form of DL defeaters. Introducing defeaters enables the expressive description and handling of scenarios such as consent revocation, data breach reporting, and legal data processing grounds.

3. Preserve Integrity Constraints: Integrity constraints are also introduced into the DL knowledge base during conversion as supplementary knowledge in order to guarantee data integrity. This process involves incorporating specific constraints from Compliance's legal

requirements to ensure compliance. These constraints are implemented as defeasible rules within the DL knowledge base.

4. Validating Correctness and Consistency: We created an inference engine to rigorously compare DL-based reasoning with the original rule-based system, ensuring consistent results for known inputs. This validation verifies that the transition to DL maintains the effectiveness of the compliance model.

For converting a rule-based system to a defeasible logic (DL) knowledge base, consider the GDPR rule on data access requests: in a rule-based system, we might have a rule stating, "IF a data subject requests access, THEN access must be granted within 30 days." In the DL knowledge base, this defeasible rule is translated directly as " $RighttoAccess(Y,X) -< AccessGrantedWithin30Days(X,Y)$", maintaining its mandatory nature. Additionally, we add flexibility with a defeater: "IF a data subject is under investigation for fraud, THEN access may be delayed." This translation allows DL to handle exceptions by letting specific conditions override the strict rule when justified, aligning with GDPR's provisions for legal exceptions.

In the above section, we elaborated on the conversion of the rule-based knowledge base to a DeLP-based knowledge base. In the next section, we will illustrate the DeLP-based system design for modelling GDPR compliance.

## 5.2 System Design

The design of our DeLP-based threat modelling technique for GDPR involves several key parts: the knowledge base, reasoning mechanism, and user interface. The knowledge base holds both general GDPR principles along with security and privacy threats (STRIDE and LINDDUN), and specific details about the*TSS*use case.

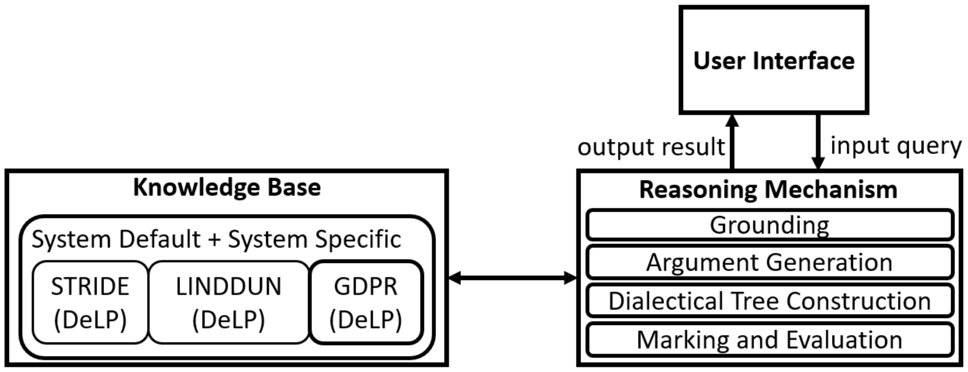


Figure 5.2: System architecture for GDPR compliance modelling system using DeLP with a combined knowledge base

### 5.2.1 System Overview

Our system design centres around constructing a GDPR-compliance knowledge base and a reasoner in which the capabilities of DeLP are leveraged. To achieve this, we first converted the STRIDE and LINDDUN threat modelling frameworks into DeLP, effectively representing various security and privacy considerations. Then, the GDPR-specific knowledge base, which includes facts, strict rules, and defeasible rules tailored to GDPR compliance requirements, is integrated to form the complete knowledge base. Fig. 5.2 depicts the overview of the system including components and interactions among them.

As described in Section 3.2, the knowledge base is comprised of two parts: the default knowledge base, which contains generic information applicable to various scenarios, and the system-specific knowledge base, which incorporates domain-specific facts and rules relevant to the TSS use case. This approach ensures flexibility and adaptability, allowing us to model different GDPR scenarios effectively. To implement the DeLP, we develop our solution based on the Tweety project[1] utilising its functionalities for logic and argumentation reasoning [135, 184]. It also offers tools and libraries that facilitate the development of intelligent systems, reasoning engines, and applications that rely on structured knowledge representation. To perform reasoning, a DeLP reasoner is implemented that takes the constructed knowledge base and a query $q$ as inputs and outputs a conclusion for the query $q$.

### 5.2.2 Logic and Formula

DeLP leverages the concept of DL and Argumentation that allows for the representation of exceptions and defeaters and facilitates defeasible logical reasoning. In this regard, the DeLP knowledge base is comprised of facts, (strict and defeasible) rules and queries that are all in the form of logical formulas. A literal, which stands for an unambiguous assertion, or a more intricate logical expression incorporating variables and predicates can be considered formulas. For instance, a patient $P$ is a DS which is a fact in the knowledge base and is specified by the literal formula `DS(P)`. A defeasible rule is also in the form of a formula like `ConsentCompliance(X,Y) -< DS(Y), DC(X), ConsentGiven(Y,X)` states that if $X$ is $DC$, $Y$ is $DS$, and $Y$ grants consent to $X$, then $X$ is compliant with the GDPR consent legal basis to process $Y$'s data. In this rule, $X$ and $Y$ are specified as variables and can be replaced by appropriate instances (i.e., ground literal).

`∼ConsentCompliance(X,Y) ← DS(Y), DC(X), ConsentExpired(Y,X)` is an example of a strict rule stating that $DC(X)$ is not compliant with consent legal basis for processing $Y$'s data as the given consent is expired. This rule also implies that there should be another fact or rule specifying the given consent between X and Y (i.e., `ConsentGiven(Y,X)`). Note that a strict rule and a defeasible rule are distinguished from one another by the symbols

[1]`https://github.com/TweetyProjectTeam/TweetyProject`

"←" and `--<`, respectively. The latter rule is a defeater for the former rule. An example of a defeasible knowledge base including these two rules is as follows:

```
r1: ConsentCompliance(X,Y) -<
    DS(Y), DC(X), ConsentGiven(Y,X)
r2: ~ConsentCompliance(X,Y) ←
    DS(Y), DC(X), ConsentExpired(Y, X)
r3: ~ConsentCompliance(X, Y) ← DS(Y), DC(X),
    ConsentGiven(Y,X), RevokeConsent(Y,X)
r4: RevokeConsent(Y,X) ← ConsentExpired(Y,X)

f1: DS(patient_P)
f2: DC(TSS_Server)
f3: ConsentGiven(patient_P, TSS_Server)
f4: ConsentExpired(patient_P, TSS_Server)
```

Here $r1$ is a defeasible rule and $r2$, $r3$ and $r4$ are strict rules. $f1$, $f2$, $f3$, $f4$ are facts specifying instances (i.e., $TSS\_Server$ as a DC and $patient\_P$ as a DS) known as ground literals. To perform reasoning over the above-mentioned knowledge base, the DeLP reasoning algorithm is employed, which is elaborated in the following section.

### 5.2.3 DeLP Reasoning Mechanism

We implement a mechanism to reason about a query in the context of DeLP, utilizing the algorithms proposed in [135, 184], as described in the various reasoning steps in Section 3.3.2. This reasoner presents a comprehensive implementation of the algorithm using a DeLP for evaluating the warrantability of a given literal formula (i.e., a query) upon the knowledge base defined in DeLP. The pseudo-code for the implementation is presented as follows:

```
INPUT: delp (DeLP program), q (query formula)
OUTPUT: result (YES, NO, UNDECIDED)

REASONER(delp, q)
// Validate the query formula
1: validate_literal_ground(q)
// Grounding facts and rules in DeLP
2: groundDelp = ground(delp)
// Initialise warrant value for the query
3: warrant = FALSE
// Generate supporting arguments for the query
4: args = get_arguments(groundDelp, q)
// Construct a dialectical tree
5: FOR arg IN args:
  5.1  dtree = createDialecticalTree(arg)
  // Initialise a stack for tree traversal
  5.1  stack = create_Deque()
  5.2  stack.add(dtree)
  // Marking and Evaluation
  5.3  WHILE !stack.isEmpty
          dtree2 = stack.pop()
          defeaters = dtree2.getDefeaters(
                    groundDelp, comparisonCriterion)
          stack.addAll(defeaters)
  5.4  IF dtree.getMarking() == Mark.UNDEFEATED THEN
          warrant = TRUE
          BREAK;
// q not warranted, get arguments for complement of q
6: IF !warrant THEN
  // Negate the query formula
  6.1:  comp_q = q.complement()
  // Initialise a warrant value for comp_q
  6.2:  comp_warrant = FALSE
  6.3:  REPEAT Step 4 & 5 for comp_q & comp_warrant
// Decision Making
7: IF warrant THEN return YES
8: ELSE IF comp_warrant THEN return NO
9: ELSE return UNDECIDED
```

Here are the explanation of each steps of the reasoning mechanism, discussed below:

1. Validate Input Formula: The reasoner first checks if the input formula *q* is valid, ensuring it is literal and in its ground form. This guarantees a meaningful query (Line 1).

2. Ground the Knowledge Base: The reasoner grounds the DeLP by creating a fully instantiated knowledge base, called groundDelp. In this step, all variables in the rules are replaced with constants, generating a specific, grounded version of the defeasible theory (Line 2).

3. Search for Warrantability of *q*: The reasoner seeks arguments concluding with *q* from groundDelp, creating a reasoning chain for *q* (Lines 2 to 5).

4. Construct Dialectical Tree: For each argument, a dialectical tree is built to explore argumentation structures, with arguments and counterarguments represented as nodes. The algorithm uses a stack to navigate through counterarguments, assessing each argument's defeasibility based on the grounded DeLP and comparison criteria (Lines 5 and 5.1 to 5.4).

5. Identify and Compute Defeaters: The dialectical tree computes defeaters by identifying arguments that counter each argument in a tree node, and these nodes are added as the current node's children (Lines 5 to 6).

6. Determine Warrantability: If an argument is "UNDEFEATED," meaning no counterarguments can refute it, $q$ is deemed warrantable. If $q$ is not warrantable, the reasoner then negates $q$ and checks if the negation (i.e., comp_q) is warrantable, following the same steps (Lines 6 and 6.1 to 6.3).

7. Final Decision: Based on the warrantability of $q$ and its negation, the reasoner determines the final result of the query (Lines 7 to 9).

Each step of the above-explained algorithm has its own complexity, which contributes to the overall complexity of the reasoner. The novel concepts of horizontal and vertical complexity are introduced based on which the complexities of the most significant reasoning stages are identified in the following section.

## 5.3 Horizontal and Vertical Complexity of DeLP Knowledge Base

We define the complexity of a DeLP knowledge base by two aspects: *Horizontal Complexity (HC)* and *Vertical Complexity (VC)*. To define these concepts, we formulate a general DeLP knowledge-base as follows:

```
DeLP KB = {
//m number of defeasible rules
rd1: head_rd1 -< tail_rd1_1, tail_rd1_2, ..., tail_rd1_x1
rd2: head_rd2 -< tail_rd2_1, tail_r2_2, ..., tail_rd2_x2
...

rdm: head_rdm -< tail_rdm_1, tail_rdm_2, ..., tail_rdm_xm
...
//n number of strict rules
rs1: head_rs1 <- tail_rs1_1, tail_rs_12, ..., tail_rs1_y1
rs2: head_rs2 <- tail_rs2_1, tail_rs_22, ..., tail_rs2_y2
...
...
rsn: head_rsn <- tail_rsn_1, tail_rs_n2, ..., tail_rsn_yn

//t number of facts.
// tail_rfi: a tail from defeasible or strict rules.
f1: tail_rf1
f2: tail_rf2
...
...
ft: tail_rft
```

For a particular query $q$, there is a set of rules relevant to query $(q)$ (that is, rules are applied in the reasoning process for query $q$) in the DeLP KB called $R_q$. Then, we define the *Vertical Complexity, VC,* of the KB to derive query $q$ is defined as the number of elements in $R_q$, which can be represented by the formula:

$$\mathrm{VC}(q) = |R_q|$$

Accordingly, the *Horizontal Complexity, HC,* is defined as the maximum *Number of Conditions* presented in any single rule ($rd_i$ and $rs_j$ in $R_q$) relevant to the query $q$. This can be represented by the following formula:

$$HC(q) = \max(\text{Number of Conditions}(r_i)), \forall r_i \in R_q$$

In other words,

$$HC = \max(xi, yj); \forall rd_i, rs_j \in R_q$$

Based on these two novel concepts of complexity in the DeLP knowledge base, we will analyze the complexity of the key stages of the defeasible Reasoner.

## 5.4 Identifying the Complexity of Defeasible Reasoner

In this section, we have analyzed the time complexity of the most important stages of the reasoner (i.e., grounding, argument generation, and dialectical tree) that affect the overall complexity of the reasoner. In contrast, the time complexity of the remaining stages of the reasoner is linear and does not significantly impact the reasoner's overall complexity.

### 5.4.1 Grounding

In our DeLP system, both strict and defeasible rules initially use "schematic rules" with variables [135]. Before grounding, these variables represent undetermined entities. During grounding, variables are replaced with specific facts from the knowledge base, merging GDPR rules with system-specific facts into a unified format. This integration forms a comprehensive set of arguments to assess GDPR compliance.

The grounding process, as outlined in Algorithm 1, Generate Ground Instances, provides the implementation details for the grounding process used in Algorithm 2: Grounding a Defeasible Logic Program. Algorithm 1 focuses on the generation of grounded rules by sub-

**Alg. 1:** Generate Ground Instances
**Input** rule (a rule in the defeasible logic program), constants (a set of constants)
**Output** groundInstances (a set of grounded rules with variables replaced by constants)

```
1: Function generateGroundInstances(rule, constants)
2: $groundInstances \leftarrow \emptyset$
3: $combinations \leftarrow$ generateCombinations(rule.variables, constants)
4: for all combination in combinations do
5:     $groundedRule \leftarrow$ substitute(combination, rule)
6:     $groundInstances.add(groundedRule)$
7: end for
8: return groundInstances =0
```

**Alg. 2:** Grounding a Defeasible Logic Program
**Input** DELP (Defeasible Logic Program)
**Output** groundedDELP

```
1: Function groundDELP(DELP)
2: $facts \leftarrow$ DELP.getFacts()
3: $groundedDELP \leftarrow$ new DefeasibleLogicProgram()
4: if DELP is ground then
5:     return copyDELP(DELP)
6: end if
7: for all rule in DELP do
8:     $groundInstances \leftarrow$ generateGroundInstances(rule, facts.getArguments())
9:     for all groundedRule in groundInstances do
10:         groundedDELP.add(groundedRule)
11:     end for
12: end for
13: return groundedDELP =0
```

stituting variables in a given rule with constants derived from the defeasible logic program's facts. The process begins by initializing an empty set, *groundInstances*, to store all grounded rules. It then generates all possible combinations of constants for the rule's variables using the *generateCombinations*() function (line 3). For each combination, the algorithm substitutes the constants into the rule's variables using the *substitute*() function (line 5), producing a grounded rule. Each grounded rule is added to the set *groundInstances* (line 6). Once all combinations are processed, the algorithm returns the complete set of grounded rules.

Algorithm 2, on the other hand, integrates the functionality of Algorithm 1 within the broader process of grounding an entire defeasible logic program. It begins by retrieving facts from the input program and initializing an empty grounded program, groundedDELP. If the program is already grounded, it directly returns a copy of the program (lines 4–6). Otherwise, it iterates over each rule in the program (line 7), invoking *generateGroundInstances*() (Algorithm 1) to create all grounded instances of the current rule (line 8). For each grounded rule returned by Algorithm 1, the algorithm adds it to *groundedDELP* (lines 9–11). This process is repeated for all rules in the program, ensuring that every possible grounded instance is generated and incorporated into the new grounded defeasible logic program. For example, if the fact "*GiveConsent*(*customer*, *serviceProvider*)" is present, and a rule contains the variables 'C' and 'S', Algorithm 1 would generate all possible grounded rules by substituting combinations of constants like 'customer', and 'serviceProvider'. Algorithm 2 uses these grounded rules to construct the final grounded logic program. Together, these algorithms enable effective reasoning by transforming abstract rules into their concrete grounded counterparts.

The time complexity of the whole grounding process is calculated by multiplying the number of rules by the permutations of constants from the facts, allowing for repetitions. This results in the formula $\mathcal{O}(n \times c^r)$, where $n$ is the number of rules in the DeLP, c is the number of distinct constants, and $r$ is the number of variables in each rule. This formula helps gauge how the grounding scales with the complexity of the knowledge base. The following example illustrates a simple knowledge base before and after grounding:

```
Before:
r1: ValidConsent(C,S) -< GiveConsent(C,S)
f1: GiveConsent(customer,serviceProvider)

After:
r1: ValidConsent(customer,customer)-<
GiveConsent(customer,customer).
r2: ValidConsent(serviceProvider,customer)-<
 GiveConsent(serviceProvider,customer).
r3: ValidConsent(customer,serviceProvider)-<
 GiveConsent(customer,serviceProvider).
r4: ValidConsent(serviceProvider,serviceProvider)-<
 GiveConsent(serviceProvider,serviceProvider).
```

```
f1: GiveConsent(customer,serviceProvider).
```

This knowledge base example demonstrates that rule $r1$ is transformed into four grounded statements by replacing variables with all possible constant permutations. Each rule generates $c^r$ instances, with both $c$ and $r$ equal to 2, leading to four instances. Despite the grounding formula suggesting exponential growth, in practice, especially in the GDPR knowledge base, rules typically involve no more than two variables. This suggests that grounding can be efficiently completed in a linear time frame when the values of $c$ and $r$ are small.

## 5.4.2 Argument Generation

The argument generation algorithm, as outlined in Algorithm 3, processes a grounded DeLP to identify all valid arguments for a given query. It starts by initializing a stack with an initial entry containing the query's conclusion and all applicable rules (line 4). The algorithm then iterative processes this stack, expanding each partial derivation by applying rules to unresolved formulas and avoiding cyclical reasoning (lines 5-20). Once a derivation has no remaining formulas, it is added to the set of finalized arguments. This process repeats until all possible derivations are explored, ensuring a comprehensive validation of arguments that support the query.

Before analyzing this algorithm's complexity, we define two key terms: *vertical complexity* and *horizontal complexity*. Vertical complexity refers to the number of distinct rules related to a query, while horizontal complexity refers to the number of facts needed to satisfy a single rule. An example knowledge base illustrates these concepts:

```
r1: A(C,S) -< B1(C,S),B2(C,S),B3(C,S).
r2: X(C,S) -< Y1(C,S).
r3: Y1(C,S) -< Y2(C,S).
r4: Y2(C,S) -< Y3(C,S).
```

In the example, rule $r1$ has a horizontal complexity of 3 because it requires three facts (B1, B2, B3) to satisfy $A$, and a vertical complexity of 1 as it is not linked to other rules. Conversely, rule $r2$ has a vertical complexity of 3 due to its connections with two additional rules, rule $r3$ and rule $r4$, but a horizontal complexity of 1, needing only $Y1$ to satisfy $X$.

Our time complexity analysis for the argument generation algorithm focuses on grounded rules related to our query, recursively exploring derivations starting from the query in the initial stack. When horizontal complexity exceeds 1 for a rule, the algorithm evaluates all permutations according to the following formula:

$$\sum_{i=0}^{\text{horizontal complexity}} P(\text{horizontal complexity}, i)$$

This results in a time complexity of $\mathcal{O}(h!)$, where $h$ represents the horizontal complexity. Extending this to all related rules, or the vertical complexity, the overall time complexity becomes

**Alg. 3:** Generating all valid arguments
**Input** rules (Collection of rules), conclusion (Conclusion to be proven)
**Output** derivations (Set of all derivations)

```
1: Function AllDerivations(rules, conclusion)
2: $stack$ ← empty stack
3: $initial$ ← (empty list, {conclusion}, rules)
4: $stack.push(initial)$
5: $derivations$ ← empty set
6: while stack is not empty do
7:     $derivation$ ← stack.pop()
8:     if derivation.second is empty then
9:         no more formulas to be proven
10:        $derivations.add(derivation.first)$
11:    else
12:        for formula in derivation.second do
13:            for rule in derivation.third.getRulesWithConclusion(formula) do
14:                $new\_derivation$ ← copy of derivation
15:                new_derivation.first.append(rule)
16:                new_derivation.second.remove(formula)
17:                new_derivation.second.add(rule.getPremise())
18:                new_derivation.third.remove(rule)
19:                if no cycles in new_derivation.first then
20:                    stack.push(new_derivation)
21:                end if
22:            end for
23:        end for
24:    end if
25: end while
26: return $derivations$ =0
```

$\mathcal{O}(v \times h!)$, where v is the vertical complexity.

Unlike scenarios described in Section 5.4.1 where linear time complexity may be achievable by optimizing constants, reducing complexity in rules with high horizontal complexity is more challenging due to their inherent "AND" relationships, which generally resist simplification.

### 5.4.3 Dialectical Tree

After generating all valid arguments, the dialectical tree is created, marked, and evaluated to determine if the query is warranted, assessing the validity of arguments within the tree structure.

**Alg. 4:** Get Defeaters
**Input** delp (a defeasible logic program), comparisonCriterion (a comparison criterion)
**Output** children (a set of dialectical tree nodes representing defeaters)

```
1:  Function getDefeaters(delp, comparisonCriterion)
2:  if $delp == NULL$ then
3:      throw IllegalArgumentException("Cannot compute defeaters for NULL DeLP")
4:  end if
5:  $attackOpportunities \leftarrow argument.getAttackOpportunities(delp)$
6:  $attacks \leftarrow \emptyset$
7:  for all $lit \in attackOpportunities$ do
8:      $attacks.addAll($DelpReasoner.getArgumentsWithConclusion(delp, lit)$)$
9:  end for
10: $defeaters \leftarrow \emptyset$
11: $defeaters \leftarrow attacks.stream()$
12:     .filter(attack $\rightarrow$ isAcceptable(attack, delp, comparisonCriterion))
13:     .collect(Collectors.toSet())
14: $children.clear()$
15: $children.addAll(defeaters.stream()$
16:     .map(defeater $\rightarrow$ new DialecticalTree(this, defeater, this.depth + 1))
17:     .collect(Collectors.toSet()))return $children = 0$
```

**Alg. 5:** Process Arguments and Construct Dialectical Trees

```
1: for each $arg$ in $args$ do
2:     $args \leftarrow$ allDerivations
3:     Create $dtree$ with $arg$
4:     Initialize $stack$ as an empty dequeue
5:     Add $dtree$ to $stack$
6:     while $stack$ is not empty do
7:         $dtree2 \leftarrow$ pop from $stack$
8:         Add all defeaters of $dtree2$ to $stack$ using
              getDefeaters(groundDelp, comparisonCriterion) method
9:     end while
10:    if $dtree$.getMarking() equals DialecticalTree.Mark.UNDEFEATED then
11:        Set $warrant$ to true
12:        Break
13:    end if
14: end for=0
```

The process of constructing the Dialectical tree starts by initially identifying the defeaters. Algorithm 4 is responsible for identifying defeaters for a given argument within a dialectical tree structure. A defeater is an argument or set of rules that conflicts with or overrides the current argument in the defeasible logic program (delp). The algorithm begins by validating the input delp (line 2), throwing an exception if it is null. It then retrieves the attack opportunities (line 5), which represent the points where the current argument can be challenged. For each attack opportunity (lines 7–9), the algorithm identifies potential attacking arguments using the knowledge base $(DelpReasoner.getArgumentsWithConclusion())$.

The algorithm proceeds to filter the attacking arguments by determining their acceptability (lines 11–13), based on a predefined comparison criterion $(comparisonCriterion)$. Only acceptable attackers are retained in the defeaters set. For each defeater, a new dialectical tree node is created, which represents the defeater in the tree structure (lines 15–16). These nodes are added to the children set of the current argument node and returned as the output of the algorithm (line 17). This step-by-step process ensures that all relevant and acceptable defeaters are identified, making the dialectical tree ready for evaluation.

Algorithem 5 uses the $getDefeaters()$ subroutine (4) to construct and evaluate dialectical trees, which determine whether a query is warranted (justified). The algorithm starts by iterating over all valid arguments $(args)$ in the defeasible logic program (line 1). For each argument, a dialectical tree node $(args)$ is created (line 2), and a deque stack is initialized to manage further exploration of the tree (line 3). The root node is then added to the stack (line 4).

While the stack is not empty, the algorithm pops a tree node (dtree2) from the stack (line 6) and computes its defeaters using the $getDefeaters()$ method (line 7). These defeaters are added as children to the current tree node, effectively growing the dialectical tree structure. The algorithm continues to process all child nodes until the stack is empty. At this point, it evaluates the marking of the dialectical tree (lines 9–12). If the tree is marked as UNDEFEATED, meaning no defeaters can invalidate the root argument, the query is deemed warranted, and the evaluation halts (line 13). Otherwise, the tree is further processed or evaluated as DEFEATED.

This results in a complexity of $\mathscr{O}(n^2)$, where $n$ is the size of the valid arguments. However, for "YES" and "NO" outcomes, we can approximate the complexity to $\mathscr{O}(n)$, as we do not need to exhaustively search through all possible valid arguments as in "UNDECIDED" query.

To resolve the query's outcome, the algorithm assesses whether the query or its complement is warranted without defeaters. If neither is warranted, it returns "UNDECIDED," which generally takes longer to process than "YES" or "NO" outcomes. Once a query is warranted, displaying the dialectical tree can provide detailed insights into GDPR compliance by illustrating the exact rules and facts that influenced the decision. The time complexity of this process is $\mathscr{O}(a * n)$, where $a$ is the number of valid arguments and $n$ is the size of the knowledge base, allowing for a comprehensive yet efficient analysis to identify potential defeaters.

### 5.4.4 Complexity of the Reasoner

Combining the above-mentioned three important stages of Reasoner, we conclude that for a knowledge base with low horizontal complexity, the time complexity can be approximated as $\mathcal{O}(n)$, where $n$ is the size of the knowledge base for "YES"/"NO" queries, and $\mathcal{O}(n^2)$ for "UNDECIDED" queries. However, with high horizontal complexity, the factorial time complexity becomes dominant, leading to a time complexity of $\mathcal{O}(h!)$ where $h$ represents the horizontal complexity. This assumes a low count of variables and constants in the grounding process, which is a practical approach to adopt.

## 5.5 Summary

This chapter describes why the characteristics of DeLP are more suitable for modeling non-compliance threats to GDPR compliance and how we convert a Rule-based knowledge base into a DeLP knowledge base. It further provides an in-depth explanation of the system design, outlining the structure of the logic formulas and the implementation of the DeLP reasoner. Additionally, the complexities of each stage of the reasoner are analyzed, introducing the concepts of vertical and horizontal complexity to better understand the challenges and performance metrics of the system. These complexities are crucial for optimizing the reasoner's efficiency and effectiveness in various scenarios.

The next chapter will elaborate on the implementation of reasoning outputs handling and threat mitigation.

# Chapter 6

# Outputs Handling and Implementation

This chapter firstly describes how our GDPR-compliance threat modelling tool resolves "UNDECIDED" query results which are the consequences of conflicting rules or missing information. This is crucial because any system should be either compliant (i.e., "YES") or non-compliant (i.e., "NO") with the GDPR; but not "UNDECIDED". To tackle this challenge, we ask users (i.e., modellers) to provide more information, either explicit priorities for some rules or more knowledge to describe their systems (i.e., facts). Secondly, we implemented the "UNDECIDED" results handlers by developing the conflicting information handler and the missing information handler. Finally, if a system is not compliant with the GDPR, we carry out the non-compliance *Threat Mitigation* by providing insights into the system to unveil the reasons for its non-compliance and how to potentially mitigate the threat of non-compliance.

The proposed DeLP-based modelling tool for handling the "UNDECIDED" results consists of key components such as User Interface, Knowledge Base, Reasoner, and Results Handlers as illustrated in Figure 6.1. The *User Interface* enables users to query and interact with the system; the *Knowledge Base* contains facts and a combination of defeasible and strict rules; the *Reasoner* employs a structured reasoning process, and the *Result Handler* tackles the issue of "UNDECIDED" query results and recommends mitigation strategies for potential threats of non-compliance.

## 6.1 Handling "UNDECIDED" Results

As discussed in Section 3.3.2, the DeLP-based Reasoner executes the reasoning algorithm to determine whether or not the query (or the negation of the query) is warranted ("YES" or "NO"). If there are no valid or sufficient supporting arguments for the query, the reasoner yields an "UNDECIDED" result. To address this, we customise the reasoning algorithm proposed in [135] and integrate an "UNDECIDED" query results handler. As can be seen in Figure 6.1, if the query result is "UNDECIDED" then the reasoner will identify the reasons (conflicting arguments & missing information) and pass them to the Handler to re-evaluate the query for a warrant (i.e., "YES" or "NO").

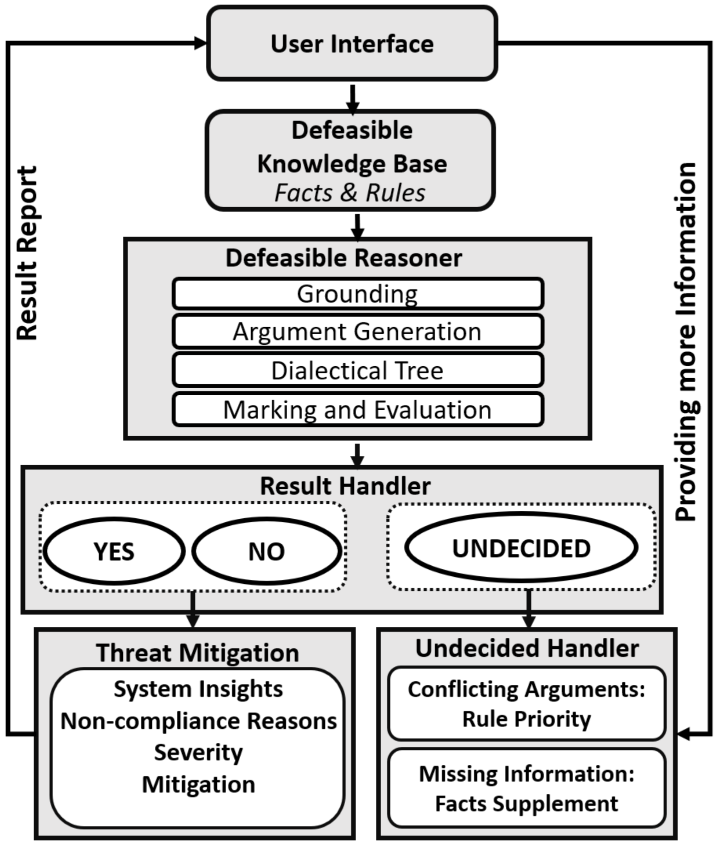


Figure 6.1: System architecture for GDPR compliance modelling system integrated with "UNDECIDED" result handler.

Suppose we have two defeasible rules in the knowledge base for data access:

```
r1: GrantAccess(X,Y) -< ConsentProvided(Y,X)
r2: DenyAccess(X,Y) -< nonComplianceThreat(Y,X)
```

Now, if a data subject requests access and we know they have provided consent (satisfying r1) but lack information on whether there is a security risk, the system may return an "UNDECIDED" result due to missing information. Alternatively, if it's determined that there is both consent and a potential security risk, the conflicting rules (grant vs. deny) will also lead to an "UNDECIDED" result.

In both cases, the DeLP system marks the query as "UNDECIDED" until more information is available or a priority rule is set to resolve conflicts. This ensures that the system only provides a conclusive result when conditions are clear, preventing unauthorized access.

In the next section, we will discuss that how the conflicting information and missing information handlers has been implemented.

### 6.1.1 Conflicting Rules

Conflicting rules arise when two or more rules lead to contradictions or opposing directives during reasoning over a knowledge base, potentially resulting in an "UNDECIDED" query outcome. For instance, under GDPR's *Article 17*, individuals have the right to erasure. This right applies when the personal data is no longer necessary for the purpose for which it was collected. However, legislation requiring data retention for specific periods may also apply to organizations, which could lead to a conflict between individual rights and legal obligations. Without predefined rules or priorities to resolve such conflicts, the reasoner might be unable to conclusively decide whether to comply with the erasure request or deny it based on legal obligations. Consequently, this leads to the "UNDECIDED" query result due to the two contradictory rules.

The system is unable to identify conflicting rules and to decide which rule should take precedence resulting in "UNDECIDED" outcomes. As a result, the lack of clarity complicates GDPR compliance, undermines data protection, and creates uncertainty about rights and responsibilities for both individuals and organizations.

### 6.1.2 Incomplete Knowledge Base

The DeLP-based GDPR knowledge base often has an "UNDECIDED" query result on compliance when important information is missing. The important information is system-related facts, if they are missing, leads to an incomplete knowledge base. For example, the compliance requirements regarding data processing activities, consent records, or data sharing agreements missing from the set of facts. Similarly, the knowledge base may not contain comprehensive records of DS requests, such as requests for data access, rectification, or erasure etc. Without this information, it becomes challenging to ensure that the organization processes DS requests in compliance with GDPR requirements. Thus such gaps in the knowledge base lead to the "UNDECIDED" query result.

### 6.1.3 Rule Priorities for Conflicting Arguments

We handle the "UNDECIDED" outcomes by applying rule priorities on conflicting arguments that occurred during the reasoning process. The main purpose of incorporating rule priorities is to efficiently resolve ambiguities caused by conflicting rules. This is particularly important for our DeLP-based GDPR knowledge base where compliance to the GDPR requirements is crucial. This mechanism is also beneficial for complex knowledge bases with hundreds of thousands of rules and facts, where manually identifying and prioritizing conflicting rules can be challenging. Additionally, the rule-priority approach facilitates effective reasoning within the complex knowledge base. It enables the system to navigate through complex GDPR requirements scenarios by focusing on the most pertinent rules, thus allowing desired derivations and resolving the issue of "UNDECIDED" query results.

Rule prioritization in our system efficiently resolves contradictions and speeds up query processing by preventing unwanted inferences. The priorities are assigned to the two strictly driven defeasible rules. This implies that an explicit priority for strict derivation will always prevail over arguments based on defeasible rules. We regard priority not just as a contradiction-resolution mechanism, but also as a tool to restrict undesired derivations. By blocking lower-priority rules when a higher-priority rule is activated, the system eliminates the need for unnecessary computations that arise from evaluating conflicting rules. This results in a more efficient processing of queries and faster generation of outcomes.

Our DeLP-based GDPR compliance model deals with "UNDECIDED" query outcomes by (i) identifying conflicting arguments and then (ii) assigning priorities to them to resolve the conflict. As shown in Fig 6.2, our proposed mechanism first identifies the conflicting arguments and then prompts the user to explicitly prioritize these identified conflicting arguments. If $arg1$ is assigned priority **'2'** and $arg2$ is assigned priority **'1'**, then $arg2$ will be considered to have a higher priority than $arg1$, and the reasoner will consider it as a supporting argument for re-evaluation to warrant the query. In our proposed system, assigning priorities is practical due to simple rules and automatic conflict identification displayed on the console, allowing user-friendly and efficient priority management. Therefore, by prioritising conflicting arguments, it guarantees that the final query output is either "YES" or "NO" but not "UNDECIDED".

Consequently, the system checks if a priority rule exists, which may rank certain rules (like complaince-based rules) higher than others (like consent-based rules) to automatically resolve the conflict.

### 6.1.4 Supplement Additional Knowledge for Missing Information

Generally, to overcome the "UNDECIDED" query results caused by missing information, our developed modelling tool provides a list of possible facts that can be inserted into the knowledge base for logically deriving a conclusion of "YES" or "NO". The primary aim of supplementing these facts is to efficiently fill the information gaps in the knowledge base. Based on these suggested facts, system modellers can then choose some of the facts which are appropriate for their systems.

The completeness of the information within our DeLP-based GDPR model directly influences the accuracy and reliability of our customized reasoning mechanism. By providing the missing information, the system is capable of generating either a "YES" or "NO" outcome. In addition, our system's ability to identify and handle missing information is essential for mitigating threats of non-compliance. Because the system must first identify the information causing a threat of non-compliance before recommending a potential mitigation measure. The required addition of facts to the knowledge base ensures that all pertinent elements are taken into account when making decisions, hence reducing "UNDECIDED" outcomes.

As illustrated in Figure 6.3, our proposed solution is practical because the handling mecha-

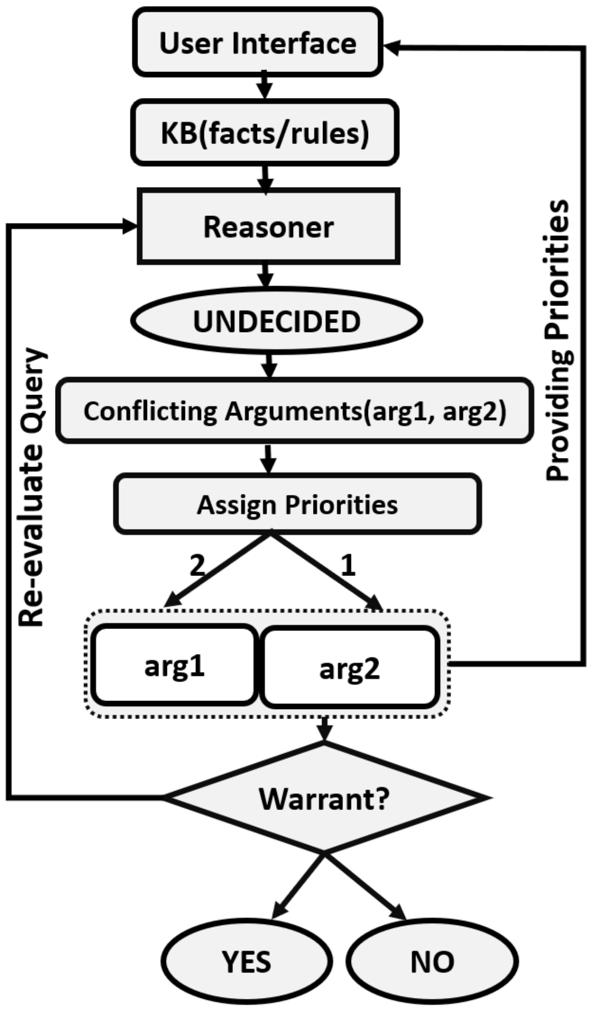


Figure 6.2: Defeasible Reasoning with handling mechanism for Conflicting information.

nism offers an interactive and methodical solution for resolving "UNDECIDED" query results caused by missing information in the proposed DeLP system. If the evaluation of query (q) returns a "UNDECIDED" result, the reasoner will first determine which required facts are missing from the defeasible rules. The user will then be prompted to add the relevant facts to fill any gaps in the knowledge base. The reasoner will re-evaluate the query and determine whether it warrants a query (i.e., "YES" or "NO") after adding the missing facts to the knowledge base. Ultimately, this method systematically resolves "UNDECIDED" outcomes caused by missing information in our system.

In this section, we have provided a critical analysis of the conflicting arguments and the missing information in the knowledge base. In the following section, we will detail the implementation of the conflicting information handler and the missing information handler.

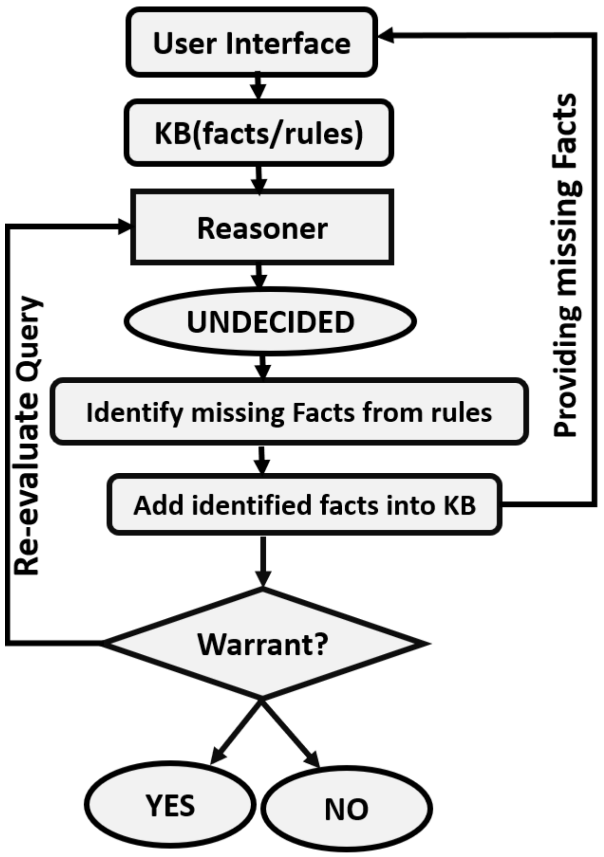


Figure 6.3: Defeasible Reasoning with handling mechanism for Missing information.

## 6.2 Implementation of "UNDECIDED" Results Handlers

We have implemented the "UNDECIDED" handling mechanisms proposed in Section 6.1 and integrated them with the DeLP reasoning mechanism introduced in Section 3.3.2. This task involves two key handlers: (1) the conflicting information handler and (2) the missing information handler. When an "UNDECIDED" result occurs, our reasoner first identifies the underlying cause, which may be the consequence of conflicting arguments or missing facts within the knowledge base. This dual approach ensures the effective resolution of "UNDECIDED" results, thereby enhancing the decision-making process's reliability and comprehensiveness. Thus, by allowing users to intervene, supplement missing information, and prioritise conflicts, we enhance the system's flexibility and reliability while ensuring deterministically accurate outcomes.

### 6.2.1 Conflicting Information Handler

We have implemented the conflicting information handler for the GDPR compliance modelling tool through the application of rule priorities. This handler is crucial in scenarios where contradictory arguments occur during the decision-making process, which could lead to indecision or "UNDECIDED" outcomes.

We implemented Algorithm 6 to resolve ambiguities by identifying conflicting arguments within the DeLP program. Firstly, the algorithm grounds the DeLP program for clarity (line 6) and constructs a stack for managing the argument analysis process (line 7). Each argument generates a dialectical tree, illustrating argument interactions, and is placed on the stack (lines 8-10). The subroutine createDialecticalTree() builds a dialectical tree to evaluate arguments and counterarguments. Algorithm 5 illustrates the process of constructing the dialectical tree. Furthermore, the algorithm processes each tree, identifying "Defeaters" or opposing arguments (line 14). An undefeated defeater marks its argument as conflicting, adding it to the conflict list (lines 16-17), while defeated ones are re-examined. This loop continues until the stack is empty (lines 12-22), ensuring all argument relations are scrutinized. The end result is a detailed list of arguments that have been identified as conflicting within the context of the provided DeLP program (line 23), thus highlighting areas of potential logical contradiction. This function ensures that potential conflicts have been identified in the decision-making process; thus, this step is essential because it establishes the basis for further analysis and resolution of conflict.

Following this, Algorithm 7 processes queries by incorporating explicit priorities assigned to arguments by the user. It begins by determining the argument with the highest priority, based on the lowest numerical value in the provided priorities list (lines 4-8). The algorithm then identifies whether this highest-priority argument aligns with or contradicts the query in question. If the argument's conclusion matches the query (line 12), the algorithm returns "YES" (line 13), indicating compliance. Conversely, if the highest-priority argument opposes the query (line 14), it returns "NO" (line 15), signifying non-compliance. This process ensures that the evaluation of queries is directly influenced by the prioritization of arguments (lines 6-16), enabling a structured approach to resolving queries based on the priority assigned to each argument.

The runtime priority assignment significantly simplifies managing rule-based contradictions and system complexity. Our implementation of applying priorities at runtime offers several advantages. First, it eliminates the need for users to anticipate which rules require prioritization, a task that can be both challenging and unnecessarily burdensome. In a complex knowledge base, determining which set of rules might lead to a contradiction is challenging, as it depends on the given facts and specific queries. Second, this approach offers a simpler implementation compared to assigning priorities directly within the knowledge base. By allowing priority assignment at runtime, the need to modify the parser is eliminated, simplifying the system and reducing the overall complexity.

By identifying conflicts, evaluating their significance, selectively prioritising them, and ultimately basing decisions on the highest-priority arguments, we ensure a logical resolution to potential "UNDECIDED" outcomes. This approach highlights the value of a structured and hierarchical decision-making process within the DeLP-based GDPR knowledge base.

**Alg. 6:** Identifying conflicting arguments

```
1: Input delp (DeLP program), arguments(List of arguments)
2: Output conflictingArguments(List of conflicting arguments)
3: Function FindConflictingArguments($delp$, $arguments$)
4: $conflictingArguments \leftarrow \{\}$
5: $foundSupport \leftarrow$ false
6: $groundedDelp \leftarrow$ ground($delp$)
7: $stack \leftarrow$ createStack()
8: for each $argument$ in $arguments$ do
9:    $tree \leftarrow$ createDialecticalTree($argument$)
10:   $stack.push(tree)$
11: end for
12: while $stack$ is not empty do
13:   $currentTree \leftarrow stack.pop()$
14:   $defeaters \leftarrow currentTree.getDefeaters$
15:   for each $defeater$ in $defeaters$ do
16:      if $defeater.getMarking()$ is UNDEFEATED then
17:         add $defeater.getArgument()$ to $conflictingArguments$
18:      else
19:         $stack.push(defeater)$
20:      end if
21:   end for
22: end while
23: return $conflictingArguments$ =0
```

**Alg. 7:** Re-evaluating query with explicit priorities

```
1: Input delp (DeLP program), priorities(List of arguments with priorities), query(Query
   formula)
2: Output result (YES,NO)
3: Function ReevaluateWithPriority($delp$, $priorities$, $query$)
4: $minPriority \leftarrow \min(priorities.values())$
5: $highPriorityArguments \leftarrow \{\}$
6: for each $entry$ in $priorities$ do
7:    if $entry.value$ is $minPriority$ then
8:       Add $entry.key$ to $highPriorityArguments$
9:    end if
10: end for
11: $highestPriorityArgument \leftarrow highPriorityArguments[0].getConclusion()$
12: if $highestPriorityArgument$ equals $query$ then
13:    return YES
14: else if $highestPriorityArgument$ equals $query.complement$ then
15:    return NO
16: end if=0
```

### 6.2.2 Missing Information Handler

Another mechanism we have implemented to address the "UNDECIDED" query results involves the development of a missing information handler. This mechanism identifies required missing facts and introduces a method for integrating these missing facts into the knowledge base. This step is vital as it ensures the identification of all potentially relevant yet missing facts, thus suggesting system modellers to fill the gaps in the knowledge base to eventually derive a "YES"/"NO" query outcome.

Algorithm 8 provides a comprehensive implementation for identifying missing information in a DeLP with respect to a query. The algorithm begins by gathering all available facts from the DeLP program by iterating over its rules and collecting their conclusions into the $allFacts$ set (lines 6–8). This ensures that the algorithm has a complete set of known facts for further evaluation. Next, the $extractBodyLiterals()$ subroutine is invoked (line 9) to identify relevant literals from the body of the rules associated with the query. This step isolates the conditions required for the query to hold, providing a focused list of conditions to validate against the available facts.

The $extractBodyLiterals()$ function (Algorithm 9) works by iterating through all rules in the DeLP program and extracting the literals from the body (premises) of these rules. For a specified query formula, it retrieves the corresponding rule(s) and collects all their body literals. These literals represent the conditions that need to be satisfied for the query to be valid. The function outputs these conditions as *bodyLiterals*, a set of premises directly relevant to the query.

Once *bodyLiterals* are retrieved, Algorithm 8 compares each literal against the gathered allFacts set to check whether it is missing (lines 10–13). A literal is considered "missing" if it is neither present in $allFacts$ nor negated (i.e., its complement is not present). Any such missing literals are added to the *missingFacts* set, which represents the specific pieces of information absent from the knowledge base and necessary for fully evaluating the query.

The algorithm outputs this concise list of *missingFacts* (line 14), effectively highlighting the information gaps that need to be addressed for the query to hold. This process not only identifies missing knowledge but also isolates the conditions crucial for reasoning and evaluation, ensuring that the DeLP program is complete with respect to the specified query.

Following this, Algorithm 10 provides a dynamic mechanism for updating a DeLP with user-supplied facts to re-evaluate an undecided query. It begins by checking for potential missing facts, identified as $potentialFacts$ (line 4). If there are missing facts, the algorithm invokes Algorithm 11 ($askUserForFacts$), which prompts the user to select the facts they wish to add to the program (line 5). The $askUserForFacts()$ function interacts with the user, presenting a list of missing facts and allowing them to select the relevant ones by their assigned numbers.

The selected facts, returned as $user\_added\_facts$, are then incorporated into the DeLP program, effectively updating the knowledge base (line 6). With the updated program, the algorithm re-evaluates the query to determine its final result: "YES," "NO," or "UNDECIDED" (line 8).

This process ensures that queries previously unresolved due to insufficient information can be dynamically reassessed in light of new evidence. By allowing user interaction and seamlessly integrating the provided facts, Algorithm 10 offers a flexible and adaptive approach to reasoning within a DeLP system.

Algorithm 11 is a subroutine that supports Algorithm 10 by gathering additional knowledge from the user in an interactive manner. It begins by checking if the *missingFacts* set is empty (line 2). If no missing facts are found, the function notifies the user and terminates (lines 3–4). If there are missing facts, it presents a numbered list of these facts to the user for easy reference (lines 6–11). Each fact is displayed with an assigned number, allowing the user to identify and select the relevant facts.

The user is then prompted to input the number corresponding to the fact they wish to add or type "done" to finish (line 12). The algorithm initializes an empty set, *userAddedFacts*, to store the user's selections. In a loop (lines 14–23), the algorithm processes the user's input:

1. If the input is valid (a number corresponding to a fact in the list), the selected fact is retrieved and added to *userAddedFacts* (lines 16–19). A confirmation message is displayed for each added fact.

2. If the input is invalid, the algorithm displays an appropriate error message and prompts the user again (lines 20–22).

When the user indicates they are done by typing "done," the algorithm exits the loop and returns the set of selected facts (line 24). These facts are then integrated into the DeLP program by Algorithm 10 for further reasoning. By combining the interactive fact-gathering of Algorithm 11 with the knowledge-enhancement and re-evaluation capabilities of Algorithm 10, the system achieves a robust, user-centric reasoning framework.

The implementation of such systematic and comprehensive algorithms underscores the significance of interactive and incremental refinement of the knowledge base in our DeLP-based GDPR compliance model, ensuring each query receives a decisive answer backed by supporting data.

After implementing the "UNDECIDED" results handler, our reasoner will yield a result of either "YES" (compliant) or "NO" (non-compliant). For non-compliance (NO), we introduce a strategy for threat mitigation. In the following section, we will elaborate on how we customized the DeLP reasoner to integrate threat mitigation.

## 6.3 Threat Mitigation Integration into Defeasible Reasoning Mechanism

Upon resolving the issue of "UNDECIDED" query results, the reasoning mechanism will eventually return a query output as either compliant ("YES") or non-compliant ("NO"). In the case

**Alg. 8:** Identifying missing information

```
1: Input delp (DeLP program), potentialFacts(List of , query(Query formula)
2: Output filteredFacts(Missing facts)
3: Function identifyMissingFacts($delp$, $query$, $potentialFacts$)
4: $allFacts \leftarrow \{\}$
5: $missingFacts \leftarrow \{\}$
6: for each $formula$ in $delp$ do
7:     Add $formula$ to $allFacts$
8: end for
9: $bodyLiterals \leftarrow$ extractBodyLiterals($delp, query$)
10: for each $literal$ in $bodyLiterals$ do
11:     if $literal$ is not in $allFacts$ and its complement is not in $allFacts$ then
12:         Add $literal$ to $missingFacts$
13:     end if
14: end for=0
```

**Alg. 9:** Extract Body Literals
**Input** delp (a defeasible logic program)
**Output** bodyLiterals (a set of all body literals from the premises of rules in the delp program)

```
1: Function extractBodyLiterals($delp$)
2: $bodyLiterals \leftarrow \emptyset$
3: for all $rule \in delp$ do
4:     $bodyLiterals.addAll(rule.getPremise())$
5: end for
6: return $bodyLiterals$ =0
```

**Alg. 10:** Re-evaluating query with added facts

```
1: Input delp (DeLP program), query(Query formula)
2: Output result (YES,NO,UNDECIDED)
3: Function ReevaluateLogic($delp$, $query$, $potentialFacts$, $filteredFacts$)
4: if $potentialFacts$ is not empty then
5:     $user\_added\_facts \leftarrow$ askUserForFacts()
6:     $delp \leftarrow delp + user\_added\_facts$
7: end if
8: return $query(delp, query)$ =0
```

**Alg. 11:** Ask User for Facts
**Input** missingFacts (a set of potential missing facts)
**Output** userAddedFacts (a set of facts selected by the user)

```
Function askUserForFacts($missingFacts$)
if $missingFacts.isEmpty()$ then
    print "No missing facts identified."
    return $\emptyset$
end if
print "Possible missing facts identified:"
$counter \leftarrow 1$
for all $fact \in missingFacts$ do
    print $counter + "." + fact$
    $counter \leftarrow counter + 1$
end for
print "Enter the number of the fact you wish to add or type 'done' to finish:"
$userAddedFacts \leftarrow \emptyset$
while $userInput \neq "done"$ do
    $userInput \leftarrow$ read user input
    if $userInput$ is a valid number and $1 \leq userInput \leq missingFacts.size()$ then
        $selectedFact \leftarrow$ fact at position $userInput$ in $missingFacts$
        $userAddedFacts.add(selectedFact)$
        print "Fact added: " + $selectedFact$
    else
        print "Invalid input. Please enter a valid number or type 'done'."
    end if
end while
return $userAddedFacts$ =0
```

of non-compliance (i.e., "NO"), we incorporate threat mitigation strategies into the reasoning mechanism to model the system effectively. Providing insights into the system and severity of mitigation measures during instances of non-compliance has significantly enhanced our model's decision-making process. Our proposed solution enables organizations to more effectively manage the threats of non-compliance, ensuring that the measures implemented are both appropriate and efficient.

For example, when a query results in "NO", the reasoner identifies the specific facts and rules that lead to this outcome of non-compliance. During the reasoning process, a dialectical tree is constructed based on the supporting argument (root) to identify its potential defeaters (leaves). If the supporting argument (A) for query (q) encounters potential defeaters (e.g., $\sim$A1 & $\sim$A2), it is marked as defeated (D), as illustrated in Figure 6.4. Because the supporting argument is marked as (D), the reasoner concludes that the query cannot be warranted. Consequently, the reasoner constructs a dialectical tree for the negation of the query (q) to determine if the negation could be warranted. If the supporting argument for the negation of the query gets no defeaters and is marked as undefeated (U), then the query is considered unwarranted, resulting in a "NO" outcome. As a result, the system generates a supporting argument for the negation of the query that illustrates the cause of non-compliance, which is displayed on the console. Based on this supporting argument, our threat mitigation analysis suggests that the system does not meet the specific query's compliance requirements due to the conditions displayed on the console. If the conditions leading to non-compliance, as outlined in the supporting argument, are addressed, then the threats of non-compliance will be mitigated.

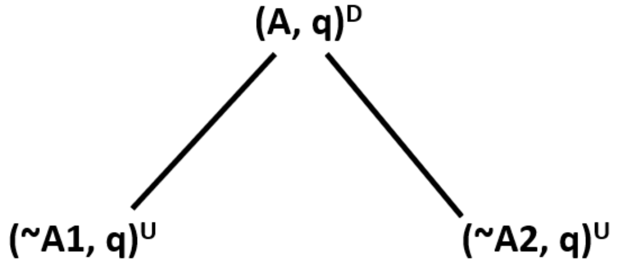


Figure 6.4: Dialectical tree construction: A supporting argument (A) with potential defeaters.

In our proposed DeLP-based GDPR-compliance model, the reasoning process adopts a proactive approach by incorporating threat mitigation strategies for instances of non-compliance. This enhances the system's applicability in real-world scenarios by not only identifying instances of non-compliance but also offering measures to resolve and mitigate these threats. Consequently, with the incorporation of threat mitigation into the reasoning mechanism, our DeLP-based GDPR-compliance model becomes more relevant in real-world scenarios.

## 6.4 Summary

This chapter presents how our GDPR-compliance threat modelling tool addresses "UNDECIDED" query results caused by conflicting rules or missing information. Resolving these uncertainties is crucial, as systems should be classified as either compliant ("YES") or non-compliant ("NO") with GDPR. To address this, we prompt users to provide additional information, such as explicit rule priorities or more detailed system descriptions. We implemented handlers for "UNDECIDED" results to manage conflicting and missing information. For non-compliant systems, we provide insights into reasons for non-compliance and suggest mitigation strategies.

In the next chapter, we will elaborate on the system implementation and demonstrate the experiments for dealing with "UNDECIDED" query results.

# Chapter 7

# System Experiments and Demonstration

This chapter demonstrates how our system identifies and mitigates non-compliance threats within the *TSS* use case. This use case exemplifies the challenges faced by healthcare systems in complying with stringent data protection laws and showcases the effectiveness of our solution in addressing these challenges. First, we demonstrated how non-compliance threats can be inferred from a combined knowledge base of STRIDE (security threats), LINDDUN (privacy threats), and GDPR (requirements and legal obligations) for the *TSS* use case. Second, we illustrated how non-compliance threats can be inferred solely from the DeLP-based GDPR knowledge base for the *Fitbit* use case.

Moreover, this chapter presents experiments conducted to validate the effectiveness of our approach in managing conflicting rules and missing information. These experiments are crucial for demonstrating how our solution handles real-world complexities and enhances decision-making reliability. Specifically, we focus on the use case involving data collection and processing in *Fitbit* wearable devices, where the accuracy and completeness of information are vital for user privacy and compliance with regulations.

Through these demonstrations and experiments, we aim to show the versatility and reliability of our DeLP-based modeling technique in various scenarios, highlighting its potential to improve compliance and decision-making in data-driven applications.

## 7.1 Telehealth Service Use-case Demonstration

We have already discussed the background of the TSS use case in Chapter 4. Here, we will elaborate on how the DeLP-based modeling technique can be applied to demonstrate GDPR compliance in the real-world scenario of the TSS use case. In this demonstration, we conducted experiments to evaluate system performance across various scenarios, using 29 distinct knowledge bases with varying complexities. Each knowledge base ranged from 16 to 128 facts and rules, and we tested multiple queries to assess the computational demands associated with all three possible outcomes: "NO," "YES," and "UNDECIDED." This experimentation included various GDPR principles, such as the right to rectification, accountability, and consent, ensuring comprehensive testing for GDPR compliance. We also incorporated security threat models using STRIDE and privacy threat models using LINDDUN to construct robust knowledge bases that address potential vulnerabilities. Each query was evaluated to determine response time and computational load, providing insights into how efficiently the system handles different types of queries. Further experimental details, including in-depth analysis and performance metrics, are available in Chapter 8.

As mentioned in Section 4.2, STRIDE and LINDDUN have been utilized to identify potential security and privacy threats in *TSS*, revealing that non-compliance threats could arise from various security and privacy issues within the system, including unauthorized data sharing and lack of accountability. In this section, we will demonstrate how we converted STRIDE and LINDDUN security/privacy requirements into a DeLP knowledge base and combine them to infer non-compliance threats to GDPR.

## 7.1.1 Security Threats Obtained from STRIDE

There is a direct connection between security threats and non-compliance threats [191]. The instances of unauthorised data access, integrity breaches, or service disruption, all fall within the domain of both security breaches and non-compliance threats, making STRIDE an apt choice due to its comprehensive coverage. In this project, we use the STRIDE knowledge base to extract the system-related information and translate that information into DeLP. As a practical illustration, we convert a security threat knowledge base (i.e., *information disclosure*) into DeLP, tailored to our specific use case of *TSS*, as depicted below. The converted rules and facts illustrate the information disclosure threat in DeLP, capturing the conditions under which the threat may occur and the exceptions that can prevent it.

```
r1: information_disclosure(I) -< privacy_breach(I),
    data_leak(I).
r2: ~information_disclosure(I) -<
    PreserveConfidentiality(I).
r3: data_leak(I) ← accidental_exposure(I),
    sensitive_information(I).
r4: privacy_breach(I) ← planned_cyberattack(I),
    sensitive_information(I).

f1: sensitive_information(patient_data).
f2: accidental_exposure(patient_data).
f3: planned_cyberattack(patient_data).
```

For instance, there is $information\_disclosure(I)$ threat if both $privacy\_breach(I)$ and $data\_leak(I)$ occur (rule $r1$). The $privacy\_breach(I)$ and $data\_leak(I)$ can be considered as sub-threats or contributing factors that collectively lead to the information disclosure threat.

In rule, $r2$, $\sim information\_disclosure(I)$ indicates that information disclosure is negated if $PreserveConfidentiality(I)$ happens. $PreserveConfidentiality(I)$ represents a condition or action taken to preserve the confidentiality of the information, which serves as a defeater to the information disclosure threat. The other rules ($r3$ and $r4$) are strict rules for `data_leak(I)` and `privacy_breach(I)`. Finally, the facts related to the *TSS* use case are represented with $f1$, $f2$, and $f3$. Based on the above-provided knowledge base, the reasoning mechanism provides the result as `'YES'` for the query `information_disclosure(patient_data)`.

## 7.1.2 Privacy Threats Obtained from LINDDUN

We opt for LINDDUN threat modelling technique due to its close alignment with GDPR's compliance aspects. LINDDUN comprehensively covers privacy and non-compliance threats such as the GDPR legal basis for data processing. We extract LINDDUN's knowledge base from additional threat trees defined in its specification[1]. This knowledge is then translated into DeLP and seamlessly integrated into our system. This approach enables us to effectively tackle privacy concerns and ensure GDPR compliance while keeping data privacy intact.

```
r1: DataDisclosure(X, Y) -< DS(Y), DC(X),
    DP(T), Share(X, T, "data").
r2: ~DataDisclosure(X, Y) ← DS(Y), DC(X),
    DP(T), ~Share(X, T, "data"),
    inform(X, Y, "ShareData").
r3: ~DataDisclosure(X, Y) ← DS(Y), DC(X),
    DP(T), ~Share(X, T, "data"),
    Consent(X, Y, "ShareData").

f1: DS(patient_P).
f2: DC(TSS_Server).
f3: DP(advertiser_T).
f4: Share(TSS_Server, advertiser_T, "data").
f5: ~inform(TSS_Server, patient, "ShareData").
```

For instance, above knowledge base extracted from LINDDUN to reason about Data Disclosure threats is converted into DeLP for the TSS use case. This knowledge base consists of a set of logical rules that define the conditions under which Data Disclosure threat occurs. For instance, $r1$ asserts that there is the threat of `Data Disclosure` if `DC(X)` shares `DS(Y)`'s data with a third-party advertiser (i.e., a DP) [`DP(T)`]. On the other hand, $r2$ and $r3$ indicate the instances where a Data Disclosure threat will not occur when `DC(X)` informs `DS(Y)` about sharing the data with third parties given that the associated consent allows to share data with a third-party. f1-f5 are facts to specify instances and provide system-specific information for the TSS use

[1]`https://downloads.linddun.org/linddun-trees/tree-examples/v20230802/Data Disclosure.pdf`

case. For instance, there is a DS specified as `'patient_P'`, a DC as `'TSS_Server'`, DP as `'advertiser_T'`, and information about sharing data between `DS(Y)` and `DP(T)`. Given these facts and rules, the conclusion to query *q*: `DataDisclosure(TSS_Server, patient_P)` is `'YES'`.

### 7.1.3 Combination of Three Knowledge Bases

The process of combining DeLP knowledge bases from STRIDE and LINDDUN to infer GDPR non-compliance threats follows a well-structured methodology. It begins with semantic alignment to ensure that terminologies match GDPR principles, enabling the mapping of security and privacy threats. These mapped threats are then converted into a DeLP-compatible format, such as rules or facts. The inference is rule-based, utilizing DeLP's deductive reasoning to identify non-compliance threats.

The GDPR non-compliance threats can only be sufficiently identified by harmonising the GDPR principles with security and privacy threats. As already mentioned in Section 5.4 regarding system overview there is a strong relationship between security and privacy threats with non-compliance threats. The STRIDE security threats and LINDDUN privacy threats are translated into DeLP for inferring GDPR threats. Thus the integration of the knowledge obtained from STRIDE and LINDDUN (and is then converted into DeLP) is crucial to the success of our solution.

```
r1: ~Accountability(X, R) -< DC(X),
ReportingMechanism(R), ~Report(X, R, "DataBreach").
r2: ~Accountability(X, R) -< DC(X),
ReportingMechanism(R), ~Report(X, R, "DataBreach"),
information_disclosure(I).
r3: ~Accountability(X, R) -< DC(X),
ReportingMechanism(R), ~Report(X, R, "DataBreach"),
~PreserveConfidentiality(I).
r4: ~Accountability(X, R) -< DC(X), DP(T),
ReportingMechanism(R), ~Report(X, R, "DataBreach"),
Share(X, T, "data").
r5: ~Accountability(X, R) -< DC(X),
ReportingMechanism(R), ~Report(X, R, "DataBreach"),
DataDisclosure(X, Y).
r6: ~Accountability(Y, R) -< DS(Y),
ReportingMechanism(R), ~Complain(Y, R, "DataBreach").
r7: Accountability(X, R) -< DC(X),
ReportingMechanism(R), Report(X, R, "DataBreach").
r8: Accountability(Y, R) -< DS(Y),
ReportingMechanism(R), Complain(Y, R, "DataBreach").
r9: ReportSupervisoryAuthority(R, S) ←
ReportingMechanism(R), Report(R,S, "DataBreach").
```

```
r10: ReportSupervisoryAuthority(R, S) ←
ReportingMechanism(R), Complain(R,S, "DataBreach").

f1: DS(patient_P).
f2: DC(TSS_Server).
f3: DP(advertiser_T).
f4: ReportingMechanism(reportingbody).
f5: ~Report(TSS_Server, reportingbody,
    "DataBreach").
f6: Complain(patient_P, reportingbody, "DataBreach").
f7: Share(TSS_Server, advertiser_T, "data").
f8: inform(TSS_Server, patient, "ShareData").
f9: sensitive_information(patient_data).
f10: accidental_exposure(patient_data).
f11: planned_cyberattack(patient_data).
```

We provide a simple demonstration of how to combine such knowledge by scrutinising the relationships between a non-compliance threat (i.e., non-Accountability), a security threat obtained from STRIDE (i.e., Information Disclosure) and a privacy threat obtained from LINDDUN (i.e., Data Disclosure) as depicted in above combined knowledge base. In this combined knowledge base, the conditions for complying with the GDPR `'Accountability'` principle are determined. `Accountability` and `non-Accountability` requirements (defined by rules $r1$ to $r8$) are inferred based on factors like reported data breaches, complaints of data breaches, Information Disclosure (security threat) and Data Disclosure (privacy threat). Two rules $r9$ and $r10$ address the reporting of incidents to the Supervisory Authority implying that in case of an incident, such as data breaches or complaints, it should be reported to the Supervisory Authority through a `'ReportingMechanism'`.

The knowledge base also contains some facts ($f1 - f11$) for specific entities such as `'DS'`, `'DC'`, and `'ReportingMechanism'`, as well as their interactions and attributes, such as data breach reporting for the default knowledge base. Similarly, for the system-specific system use case (i.e., TSS), the `'patient_P'` and `'TSS_Server'` are also defined in some facts. A query $q$ like `'Accountability (TSS_Server, reportingbody)'` can pass to the reasoning mechanism upon the combined knowledge base to get the answer, in this case, is `'NO'`.

In this section, we presented how non-compliance threats can be inferred from the combined DeLP-based knowledge base of STRIDE and LINDDUN. In the following section, we will demonstrate the *Fitbit* use case to infer non-compliance threats from the DeLP-based knowledge base for GDPR requirements and legal obligations.

## 7.2 Case Study: *Fitbit* Demonstration

In our demonstration, we inferred non-compliance threats from the DeLP-based knowledge base for GDPR. Furthermore, we tackled unresolved queries in our use case (*Fitbit)* through two key experiments: addressing conflicting rules by assigning rule priorities and resolving missing information by identifying and supplementing missing facts. This demonstration highlighted our system's capability to effectively manage "UNDECIDED" query outcomes and recommend threat mitigation measures in instances of non-compliance, thus enhancing the reliability of our GDPR compliance modelling. The implementation data and demonstration results are publicly available on GitHub[2] for reproducing, validating, and further improving the work.

### 7.2.1 *Fitbit* Use-case

*Fitbit* devices are widely recognised for their ability to track users' health and fitness by collecting a variety of personal data, such as location, heart rate, physical activity levels, and sleeping patterns [192]. Once gathered, *Fitbit* platforms process this data, using advanced algorithms to analyse it and provide users with personalized feedback, trends, and health improvement suggestions. This comprehensive process of data collection, storage, management, and analysis necessitates a robust framework to safeguard user privacy and data security.

To protect user data and ensure compliance with GDPR rights, *Fitbit* has implemented several measures[3]. These include allowing users to access, edit, and delete their data, employing data encryption, and conducting security evaluations to safeguard personal information against breaches and unauthorized access [193]. Moreover, the company conducts periodic security audits and implements extra measures to protect confidential information, thus adhering to GDPR's requirement for data security and privacy by design and by default.

However, there are concerns that *Fitbit* may not fully comply with GDPR, raising questions about user consent, right to be informed, right of access, and the sharing of data with third parties[4]. Specific allegations suggest that Fitbit's practices for obtaining consent might not meet GDPR standards, as they may not be freely given, sufficiently explicit, or fully informed. Additionally, the adequacy of data anonymization before sharing of personal data with third parties without explicit user consent[5] have been questioned. Serious privacy concerns and potential GDPR violations have been highlighted due to the collection of highly sensitive personal and potentially identifying data, such as health and menstruation tracking information [194].

Moreover, in the Fitbit use case demonstration, we conducted experiments to evaluate the DeLP-based system's performance across a highly complex scenario involving 1,000 rules and

[2]`https://github.com/nailaazam/Efficient_DeLP_GDPR-Project/tree/master`

[3]https://www.fitbit.com/global/us/legal/privacy-policy

[4]https://cybernews.com/news/fitbit-violates-gdpr/

[5]https://www.cpomagazine.com/data-protection/schrems-continuing-international-data-transfer-crusade-with-gdpr-complaint-against-fitbit/

facts. We tested various queries to gauge the computational complexities associated with the three possible outcomes: "NO," "YES," and "UNDECIDED." This experiment incorporated nearly all GDPR principles listed in Table 2.4, providing a comprehensive assessment of compliance requirements. We evaluated the results for different types of queries by examining both vertical and horizontal complexity, with and without the "UNDECIDED" result handler. This allowed us to analyze how the presence of the result handler impacts computational efficiency. We measured response times to determine the system's performance under these conditions. Further details on experimental setup, metrics, and results can be found in Chapter 8.

### 7.2.2 Knowledge-base Construction for the Use-case

We build a knowledge base specifically for the *Fitbit* use case to illustrate the application of DeLP in modelling GDPR compliance and addressing "UNDECIDED" query results. During this process, for the system-default knowledge base, we translated the GConsent ontology into DeLP using strict and defeasible rules to capture Fitbit's consent as depicted in Appendix 9.4. For the system-specific knowledge base, we utilized relevant facts specific to Fitbit.

The developed DeLP knowledge base example demonstrates the implementation of the *Fitbit* use case to comply with the GDPR's lawful basis of consent under Article 7. Core concepts from the GConsent[6] ontology (Data Subject, Status, Purpose, Personal Data, Processing Personal Data, etc.) are translated into a DeLP knowledge base. This knowledge base contains strict and defeasible rules (r1-r95) and facts (f1-f3). The rules for consent compliance in GDPR are established by Rule r1, which sets the general condition (i.e., GivenConsent), while Rules r2-r9, r12, r22, r25, r34, r43, r48, r53, and r73 address specific conditions for assessment. Rules r2 to r9 detail how consent must be given and validated, including explicit and implicit consent. Rules r10 and r11 cover consent given by guardians for minors. Consent statuses are defined by Rules r12 to r21, detailing valid and invalid statuses. Rules r22 to r24 ensure consent is linked to specific purposes. Validity of consent, such as being freely given and informed, is detailed in Rules r25 to r33. Personal data is specified in Rules r43 to r47. The context of consent is addressed in Rules r48 to r52, while consent for various data processing activities is covered by Rules r53 to r72. Finally, Rules r73 to r84 focus on Fitbit's consent specifics as defined in its privacy policy[7], particularly regarding external data processing and the sharing of sensitive personal data. These rules include conditions for valid consent for external processing (r73), third-party data transfers (r74), ensuring consent is freely given (r75), providing specific purposes (r78), and handling the right to data erasure (r79-r81). They also stipulate conditions for sharing sensitive personal data, such as health information (r84). Moreover, Rules r87 to r89 provide the consent requirements related to informed consent, and Rules r90 to r92 outline

[6] https://openscience.adaptcentre.ie/ontologies/GConsent/docs/ontology#SharingOfPersonalData

[7] https://www.fitbit.com/global/us/legal/privacy-policy#how-info-is-shared

the conditions for the freely given consent obligation. Finally, Rules r93 to r95 specify the requirements related to the right of access.

The reasoner evaluates a query (q): ConsentCompliance(fitbit,user) using the available *Fitbit* specific information (i.e., facts f1 to f3) in the knowledge base. These given facts highlight the compliance challenges that *Fitbit* faces regarding consent, leading to the result=NO (indicating non-compliance) due to DeletionPeriodUpTo90Days (f3). Fitbit's privacy policy states that users can delete their account to exercise their right to erasure, which may take up to 90 days to fully delete personal data (r79-r80). However, users are unable to easily withdraw consent for external data processing instead deleting their account (r82), which results in losing all tracked workouts. This essentially penalizes *Fitbit* users for revoking consent, as they must sacrifice their product experience to do so, which is contrary to GDPR requirements that consent should be easily withdrawable. Based on this scenario, our reasoner processes the query where the supporting argument= <∼ConsentCompliance(fitbit,user) - < ∼ConsentForProcessingPersonalData(fitbit,user).,∼ConsentCompliance(fitbit,user)>. The result ("NO") of the query (q) illustrates that *Fitbit* does not comply with consent lawful basis of data processing.

## 7.2.3 Conflicting Rules Demonstration

We implemented a mechanism to resolve conflicting rules using a *Fitbit* use case. This example focuses on consent requirements related to the informed consent obligation and requirements of exercising right to be informed (Articles 13 & 14) for data transfer, as outlined in the *Fitbit* DeLP knowledge base. Under GDPR Article 7, informed consent requires that data subjects are provided with clear and accessible information about data processing activities, including the purpose, legal basis for processing, and recipients of the data.

The rules (r87-r89) mentioned in Appendix 9.4 demonstrate the scenario of conflicting rules when applying the facts f4: GivenConsent(user, fitbit) and f5: ~ProvideDetailsOfTransfersOfPersonalDataToThirdParties(fitbit, user). Rules r87 to r89 establish a hierarchy of requirements for validating consent for external processing. Rule r87 states that consent for external processing is not valid if the informed consent obligation is not met. Rule r88 specifies that the informed consent obligation includes informing recipients of personal data. Rule r89 further clarifies that informing recipients must include providing details of data transfers to third parties. According to *Fitbit*'s privacy policy, the shared data includes not only a user's email address, date of birth, and gender, but also logs related to food, weight, sleep, water intake, female health tracking, alarms, and messages on discussion boards or to friends on the Services. The policy states that this collected data may be shared with third-party companies, without specifying their locations. Additionally, users are unable to determine which specific data is being shared and what are the implications of sharing such information to third parties. Therefore, on one hand, fact (f4) indicates that the user has given consent to the DC for processing their data. On the other hand, fact (f5) highlights that *Fitbit* does not provide details about the transfer of personal data to third parties. The conflict arises when both rules (r1 & r89) are applicable simultaneously because of the existing facts(f4 & f5), leading to an "UNDECIDED" outcome for the query (q): `ConsentCompliance(fitbit,user)`. Notably, Rule r89 is linked to r88 and r87, while rule r87 is linked to r73, making r73 the main supporting argument. Therefore, to address the issue of "UNDECIDED" query results, we implement a priority mechanism that evaluates the precedence of one rule on the other.

As discussed in Section 6.1.3, our reasoner first identifies the conflicting arguments within the knowledge base and then prompts the user to assign explicit priorities to these conflicting arguments for further processing. For example, consider the following structure of arguments with assigned priorities: if the user assigns a higher priority (i.e., **1**) to arg2 and a lower priority (i.e., **2**) to arg1, this indicates that the consent requirement for processing personal data takes precedence over the given consent.

**2** `arg1 = <{ConsentCompliance(fitbit,user) -< GivenConsent(user,fitbit),DataSubject(user), DataController(fitbit)}, ConsentCompliance(fitbit,user)>`

**1** `arg2 = <{~ConsentCompliance(fitbit,user) -< ~ConsentForProcessingPersonalData(fitbit,user)}, ~ConsentCompliance(fitbit,user)>`

Ultimately, our system resolves the conflict in favor of arg2 (r73), which prioritizes the consent requirement for external processing over the user's given consent. After re-evaluation, the reasoner responds to the query (q) with "NO", indicating non-compliance with the consent requirements.

### 7.2.4 Missing Information Demonstration

We implemented a mechanism to identify missing information in the *Fitbit* use case. For demonstration purposes, we refer to the consent requirements of data transfer under Article 44. Due to uncertainties or incompleteness in the data, *Fitbit*'s data processing operations might not fully comply with the consent requirements.

The rules (r90-r92) mentioned in Appendix 9.4 demonstrate the example of missing information when applying the facts: f6: ThirdParty(processor), f7: ProcessingForDataTransfer(fitbit, processor), and f8: DeletePersonalInformation(fitbit, user). Rules r90 and r92 state that there will be consent compliance when the requirements for deleting personal information by exercising the right to erasure, freely given consent obligation, and processing for data transfer are satisfied. Rule r91 describes a counter-scenario where consent compliance is not fulfilled if the user is not allowed to delete their personal data and the requirements for processing data transfers are not met. Additionally, facts f7 and f8 demonstrate that *Fitbit* processing for data transfer and allows users to delete their account if they do not want their personal data to be transferred.

After applying these rules to the provided facts, the reasoner yields an "UNDECIDED" outcome regarding Fitbit's compliance with the query `ConsentCompliance(fitbit, user)`. This uncertainty arises because the knowledge base does not specify whether *Fitbit* fulfilled the freely given consent obligation. *Fitbit*[8] does not provide the required opt-out option for users who do not want to transfer their data, effectively enforcing a 'take it or leave it' approach. To resolve this, our system identifies the need for additional information to fully assess compliance with consent requirements. The reasoner determines that the freely given consent for data processing, as outlined in Rule r90 and Rule r92, is missing from the provided facts.

Upon identifying this gap, the system prompts for the addition of relevant facts, such as `FreelyGivenConsentObligation`, to allow users the option to consent to data transfer. For instance, if users are provided with freely given consent option, they may choose the option of not to allow data transfer while still using the app for health and fitness tracking purposes. This information is then added to the knowledge base as a new fact (f9):
`FreelyGivenConsentObligation(fitbit, user).`

With the addition of the previously missing fact (f9), our system re-evaluates Fitbit's data processing activities against the consent compliance requirements. Considering the facts such as `ProcessingForDataTransfer` (f7), `DeletePersonalInformation` (f8), and now the `FreelyGivenConsentObligation` (f9), the system can conclude that *Fitbit* is compliant with the consent requirements. As a result, our reasoner returns "YES" for the query `ConsentCompliance(fitbit, user).`

In the following section, we will demonstrate how we mitigated non-compliance threats in our GDPR threat model after resolving the issue of "UNDECIDED" query results.

### 7.2.5 Threat Mitigation Demonstration

We address non-compliance threat mitigation by offering insights into the system. This helps explain the reasons for non-compliance and potential mitigation strategies, particularly for the *Fitbit* use case. We focus on the requirements of the right of access (i.e., Article 15) to illustrate this process.

The rules (r93-r95) mentioned in Appendix 9.4 illustrate the example of threat mitigation by applying the fact f10: ~GivesRightToObtainCopyOfPersonalData(fitbit, user). For this part of knowledge base scenario, query (q):
`ConsentForExternalProcessing(fitbit,user)` is decided unwarranted, resulting in a "NO" outcome (i.e., non-compliance). This is due to the facts within the knowledge base (i.e., f10), which serve as the basis for the Reasoner to process the query. In the reasoning algorithm, after receiving a query (q) and completing the grounding step, the reasoner identifies the augments with the conclusion: `ConsentForExternalProcessing(fitbit,user)` from

[8]`https://techcrunch.com/2023/08/30/fitbit-gdpr-data-transfer-complaints-noyb/`

the rule set and iterates until the supporting argument is found (i.e., (r93). The reasoner constructs the dialectical tree based on the supporting argument once it has been identified. The reasoner identifies a potential defeater (r94) for the supporting argument (r93), hence it is marked as defeated (D). Since the supporting argument is marked as (D), the reasoner concludes that the query cannot be warranted. Therefore, the reasoner constructs a dialectical tree for the supporting argument (i.e., (r94)) of the negation of the query (i.e., `∼ConsentForExternalProcessing(fitbit,user)`). The reasoner does not identify any defeaters for the supporting argument (r94) of the negation of the query and thus it is marked as Undefeated (U). Rule r95 is linked to and specifies Rule r94. As the supporting argument (r94) of the negation of the (q) is marked as Undefeated (U), the query is deemed Unwarranted. Based on this process, the query result is returned as "NO".

The system generates a supporting argument (i.e., the negation of the query) that illustrates the cause of non-compliance, displayed on the console as:

```
<{∼ConsentForExternalProcessing(X, Y) ← ∼RightForAccess(X, Y)},
∼ConsentForExternalProcessing(X, Y)>
```

This output is crucial because it highlights a case of non-compliance within the system: Fitbit does not provide access to personal data used for external processing. According to Fitbit's privacy policy[9], the company shares sensitive user data, including health-related data, with third-party companies whose locations are unspecified. Furthermore, users cannot determine which specific categories of personal data are shared, and despite exercising their right of access to obtain a copy of this information from *Fitbit*[10], the complainants never received a response. Consequently, our threat mitigation analysis concludes that the system fails to comply with the Right of Access principle. This non-compliance is due to *Fitbit*'s inability to provide user access to information, which is vital for ensuring transparency in processing. Allowing users to exercise their right of access would let them know where their information is being transferred, including to third countries.

## 7.3 Summary

This chapter demonstrated how our system identifies and mitigates non-compliance threats within the *TSS* use case, addressing the challenges healthcare systems face in complying with stringent data protection laws. We showed how non-compliance threats can be inferred from a combined knowledge base of STRIDE (security threats), LINDDUN (privacy threats), and GDPR (requirements and legal obligations). Additionally, we illustrated how non-compliance threats can be inferred solely from the DeLP-based GDPR knowledge base in the *Fitbit* use case. Moreover, we conducted experiments to validate the effectiveness of our approach in managing

[9] `https://www.fitbit.com/global/us/legal/privacy-policy#info-we-collect`
[10] https://techcrunch.com/2023/08/30/fitbit-gdpr-data-transfer-complaints-noyb/

conflicting rules and missing information. These experiments are crucial for demonstrating how our solution handles real-world complexities and enhances decision-making reliability. Specifically, we focused on the *Fitbit* use case, where the accuracy and completeness of information are vital for user privacy and regulatory compliance.

In the next chapter, we will present the results of experiments conducted for various query outcomes and provide a comparative analysis.

# Chapter 8

# Results, Analysis and Discussion

In this chapter, detailed experimental results obtained by applying the novel threat modelling techniques to the *TSS* and *Fitbit* use cases are provided. This chapter offers a comprehensive analysis and discussion aimed at demonstrating the effectiveness of the proposed solutions in addressing GDPR compliance within these applications.

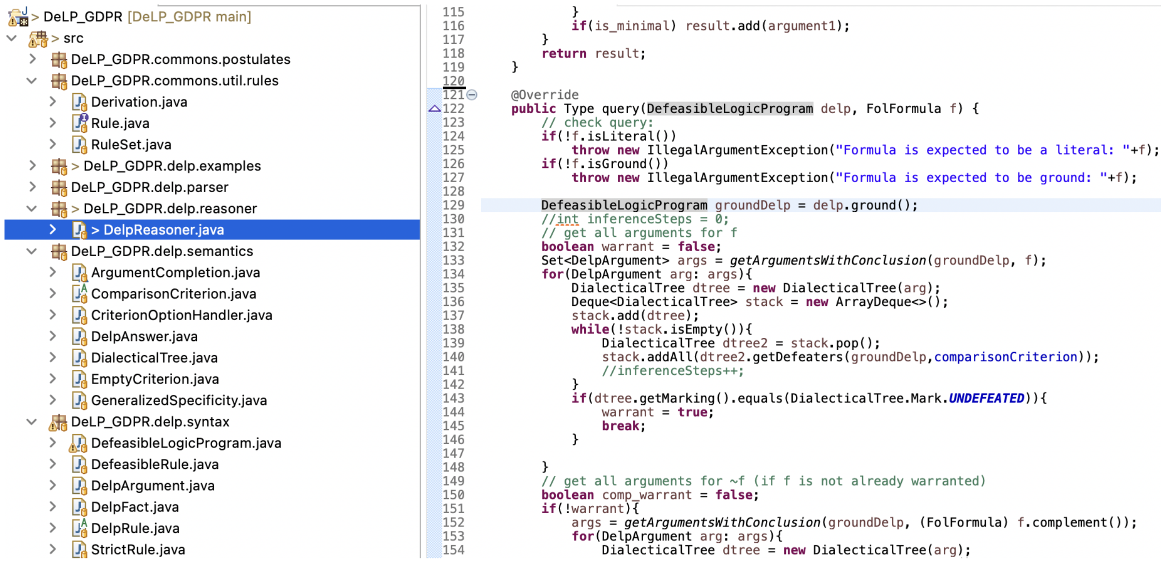
Initially, the chapter demonstrates the developed DeLP-GDPR Project for the *TSS* use case and provides detailed performance results by comparing the execution time and complexity of the knowledge base for different queries (i.e., "YES," "NO," and "UNDECIDED"). The chapter further offers a comparative analysis and extensive discussion of each query result.

The chapter then presents the detailed results and analysis of the modelling technique for GDPR compliance in the *Fitbit* use case, evaluating the "UNDECIDED" results handlers. These handlers are crucial for managing cases where the system cannot make a clear decision due to conflicting rules or missing information. The discussion includes how these handlers operate within the system and assess their impact on overall performance. This involves a detailed examination of system behaviour under varying levels of horizontal and vertical complexities, illustrating how the system scales and maintains performance despite increased complexity and potential data ambiguities.

Through these analyses, the chapter aims to validate the robustness and adaptability of the GDPR compliance modelling techniques, demonstrating their practical applicability to real-world scenarios in the healthcare and wearable technology sectors.

## 8.1 Results and Analysis of DeLP-based Modelling for GDPR in Telehealth Services

This section presents the findings from the demonstration for the proposed modelling technique tailoring to the *TSS* use case including performance results and insightful discussion.



```
> DeLP_GDPR [DeLP_GDPR main]
  > src
    > DeLP_GDPR.commons.postulates
    DeLP_GDPR.commons.util.rules
      > Derivation.java
      > Rule.java
      > RuleSet.java
    > > DeLP_GDPR.delp.examples
    > DeLP_GDPR.delp.parser
    > DeLP_GDPR.delp.reasoner
      > > DelpReasoner.java
    DeLP_GDPR.delp.semantics
      > ArgumentCompletion.java
      > ComparisonCriterion.java
      > CriterionOptionHandler.java
      > DelpAnswer.java
      > DialecticalTree.java
      > EmptyCriterion.java
      > GeneralizedSpecificity.java
    DeLP_GDPR.delp.syntax
      > DefeasibleLogicProgram.java
      > DefeasibleRule.java
      > DelpArgument.java
      > DelpFact.java
      > DelpRule.java
      > StrictRule.java
```

```java
            }
            if(is_minimal) result.add(argument1);
        }
        return result;
    }

    @Override
    public Type query(DefeasibleLogicProgram delp, FolFormula f) {
        // check query:
        if(!f.isLiteral())
            throw new IllegalArgumentException("Formula is expected to be a literal: "+f);
        if(!f.isGround())
            throw new IllegalArgumentException("Formula is expected to be ground: "+f);

        DefeasibleLogicProgram groundDelp = delp.ground();
        //int inferenceSteps = 0;
        // get all arguments for f
        boolean warrant = false;
        Set<DelpArgument> args = getArgumentsWithConclusion(groundDelp, f);
        for(DelpArgument arg: args){
            DialecticalTree dtree = new DialecticalTree(arg);
            Deque<DialecticalTree> stack = new ArrayDeque<>();
            stack.add(dtree);
            while(!stack.isEmpty()){
                DialecticalTree dtree2 = stack.pop();
                stack.addAll(dtree2.getDefeaters(groundDelp,comparisonCriterion));
                //inferenceSteps++;
            }
            if(dtree.getMarking().equals(DialecticalTree.Mark.UNDEFEATED)){
                warrant = true;
                break;
            }

        }
        // get all arguments for ~f (if f is not already warranted)
        boolean comp_warrant = false;
        if(!warrant){
            args = getArgumentsWithConclusion(groundDelp, (FolFormula) f.complement());
            for(DelpArgument arg: args){
                DialecticalTree dtree = new DialecticalTree(arg);
```

Figure 8.1: Implementation of the DeLP-GDPR project in JAVA showing packages structure and a DeLP reasoner

### 8.1.1 Demonstration and Experiment

For the demonstration, we have built a DeLP-GDPR compliance knowledge base for the *TSS* use case. The designed reasoning algorithm performs the inference for the specific query on the developed knowledge base (i.e., system-default knowledge base and system-specific knowledge base). By passing the queries to the inference algorithm as input the reasoner yields outcomes that are displayed either in a console or written in an output file. The implementation with data and demonstration results are publicly available on GitHub[1] for reproducing and validating the work and further improvement.

Fig. 8.1 shows the implementation of the `DeLP-GDPR Project` in JAVA with some packages shown in the left panel and the DeLP reasoner in the right panel. We have performed the DeLP_GDPR project for different types of queries with "YES"/"NO"/"UNDECIDED" answers over different knowledge bases taking complexity (i.e., total number of facts and rules in the knowledge bases) into consideration.

To evaluate the practical implications and efficiency of our proposed solution, we carry out experiments to measure the execution time of different types of queries over varying complexities of knowledge bases. In this demonstration, we ran experiments on a 64-bit Windows 11 Pro OS with an x64-based Intel(R) Core(TM)-i7 10th Gen processor running at 1.80GHz, equipped with 32GB RAM. We have run the experiment for all three possible results over 29 knowledge bases, ranging from 16 to 128 total numbers of facts and rules with different queries to gauge the computational complexities of three outcome results (i.e., "NO", "YES", and "UNDECIDED"). For each combination of query and knowledge base, we have run 100 times the same experiments to get the average as well as the standard deviation. In the next section, the experimental

[1]`https://github.com/nguyentb/DeLP_GDPR/tree/main`

results will be analysed and discussed in detail.

### 8.1.2 Performance Results

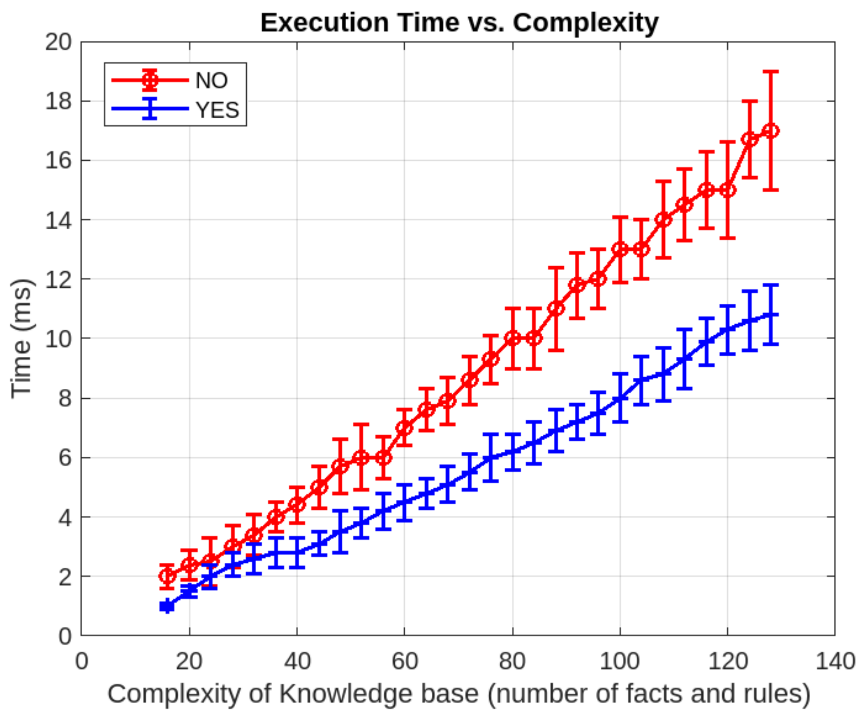


Figure 8.2: Comparative Analysis of Execution Time for DeLP-based GDPR Compliance: NO vs. YES

Fig. 8.2 illustrates the execution time versus the complexity of the knowledge base for queries with "NO" or "YES" answers. Specifically, the knowledge bases are designed for the *TSS* use case and the queries are tailored to assess the compliance of a TSS_Server (i.e., DC) with the Consent legal basis granted by various patients (i.e., DS). For instance, query $q1$: `ConsentCompliance(TSS_Server,patient1)` and query $q2$: `ConsentCompliance(TSS_Server,patient2)` always yield answers of "NO" and "YES" for all 27 knowledge bases, respectively.

It is evident that the execution time increases as the complexity of knowledge bases increases, which adheres to a linear pattern. It only takes less than $18ms$ for both $YES/NO$ queries with knowledge bases up to 128 no of rules and facts. The standard deviation is reasonable showing acceptable variability for the experimental results, implying that the reasoner works stably. It is worth noting these are expected results supported by previous studies [195, 196]. Due to the way the reasoning algorithm is implemented in our project (i.e., the reasoner searches for the warrantability of a query $q$ before searching the warrantability of its negation form (i.e., $q.complement$)), generally, the execution time for "NO" answer queries is higher than "YES" ones, as clearly depicted in Fig. 8.2.

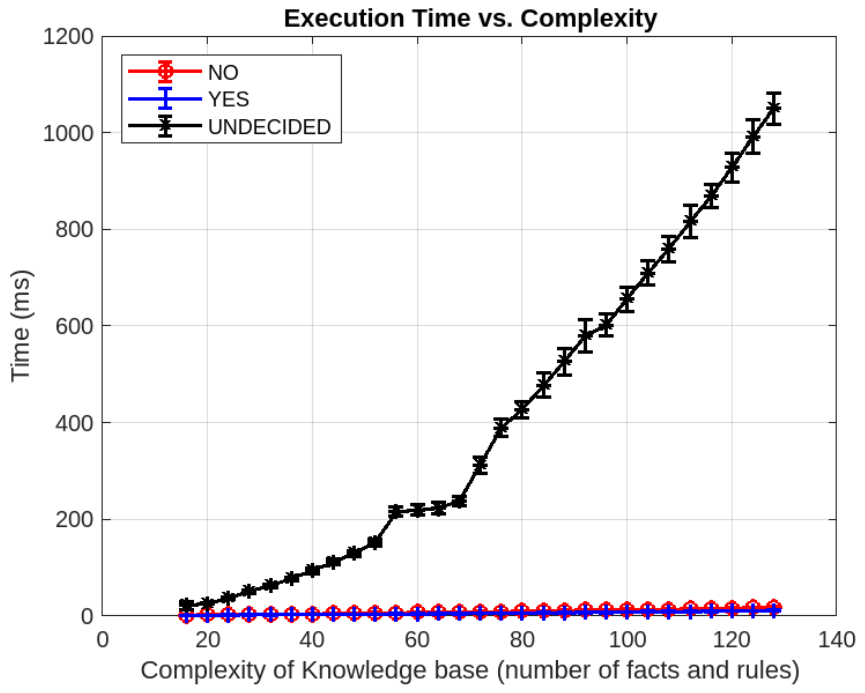


Figure 8.3: Comparative Analysis of Execution Time for Different DeLP-based GDPR Compliance Response Types

Conversely, the answer to the query $q3$: `ConsentCompliance(TSS_Server, patient3)` is "UNDECIDED" due to the absence of explicit rules and facts that substantiate the query. The provided rules and facts in the knowledge bases for $q3$ do not provide sufficient grounds for the reasoner to warrant the query. To confirm the answer, the reasoner has to go through all combinations of grounded facts and rules in the knowledge base, similar to a *brute-force* search. As a consequence, the elapsed time for this type of query is up to 100 times higher than that of "NO" and "YES" but still linear in nature, for instance, it takes 1050*ms* in knowledge bases for the 128 total of facts and rules. Fig. 8.3 visually represents the notably extended execution time for query $q3$ compared to $q1$ and $q2$ nicely, obviously showing that "YES"/"NO" queries are executed much faster compared to "UNDECIDED" ones.

In the upcoming section, we will delve deeper into our analysis and discussion of the obtained results.

### 8.1.3 Analysis and Discussion

For the analysis, we delve into the execution time versus the knowledge base complexity obtained from the experiments taking into account our implementation of the algorithms, particularly the reasoning mechanism, in the `DeLP-GDPR Project` as well as the defeasible theory in DeLP and Argumentation reasoning.

Upon scrutinising "YES" responses, intriguing patterns emerge concerning execution time for different complexity levels. The execution time generally increases with higher complexity.

Moreover, the increase is uniform, as observed from the execution time values. This observation indicates that simplifying the rules leads to a uniform increase in the execution time. This efficiency can also be attributed to the type of query. Furthermore, the relatively higher standard deviations accompanying execution time values signify response variability within the system.

In comparison with "YES" queries, the execution time for "NO" ones is generally higher. This disparity highlights the computational demands inherent in assessing negative outcomes, again, due to the way we implement the reasoning algorithm that looks for the warrantability of a query before its negation form. Additionally, the standard deviations for "YES" responses are typically on par with those of "NO" responses. This implies a certain consistency in the system's behaviour across query types. This linear complexity in the "YES"/"NO" queries is attributed to the utilisation of propositional defeasible theory that the consequences of a defeasible theory can be computed in $\mathcal{O}(N)$ time, where $N$ is the complexity of theory (i.e., the number of symbols). This has been investigated by previous research works and the inference problem within propositional defeasible logic has been demonstrated to exhibit linear complexity [135, 196].

The analysis of "UNDECIDED" queries offers valuable insights into scenarios where the knowledge base system encounters ambiguities or conflicting information. The execution time for "UNDECIDED" queries is notably higher compared to both "NO" and "YES" ones. To successfully confirm an "UNDECIDED" query, the algorithm reckons all combinations of grounded facts and rules in the knowledge base to construct dialectical trees. Thus, this prolonged execution time can be attributed to the system's attempt to resolve uncertainties by considering multiple pathways and potential resolutions. Nevertheless, the behaviour of the algorithm is *linear* to the size of the KBs, making it still suitable for a large-scale expert system. As the system uses more resources attempting to reach definitive conclusions, it signifies the growing resource-intensity of resolving uncertainties. Thus, the elevated standard deviations accompanying "UNDECIDED" responses underscore the intricate nature of managing uncertain scenarios. Gaining a nuanced understanding of "UNDECIDED" response patterns informs enhancements in the system's handling of ambiguity.

From this system performance analysis, with the linear complexity for all types of queries, the proposed approach offers promising solutions for a variety of real-world services and applications, particularly when gathered information for decision-making is incomplete and/or conflicted. In the case of an "UNDECIDED" query, the analysis also suggests a nice trick to stop performing the reasoning algorithm by introducing an execution time upper bound $\theta$ for each knowledge base. After $\theta$, which implies that there is a very high probability the response is not "YES" and "NO", the reasoning algorithm can stop and yield the result of "UNDECIDED".

In the next section, we will provide the results and analysis of the "UNDECIDED" results handlers for various queries in the *Fitbit* use case.

## 8.2 Results and Analysis of DeLP-based Modelling for GDPR in Fitbit Use-case

In this section, we demonstrate the system's performance across different scenarios, including scenarios with and without the "UNDECIDED" results handlers in various complexities of the knowledge base. We then analyse the experimental results and provide an insightful discussion on the complexity of the knowledge base and the effectiveness of the "UNDECIDED" results handlers.

### 8.2.1 Experimental Settings

Empirical experiments are carried out to evaluate the impact of incorporating Conflicting and Missing Information Handlers on query execution times. These tests take place on a 64-bit Windows 11 operating system, utilizing an Intel(R) Core(TM) i9-10900KF CPU at 3.7GHz with 32 GB of RAM. Knowledge bases designed to produce three possible outcomes ("YES", "NO", and "UNDECIDED") are created. "UNDECIDED" outcomes are further divided into two categories: one due to conflicting information and the other due to missing information.

We assess the complexity of a knowledge base through two metrics: vertical and horizontal complexity, as detailed in Section 5.4.2. In the experiments, vertical complexity is adjusted in all scenarios except for the "UNDECIDED-missing-information" case. This exception is because the "UNDECIDED" results handlers only intervene when a rule with horizontal complexity greater than 1 indicates missing information.

The experiments are conducted using identical knowledge bases across scenarios of similar complexity to ensure a fair comparison. Since our "UNDECIDED" results handlers require human input, the algorithm's execution time is measured by recording the total query execution time and then subtracting the time spent awaiting human input. The error bars in all graphs within this paper represent the standard deviation, calculated from the average execution times of 30 runs for each data point.

#### Comparative Analysis

In this section, we discuss our experimental results, showcasing system performance comparisons both with and without the "UNDECIDED" results handlers. We have categorized the query outcomes for better clarity in the presentation of results.

Figure 8.4 shows a increase in query execution times as the complexity of the knowledge base increases. The complexity on the x-axis represents the "Vertical or Horizontal Complexity of the Knowledge Base (facts & rules)," respectively. Implementing "UNDECIDED" results handlers causes a slight and consistent rise in execution times for both "YES" and "NO" outcomes, yet the increase is moderate across all levels of complexity, indicating that the handlers'

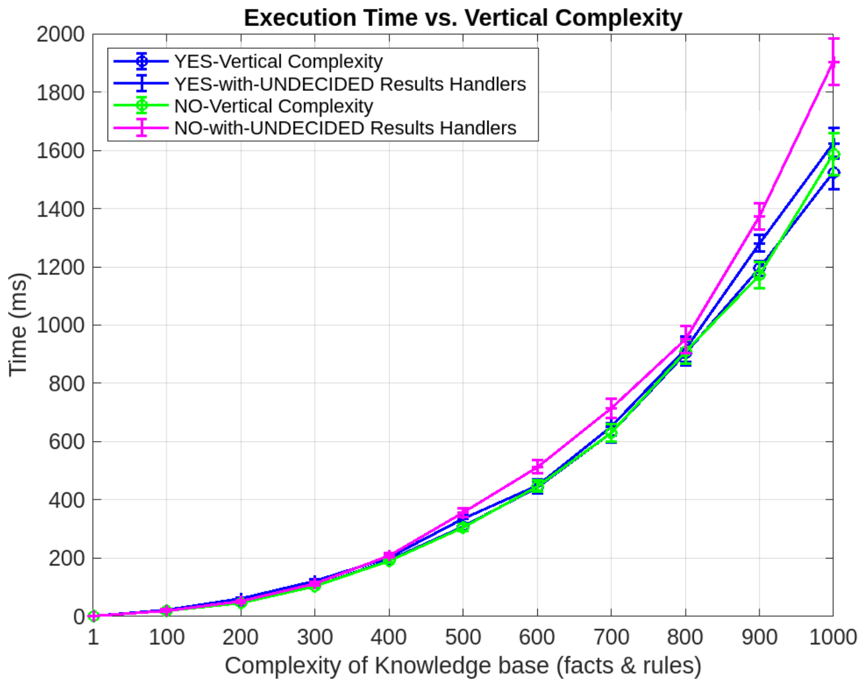


Figure 8.4: Execution Time vs. Vertical Complexity for YES/NO results.

impact on performance is minimal and well-managed. Queries are processed within 2000 milliseconds, even at the highest tested complexity of 1000 rules and facts. This suggests that the introduction of "UNDECIDED" results handlers does not significantly affect the performance for "YES" and "NO" outcomes. The system's stability is evidenced by reasonable standard deviations, suggesting acceptable variability in experimental outcomes. As previously mentioned, our findings align with those of earlier research [195, 196]. Similarly, In Figure. 8.5, in cases of high horizontal complexity, factorial time complexity dominates. "NO" responses generally take longer than "YES" because the reasoner verifies the query's truth before its negation.

In contrast, Figure 8.6 demonstrates that "UNDECIDED" outcomes, resulting from conflicting information, show a significant increase in execution time as the complexity of the knowledge base increases with the use of a conflicting information handler. The performance approximately doubles in time, likely due to the necessity to re-evaluate the query and the associated increase in processing overhead. Initially, execution times with the conflicting information handler align with those for "YES" or "NO" outcomes. However, as complexity increases, the time required to resolve conflicting information with the handler escalates sharply. This suggests that at higher complexities, the execution time for resolving conflicts may exceed that for straightforward "YES" or "NO" outcomes, with the increase becoming more pronounced as the knowledge base complexity grows. The small error bars on the graph indicate a low standard deviation, confirming that the execution times are consistently near the average, thereby showcasing the system's reliable performance across various complexity levels.

Likewise, Figure 8.7 demonstrates a substantial rise in execution times as horizontal complexity in the knowledge base increases (as cen be seen on x-axis ), particularly when the missing

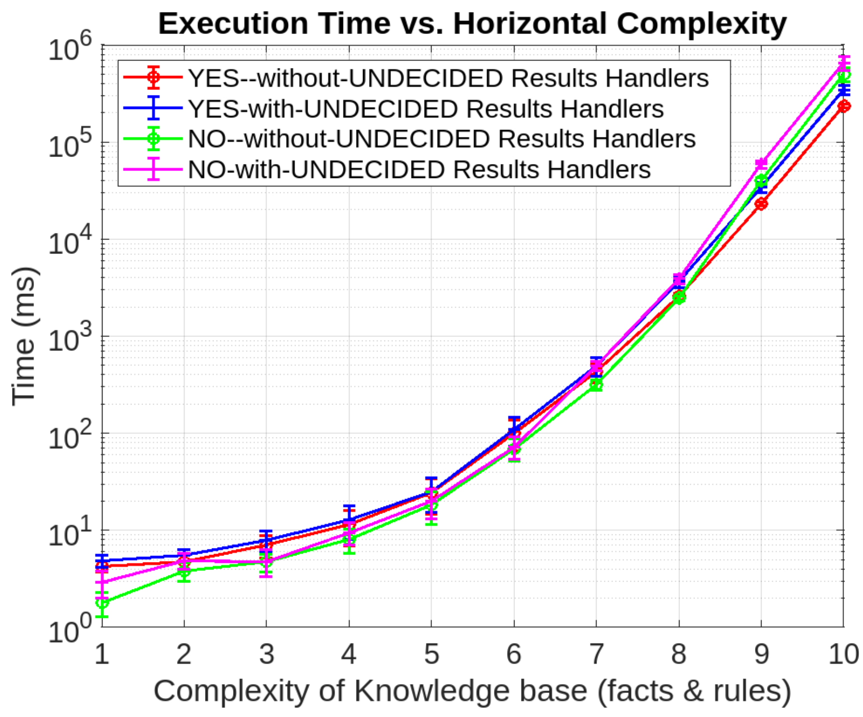


Figure 8.5: Execution Time vs. Horizontal Complexity for YES/NO results.

information handler is active. The graph, displayed on a logarithmic scale, illustrates a significant rise in execution time, escalating from milliseconds to hours as the complexity multiplies by roughly a factor of 10. The graph's leftward shift is due to re-evaluating queries when additional facts are included. This rise is steeper than the execution times for "YES" or "NO" outcomes, indicating that processing missing information takes considerably more time. Furthermore, the small error bars on the graph demonstrate that query execution times are consistent across different levels of knowledge base complexity, showcasing the system's reliability in managing "UNDECIDED" queries due to missing information.

In the next section, we will thoroughly analyze and discuss the results we have obtained.

### 8.2.2 Analysis and Discussion

When the query results in "YES" or "NO," the execution times are comparable with or without "UNDECIDED" results handlers, as the system experiences minimal alterations. The "UNDECIDED" results handlers are triggered only if the reasoning process initially yields an "UNDECIDED" outcome. Any slight increase in execution time is due to the added overhead involved in handling an "UNDECIDED" result.

When dealing with conflicting information, execution time doubles as the system requires two evaluations: initially, when the query yields an "UNDECIDED" result and conflicting arguments are identified increase time from the dialectical tree; and secondly, after the user prioritizes the conflicting rules for re-evaluation. Although this process extends the execution time, its increase in time complexity ensures that the increased duration remains manageable.

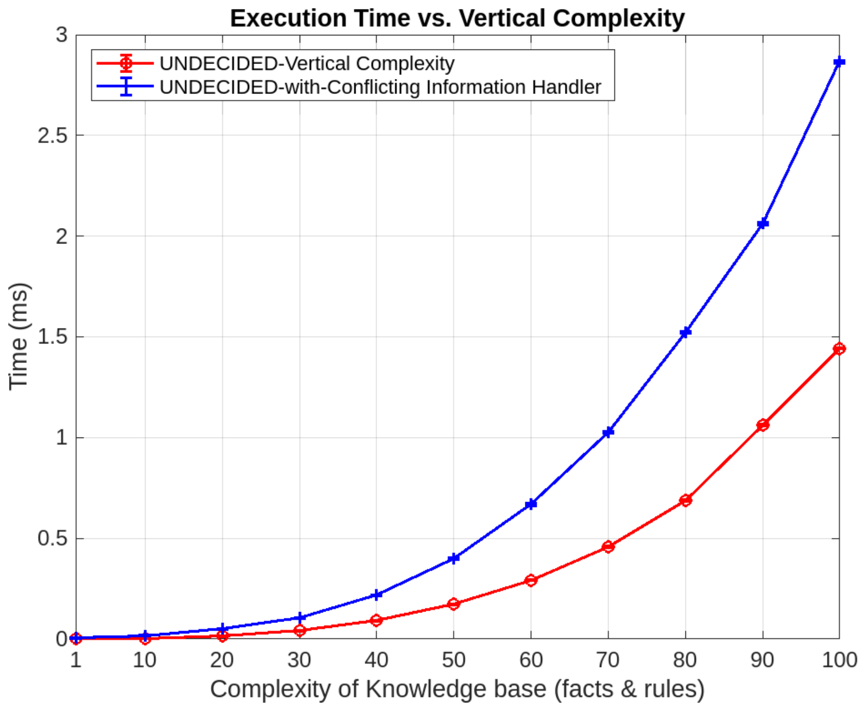


Figure 8.6: Performance comparison of queries resulting in "UNDECIDED-Conflicting Information".

However, significant performance disparities are observed during demonstrations involving missing information. Initially, without the missing information handler, the relationship between complexity and execution time increases at low horizontal complexity but becomes factorial at high horizontal complexity. With the missing information handler activated, execution times increase in both low and high-horizontal complexity scenarios. At lower horizontal complexities, the primary cause of increased execution time is the overhead associated with the handler. At higher complexities, the main factor driving increased execution time is the need for query re-evaluation; in our tests, this was effectively demonstrated by increasing the horizontal complexity by one level. This observation is supported by a one-unit leftward shift in horizontal complexity shown in Figure 8.7, indicating that detecting and re-evaluating queries due to missing information takes roughly the same time as when no information is missing.

The minimal error bars in Figures 8.4, 8.6, and 8.7 reflect low standard deviation, indicating the consistent performance of our system for "YES", "NO", and "UNDECIDED" queries across different complexity levels. Given the deterministic behaviour of our reasoner and the "UNDECIDED" results handlers, variations in execution times are primarily attributed to hardware differences.

In the above, we have analyzed and discussed our four-fold contributions: firstly, the construction of a Defeasible Logic knowledge base that seamlessly integrates facts and rules derived from GDPR requirements with existing security and privacy knowledge; secondly, the development of a sophisticated inference engine designed to discern GDPR non-compliance

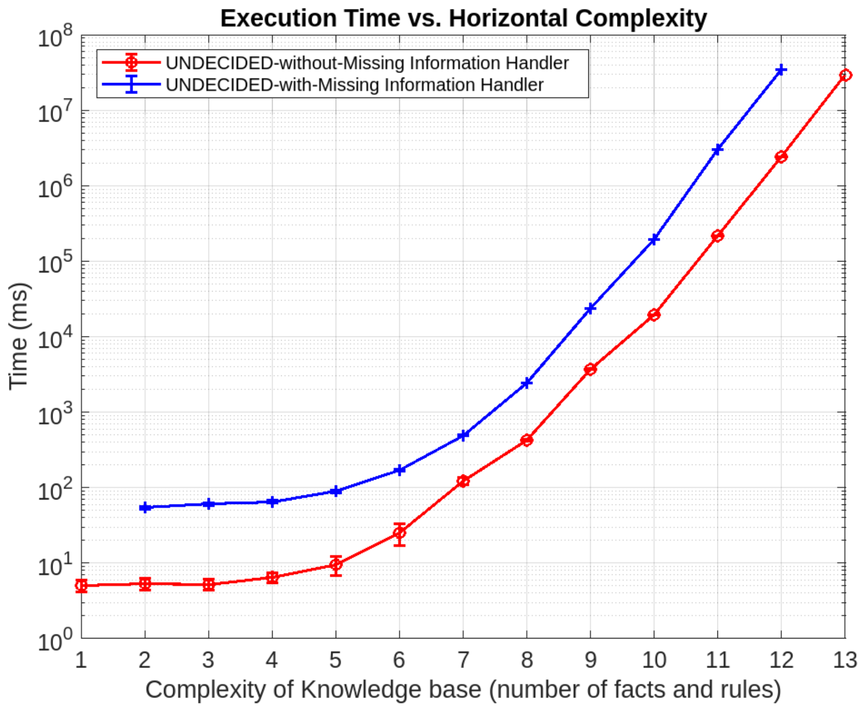


Figure 8.7: Performance comparison of queries resulting in "UNDECIDED-Missing Information".

threats within this knowledge base; thirdly, the creation of the "UNDECIDED" results handler to address uncertainties; and fourthly, the integration of threat mitigation in the reasoner to suggest strategies for mitigating threats. Finally, we provided a compelling demonstration of the proposed solution's feasibility and performance efficiency through the Telehealth Services and Fitbit use cases. This holistic approach not only advances the field of GDPR compliance but also exemplifies the practical applicability and robustness of the devised modeling technique.

Our findings/results and use case experiments successfully achieved our research objectives. We developed a novel modeling technique that effectively identifies and models GDPR non-compliance threats, demonstrating scalability and adaptability across different data-driven contexts. The technique addresses limitations of current models by handling incomplete and conflicting information, thereby improving applicability in real-world scenarios. Additionally, our approach enhances compliance risk management, providing a robust framework for identifying and mitigating risks as regulatory requirements evolve. Finally, we validated the technique's practical applicability by demonstrating its effectiveness in complex, real-world systems involving sensitive personal data.

## 8.3 Summary

In this chapter, detailed experimental results from applying the novel threat modelling techniques to the *TSS* and *Fitbit* use cases are presented. The chapter demonstrates the effectiveness of the proposed solutions in addressing GDPR compliance. It includes a comparative analysis of different query results ("YES," "NO," and "UNDECIDED") for the *TSS* use case, highlighting the performance in terms of execution time and complexity. For the *Fitbit* use case, the chapter evaluates the "UNDECIDED" results handlers, essential for managing conflicting rules and missing information. This discussion covers system behaviour under varying complexities and illustrates how the system scales and maintains performance.

The next chapter discusses the **blue**future directions of the threat modelling technique for GDPR-compliance.

# Chapter 9

# Conclusion and Future Work

## 9.1 Conclusion

The main purpose of this thesis is to develop a threat modelling technique for GDPR compliance based on logical reasoning capable of modelling non-compliance threats in real-world scenarios. The thesis details the current literature in the field of existing threat modelling techniques and highlights their limitations. For example, data privacy modelling techniques like LINDDUN have been implemented in the design phase of systems [31, 32], the suitability and effectiveness in determining and mitigating threats of non-compliance with complex data protection laws, including the GDPR [197], is often found to be insufficient [181]. Modelling non-compliance threats of GDPR is important because in the modern digital landscape, personal data undergoes frequent processing and sharing through diverse applications, spanning sectors like finance, automotive, and healthcare services. The widespread adoption of these data-intensive applications exposes our information to elevated risks of privacy breaches and data theft.

To address this challenge, this thesis presents a novel threat modelling technique for GDPR non-compliance leveraging logical reasoning. This solution is an expert system based on DeLP converted from Rule-based language that consists of a knowledge base designated specifically for GDPR compliance with an inference engine reasoning over the knowledge base. DeLP's ability to model real-world knowledge makes it ideal for scenarios where the knowledge base is incomplete, information can be conflicting or subject to change [136, 139]. These unique advantages enable DeLP to effectively model the complexity of real-world reasoning such as handling the varied exceptions and conditions found in the GDPR [198, 11] productively.

The thesis chapters elaborate on how the reasoner infers the threats over the DeLP knowledge base and resolves the problem of "UNDECIDED" queries when conflict or missing information occurs during the defeasible reasoning process. The developed modelling program is distinguished by its capacity to manage contradictions by incorporating explicit prioritisation among conflicting rules in the knowledge base. This allows for a more sophisticated analysis of a modelled system to mitigate non-compliance threats, enabling it to comply with the intricate GDPR requirements. Furthermore, to improve the reasoning abilities and resilience of the proposed technique, we also resolve the scenarios of incomplete knowledge, allowing users (i.e., system modellers) to add relevant information suggested by our modelling program to successfully determine the compliance outputs.

The work investigates how the DeLP-based modelling technique outperforms previous threat modelling techniques and introduces the novel concepts of handling the "UNDECIDED" query results. It opens the door for future advances in modelling data protection regulations. It demonstrates the experiments and evaluates the performance of our proposed modelling tool with consistent results.

To summarize, this thesis introduces a novel threat modeling technique specifically for GDPR non-compliance, using DeLP to address the complexities of real-world scenarios where information may be incomplete or conflicting. Unlike traditional models such as LINDDUN, which are limited in addressing dynamic regulatory requirements, this approach enables nuanced compliance analysis by resolving “UNDECIDED” outcomes through prioritized rule conflicts and user-input mechanisms when additional data is required. The developed system includes a GDPR-specific knowledge base and an inference engine, which together support robust reasoning for key GDPR principles, including consent, accountability, and rectification. Additionally, it integrates threat modeling standards like STRIDE and LINDDUN to enhance both security and privacy. Experimental results demonstrate that the model is effective, scalable, and consistent in handling diverse compliance queries across different knowledge base complexities, making it a valuable advancement in data protection modeling.

## 9.2 Limitation

Herein, there are some limitations of logical reasoning-based modelling tool, such as those listed in the following section.

1. **Binary nature of query results:** One significant limitation is the binary nature of the answers. The system’s ability to provide only three distinct responses might not capture the full complexity of certain real-world scenarios. In cases where the compliance determination requires nuanced reasoning, the system’s binary responses may not fully address the intricacies of the situation.

2. **Explicit priorities in the knowledge base:** Another limitation is that users cannot directly set priorities while building the knowledge base because of the system’s inability to parse priorities within the DeLP knowledge base. The prioritisation process can be enhanced by allowing users to assign priorities directly within the knowledge base. This approach eliminates the need for users to specify priorities each time a conflict arises during reasoning. This approach could greatly improve the system’s re-usability, making it

more robust and adaptable. Since predicting "UNDECIDED" queries is challenging, the main goal here is to simplify the process of storing priorities and minimize the need for user input.

3. **"UNDECIDED" result due to human error:** When users are asked to provide missing facts from an identified list to support a query, they might still fail to include all relevant facts, leading to an "UNDECIDED" result. However, this issue often results from human error rather than a flaw in the system's implementation. This highlights the challenge of ensuring that users fully understand and respond to the system's requirements for complete and accurate data input.

4. **Exponential rise in complexity:** The exponential increase in complexity from rules with numerous conditions can significantly impact the practicality of applying defeasible logic to legal reasoning. This challenge primarily arises due to the implementation approach: if the knowledge base is constructed with high horizontal complexity (i.e., high number of conditions) of any relevant rule, it amplifies computational demands. Simplifying rules and keeping horizontal complexity to a minimum can mitigate this issue, making the application of defeasible logic more feasible for legal reasoning tasks.

## 9.3 Future Work

Future research efforts will focus on the many aspects of future threat modelling techniques for GDPR-compliance, such as the following.

1. **Advancing Legal Rule Modeling with DeLP:** This study establishes a foundational framework for applying DeLP (Defeasible Logic Programming) in legal rule modeling, demonstrating its potential in complex legal decision-making. To address the limitations noted, particularly the challenge of adapting to ongoing GDPR updates, future work could explore the integration of machine learning to enable dynamic rule updates and enhanced conflict resolution, ensuring that the system can adapt to new regulations as they emerge. Testing these techniques in real-world compliance monitoring across various industries could also expand DeLP's applicability for automated legal reasoning in regulatory processes. Integrating natural language processing tools to automate the interpretation of regulatory changes could further support dynamic updates [199, 200]. These enhancements would improve DeLP's responsiveness to evolving compliance requirements, making it a robust solution for regulatory environments.

2. **Enhancing the performance of the developed modelling tool in highly complex environments:** To enhance the performance of the developed modeling tool in highly complex environments, future work should focus on developing more robust algorithms and

employing advanced optimization techniques to efficiently handle and reason with large-scale rule sets [195]. This approach is essential for domains requiring management of extensive, intricate rule systems, such as financial regulations, large-scale enterprise policies, and complex legal compliance environments [201]. Specific strategies could include leveraging parallel processing to reduce computational time and exploring heuristic optimization methods to improve reasoning efficiency [202]. Additionally, integrating machine learning and artificial intelligence could automate parts of the decision-making process, enabling real-time analytics and insights, even under high complexity. Applying reinforcement learning could further refine decision pathways, improving accuracy and adaptability in dynamic, rule-intensive contexts.

3. **Exploring the Integration of DeLP with Other Logical Frameworks for Enhanced Decision-Making:** Future research could focus on enriching DeLP by integrating it with various other logical frameworks that offer unique benefits for specific contexts. This includes exploring the synergies between DeLP and deontic logic, which is pivotal for normative reasoning in legal contexts [203]. Additionally, the integration with argumentation frameworks [204] [205], could enhance the robustness of decision-making processes by structuring interactive reasoning. Another promising area is the incorporation of negation-as-failure techniques, which offer a powerful tool for handling incomplete information as detailed [206]. By merging these methodologies with DeLP, future work could yield more adaptable and effective tools for tailored application scenarios, significantly advancing the capacity of systems to handle the complexity and specificity of regulatory compliance and other domain-specific reasoning tasks [207].

# Bibliography

[1] Michael I Jordan and Tom M Mitchell. “Machine learning: Trends, perspectives, and prospects”. In: *Science* 349.6245 (2015), pp. 255–260.

[2] Nguyen Binh Truong et al. “Gdpr-compliant personal data management: A blockchain-based solution”. In: *IEEE Transactions on Information Forensics and Security* 15 (2019), pp. 1746–1761.

[3] Finale Doshi-Velez and Been Kim. “Towards a rigorous science of interpretable machine learning”. In: *arXiv preprint arXiv:1702.08608* (2017).

[4] Mike Ananny and Kate Crawford. “Seeing without knowing: Limitations of the transparency ideal and its application to algorithmic accountability”. In: *new media & society* 20.3 (2018), pp. 973–989.

[5] Frederik Harder, Matthias Bauer, and Mijung Park. “Interpretable and Differentially Private Predictions.” In: *AAAI*. 2020, pp. 4083–4090.

[6] Amina Adadi and Mohammed Berrada. “Peeking inside the black-box: a survey on explainable artificial intelligence (XAI)”. In: *IEEE access* 6 (2018), pp. 52138–52160.

[7] Dimitri Van Landuyt and Wouter Joosen. “A descriptive study of assumptions made in LINDDUN privacy threat elicitation”. In: *Proceedings of the 35th Annual ACM Symposium on Applied Computing*. 2020, pp. 1280–1287.

[8] Peter Torr. “Demystifying the threat modeling process”. In: *IEEE Security & Privacy* 3.5 (2005), pp. 66–70.

[9] Howard Michael and Lipner Steve. *The Security Development Lifecycle: SDL: A Process for Developing Demonstrably More Secure Software*. 2006.

[10] Samuel Greengard. “Weighing the impact of GDPR”. In: *Communications of the ACM* 61.11 (2018), pp. 16–18.

[11] Chris Jay Hoofnagle, Bart Van Der Sloot, and Frederik Zuiderveen Borgesius. “The European Union general data protection regulation: what it is and what it means”. In: *Information & Communications Technology Law* 28.1 (2019), pp. 65–98.

[12] Orlando Amaral et al. “Nlp-based automated compliance checking of data processing agreements against gdpr”. In: *IEEE Transactions on Software Engineering* (2023).

[13] J Fruhlinger. "Threat modeling explained: A process for anticipating cyber attacks". In: *Tersedia pada: https://www.csoonline.com/article/3537370/threat-modeling-explained-a-process-for-anticipating-cyber-attacks.html* (2020).

[14] Wenjun Xiong and Robert Lagerström. "Threat modeling–A systematic literature review". In: *Computers & security* 84 (2019), pp. 53–69.

[15] Gerald Kotonya and Ian Sommerville. *Requirements engineering: processes and techniques*. John Wiley & Sons, Inc., 1998.

[16] Julia H Allen et al. *Software security engineering*. Pearson India, 2008.

[17] Hideaki Takeda, Paul Veerkamp, and Hiroyuki Yoshikawa. "Modeling design process". In: *AI magazine* 11.4 (1990), pp. 37–37.

[18] Amy Poh Ai Ling and Mukaidono Masao. "Selection of model in developing information security criteria on smart grid security system". In: *2011 IEEE Ninth International Symposium on Parallel and Distributed Processing with Applications Workshops*. IEEE. 2011, pp. 91–98.

[19] Umar Zakir Abdul Hamid et al. "A review on threat assessment, path planning and path tracking strategies for collision avoidance systems of autonomous vehicles". In: *International Journal of Vehicle Autonomous Systems* 14.2 (2018), pp. 134–169.

[20] Adam Shostack. *Threat modeling: Designing for security*. John Wiley & Sons, 2014.

[21] Vineet Saini, Qiang Duan, and Vamsi Paruchuri. "Threat modeling using attack trees". In: *Journal of Computing Sciences in Colleges* 23.4 (2008), pp. 124–131.

[22] Tony UcedaVelez. "Real world threat modeling using the pasta methodology. Technical report". In: *OWASP App Sec EU* 1.0 (2012). URL: `https://www.owasp.org/images/a/aa/AppSecEU2012_PASTA.pdf`.

[23] Riccardo Scandariato, Kim Wuyts, and Wouter Joosen. "A descriptive study of Microsoft's threat modeling technique". In: *Requirements Engineering* 20.2 (2015), pp. 163–180.

[24] Rafiullah Khan et al. "STRIDE-based threat modeling for cyber-physical systems". In: *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*. IEEE. 2017, pp. 1–6.

[25] Sergej Japs and Harald Anacker. "Resolution of safety relevant security threats in the system architecture design phase on the example of automotive industry". In: *Proceedings of the design society* 1 (2021), pp. 2561–2570.

[26] Syed Ghazanfar Abbas et al. "Identifying and mitigating phishing attack threats in IoT use cases using a threat modelling approach". In: *Sensors* 21.14 (2021), p. 4816.

[27] J Howell and B Kess. “Baldwin”. In: *Microsoft Threat Modeling Tool. Available online: https://docs. microsoft. com/en-us/azure/security/develop/threat-modeling-tool (accessed on 12 June 2021)* ().

[28] Simon Parkinson et al. “Cyber threats facing autonomous and connected vehicles: Future challenges”. In: *IEEE transactions on intelligent transportation systems* 18.11 (2017), pp. 2898–2915.

[29] Andreas Jacobsson, Martin Boldt, and Bengt Carlsson. “A risk analysis of a smart home automation system”. In: *Future Generation Computer Systems* 56 (2016), pp. 719–733.

[30] Dimitri Van Landuyt and Wouter Joosen. “A descriptive study of assumptions in STRIDE security threat modeling”. In: *Software and Systems Modeling* (2021), pp. 1–18.

[31] Kim Wuyts, Laurens Sion, and Wouter Joosen. “Linddun go: A lightweight approach to privacy threat modeling”. In: *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE. 2020, pp. 302–309.

[32] Kim Wuyts and Wouter Joosen. “LINDDUN privacy threat modeling: a tutorial”. In: *CW Reports* CW685 (July 1, 2015). URL: `https://lirias.kuleuven.be/retrieve/331950`.

[33] Dimitri Van Landuyt and Wouter Joosen. “A descriptive study of assumptions made in LINDDUN privacy threat elicitation”. In: *Proceedings of the 35th Annual ACM Symposium on Applied Computing*. 2020, pp. 1280–1287.

[34] Kim Wuyts, Riccardo Scandariato, and Wouter Joosen. “Empirical evaluation of a privacy-focused threat modeling methodology”. In: *Journal of Systems and Software* 96 (2014), pp. 122–138.

[35] Nancy Mead, Eric Hough, and Ted Stehney II. “Security Quality Requirements Engineering Technical Report”. In: *Tech. Rep. CMU/SEI-2005-TR-009* (2005).

[36] F Shull. “Evaluation of threat modeling methodologies”. In: *Software Engineering Institute, Carne-gie Mellon University* (2016).

[37] Nataliya Shevchenko, Brent R Frye, and Carol Woody. *Threat modeling for cyber-physical system-of-systems: Methods evaluation*. Tech. rep. AD1084209. Carnegie Mellon University Software Engineering Institute Pittsburgh United . . ., Sept. 1, 2018. URL: `https://apps.dtic.mil/sti/pdfs/AD1084209.pdf`.

[38] Tony UcedaVelez and Marco M Morana. *Risk Centric Threat Modeling: process for attack simulation and threat analysis*. John Wiley & Sons, 2015.

[39] Jane Cleland-Huang. “How well do you know your personae non gratae?” In: *IEEE software* 31.4 (2014), pp. 28–31.

[40] Nancy R Mead et al. "A hybrid threat modeling method". In: *Carnegie MellonUniversity-Software Engineering Institute-Technical Report-CMU/SEI-2018-TN-002* (2018).

[41] Nancy R Mead and Ted Stehney. "Security quality requirements engineering (SQUARE) methodology". In: *ACM SIGSOFT Software Engineering Notes* 30.4 (2005), pp. 1–7.

[42] Peter Mell, Karen Scarfone, and Sasha Romanosky. "Common vulnerability scoring system". In: *IEEE Security & Privacy* 4.6 (2006), pp. 85–89.

[43] Peter Mell, Karen Scarfone, Sasha Romanosky, et al. "A complete guide to the common vulnerability scoring system version 2.0". In: *Published by FIRST-forum of incident response and security teams*. Vol. 1. 2007, p. 23.

[44] Laurens Sion et al. "Solution-aware data flow diagrams for security threat modeling". In: *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*. 2018, pp. 1425–1432.

[45] Bruce Schneier. "Attack trees". In: *Dr. Dobb's journal* 24.12 (1999), pp. 21–29.

[46] Jesus Luna, Neeraj Suri, and Ioannis Krontiris. "Privacy-by-design based on quantitative threat modeling". In: *2012 7th International Conference on Risks and Security of Internet and Systems (CRiSIS)*. IEEE. 2012, pp. 1–8.

[47] Sri Murugarasan Muthukrishnan and Sellapan Palaniappan. "Security metrics maturity model for operational security". In: *2016 IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE)*. IEEE. 2016, pp. 101–106.

[48] Paul Saitta, Brenda Larcom, and Michael Eddington. "Trike v. 1 methodology document [draft]". In: *URL: http://dymaxion. org/trike/Trike v1 Methodology Documentdraft. pdf* (2005).

[49] Ashley Aitken. "Dual Application Model for Agile Software Engineering". In: *2014 47th Hawaii International Conference on System Sciences*. IEEE. 2014, pp. 4789–4798.

[50] Nancy R Mead et al. "A hybrid threat modeling method". In: *Carnegie MellonUniversity-Software Engineering Institute-Technical Report-CMU/SEI-2018-TN-002* (2018).

[51] Shiho Kim and Rakesh Shrestha. "Security and Privacy in Intelligent Autonomous Vehicles". In: *Automotive Cyber Security*. Springer, 2020, pp. 35–66.

[52] Christopher Alberts et al. *Introduction to the OCTAVE Approach*. Tech. rep. ADA634134. Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst, Aug. 1, 2003. URL: `https://apps.dtic.mil/sti/pdfs/ADA634134.pdf`.

[53] Christopher J Alberts and Audrey J Dorofee. *Managing information security risks: the OCTAVE approach*. Addison-Wesley Professional, 2003.

[54] Nataliya Shevchenko et al. *Threat modeling: a summary of available methods*. Tech. rep. AD1084024. Carnegie Mellon University Software Engineering Institute Pittsburgh United . . ., July 1, 2018. URL: `https://apps.dtic.mil/sti/pdfs/AD1084024.pdf`.

[55] S. Sharma. *Data Privacy and GDPR Handbook*. Wiley, 2019. ISBN: 9781119594253. URL: `https://books.google.co.uk/books?id=4XnADwAAQBAJ`.

[56] Monika Budrytė. "General data protection regulation (GDPR) in European Union: from proposal to implementation". B.S. thesis. 2021.

[57] David Wright and Charles Raab. "Privacy principles, risks and harms". In: *International Review of Law, Computers & Technology* 28.3 (2014), pp. 277–298.

[58] He Li, Lu Yu, and Wu He. *The impact of GDPR on global technology development*. 2019.

[59] Nguyen Binh Truong et al. "Gdpr-compliant personal data management: A blockchain-based solution". In: *IEEE Transactions on Information Forensics and Security* 15 (2019), pp. 1746–1761.

[60] Clément Labadie and Christine Legner. "Building data management capabilities to address data protection regulations: Learnings from EU-GDPR". In: *Journal of Information Technology* 38.1 (2023), pp. 16–44.

[61] Paul Voigt and Axel Von dem Bussche. "The eu general data protection regulation (gdpr)". In: *A Practical Guide, 1st Ed., Cham: Springer International Publishing* 10.3152676 (2017), pp. 10–5555.

[62] Gauthier Chassang. "The impact of the EU general data protection regulation on scientific research". In: *ecancermedicalscience* 11 (2017).

[63] Christian Peukert et al. "Regulatory Spillovers and Data Governance: Evidence from the GDPR". In: *Marketing Science* (2022).

[64] W Gregory Voss. "European union data privacy law reform: General data protection regulation, privacy shield, and the right to delisting". In: *The Business Lawyer* 72.1 (2016), pp. 221–234.

[65] Nicholas F Palmieri III. "Data protection in an increasingly globalized world". In: *Ind. LJ* 94 (2019), p. 297.

[66] Damian A Tamburri. "Design principles for the General Data Protection Regulation (GDPR): A formal concept analysis and its evaluation". In: *Information Systems* 91 (2020), p. 101469.

[67] Rachel L Finn and David Wright. “Unmanned aircraft systems: Surveillance, ethics and privacy in civil applications”. In: *Computer Law & Security Review* 28.2 (2012), pp. 184–194.

[68] Maryam Davari and Elisa Bertino. “Access control model extensions to support data privacy protection based on GDPR”. In: *2019 IEEE International Conference on Big Data (Big Data)*. IEEE. 2019, pp. 4017–4024.

[69] Upul Jayasinghe, Gyu Myoung Lee, and Aine MacDermott. “Trust-based data controller for personal information management”. In: *2018 International Conference on Innovations in Information Technology (IIT)*. IEEE. 2018, pp. 123–128.

[70] Mike Hintze. “Data controllers, data processors, and the growing use of connected products in the enterprise: Managing risks, understanding benefits, and complying with the GDPR”. In: *Journal of Internet Law (Wolters Kluwer), August* (2018).

[71] PTJ Wolters. “The control by and rights of the data subject under the GDPR”. In: (2018).

[72] Aloni Cohen and Kobbi Nissim. “Towards formalizing the GDPR’s notion of singling out”. In: *Proceedings of the National Academy of Sciences* 117.15 (2020), pp. 8344–8352.

[73] Cynthia Dwork. “Differential privacy: A survey of results”. In: *International conference on theory and applications of models of computation*. Springer. 2008, pp. 1–19.

[74] Latanya Sweeney. “k-anonymity: A model for protecting privacy”. In: *International journal of uncertainty, fuzziness and knowledge-based systems* 10.05 (2002), pp. 557–570.

[75] Cristòfol Daudén-Esmel, Jordi Castellà-Roca, and Alexandre Viejo. “Blockchain-based access control system for efficient and GDPR-compliant personal data management”. In: *Computer Communications* 214 (2024), pp. 67–87.

[76] Maria Rigaki and Sebastian Garcia. “A survey of privacy attacks in machine learning”. In: *arXiv preprint arXiv:2007.07646* (2020).

[77] Evelien Renate Brouwer. “Legality and data protection law: The forgotten purpose of purpose limitation”. In: (2011).

[78] John Krumm. “Inference attacks on location tracks”. In: *International Conference on Pervasive Computing*. Springer. 2007, pp. 127–143.

[79] Reza Shokri et al. “Membership inference attacks against machine learning models”. In: *2017 IEEE symposium on security and privacy (SP)*. IEEE. 2017, pp. 3–18.

[80] Viktoria HSE Robertson. “Excessive data collection: privacy considerations and abuse of dominance in the era of big data”. In: *Common Market Law Review* 57.1 (2020).

[81] Marius Wernke et al. "A classification of location privacy attacks and approaches". In: *Personal and ubiquitous computing* 18.1 (2014), pp. 163–175.

[82] Elisabetta Biasin. "Why accuracy needs further exploration in data protection". In: *Proceedings of the 1st International Conference on AI for People: Towards Sustainable AI*. EAI. 2021, pp. 1–7.

[83] Andraž Krašovec, Gianmarco Baldini, and Veljko Pejović. "Opposing Data Exploitation: Behaviour Biometrics for Privacy-Preserving Authentication in IoT Environments". In: *The 16th International Conference on Availability, Reliability and Security*. 2021, pp. 1–7.

[84] Csilla Farkas. "Big Data Analytics: Privacy Protection using Semantic Web Technologies". In: *NSF Workshop on Big Data Security and Privacy*. 2014. URL: `https://csi.utdallas.edu/events/NSF/papers/paper06.pdf`.

[85] Jawwad A Shamsi and Muhammad Ali Khojaye. "Understanding privacy violations in big data systems". In: *It Professional* 20.3 (2018), pp. 73–81.

[86] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. "t-closeness: Privacy beyond k-anonymity and l-diversity". In: *2007 IEEE 23rd international conference on data engineering*. IEEE. 2007, pp. 106–115.

[87] Jordi Soria-Comas and Josep Domingo-Ferrert. "Differential privacy via t-closeness in data publishing". In: *2013 Eleventh Annual Conference on Privacy, Security and Trust*. IEEE. 2013, pp. 27–35.

[88] M Zekeriya Gunduz and Resul Das. "Analysis of cyber-attacks on smart grid applications". In: *2018 International Conference on Artificial Intelligence and Data Processing (IDAP)*. IEEE. 2018, pp. 1–5.

[89] Shams Tabrez Siddiqui et al. "Security threats, attacks, and possible countermeasures in internet of things". In: *Advances in data and information sciences*. Springer, 2020, pp. 35–46.

[90] Inayat Ali, Sonia Sabir, and Zahid Ullah. "Internet of things security, device authentication and access control: a review". In: *arXiv preprint arXiv:1901.07309* (2019).

[91] Mohamed Sharaf. "Non-repudiation and privacy-preserving sharing of electronic health records". In: *Cogent Engineering* 9.1 (2022), p. 2034374.

[92] Daniel Omeiza et al. "Explanations in autonomous driving: A survey". In: *IEEE Transactions on Intelligent Transportation Systems* (2021).

[93] David S Hall. *High definition lidar system*. US Patent 7,969,558. June 2011.

[94] David K Barton. *Radar system analysis and modeling*. Artech House, 2004.

[95] Taehyun Ha et al. “Effects of explanation types and perceived risk on trust in autonomous vehicles”. In: *Transportation research part F: traffic psychology and behaviour* 73 (2020), pp. 271–280.

[96] Fabio Cuzzolin et al. “Knowing me, knowing you: theory of mind in AI”. In: *Psychological medicine* 50.7 (2020), pp. 1057–1061.

[97] M McFarland. “Who’s responsible when an autonomous car crashes?” In: *Accessed: Jul* 24 (2016), p. 2020.

[98] Ekim Yurtsever et al. “A survey of autonomous driving: Common practices and emerging technologies”. In: *IEEE access* 8 (2020), pp. 58443–58469.

[99] Jeamin Koo et al. “Why did my car just do that? Explaining semi-autonomous driving actions to improve driver understanding, trust, and performance”. In: *International Journal on Interactive Design and Manufacturing (IJIDeM)* 9.4 (2015), pp. 269–275.

[100] Rasheed Hussain and Sherali Zeadally. “Autonomous cars: Research results, issues, and future challenges”. In: *IEEE Communications Surveys & Tutorials* 21.2 (2018), pp. 1275–1313.

[101] Swaroop Darbha, Shyamprasad Konduri, and Prabhakar R Pagilla. “Benefits of V2V communication for autonomous and connected vehicles”. In: *IEEE Transactions on Intelligent Transportation Systems* 20.5 (2018), pp. 1954–1963.

[102] John Harding et al. *Vehicle-to-vehicle communications: readiness of V2V technology for application.* Tech. rep. DOT HS 812 014. United States. National Highway Traffic Safety Administration, Aug. 1, 2014. URL: `https://rosap.ntl.bts.gov/view/dot/27999`.

[103] Kakan Chandra Dey et al. “Vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication in a heterogeneous wireless network–Performance evaluation”. In: *Transportation Research Part C: Emerging Technologies* 68 (2016), pp. 168–184.

[104] Hesham Rakha and Raj Kishore Kamalanathsharma. “Eco-driving at signalized intersections using V2I communication”. In: *2011 14th international IEEE conference on intelligent transportation systems (ITSC)*. IEEE. 2011, pp. 341–346.

[105] Jonathan Petit and Steven E Shladover. “Potential cyberattacks on automated vehicles”. In: *IEEE Transactions on Intelligent transportation systems* 16.2 (2014), pp. 546–556.

[106] Taxonomy SAE. “Definitions for terms related to driving automation systems for on-road motor vehicles”. In: *SAE Standard J* 3016 (2016), p. 2016.

[107] Karl Koscher et al. “Experimental security analysis of a modern automobile”. In: *2010 IEEE symposium on security and privacy*. IEEE. 2010, pp. 447–462.

[108] Jiajia Liu et al. “In-vehicle network attacks and countermeasures: Challenges and future directions”. In: *IEEE Network* 31.5 (Jan. 2017), pp. 50–58.

[109] Hyunjae Kang et al. “Car Hacking and Defense Competition on In-Vehicle Network”. In: *Workshop on Automotive and Autonomous Vehicle Security (AutoSec)*. Vol. 2021. 2021, p. 25.

[110] Bryson R Payne. “Car Hacking: Accessing and Exploiting the CAN Bus Protocol”. In: *Journal of Cybersecurity Education, Research and Practice* 2019.1 (2019), p. 5.

[111] Sohan Gyawali et al. “Challenges and solutions for cellular based V2X communications”. In: *IEEE Communications Surveys & Tutorials* 23.1 (2020), pp. 222–255.

[112] Kyounggon Kim et al. “Cybersecurity for autonomous vehicles: Review of attacks and defense”. In: *Computers & Security* 103 (2021), p. 102150.

[113] Vrizlynn LL Thing and Jiaxi Wu. “Autonomous vehicle security: A taxonomy of attacks and defences”. In: *2016 ieee international conference on internet of things (ithings) and ieee green computing and communications (greencom) and ieee cyber, physical and social computing (cpscom) and ieee smart data (smartdata)*. IEEE. 2016, pp. 164–170.

[114] Simona Gifei and Alexandru Salceanu. “Integrated Management System for quality, safety and security in developing autonomous vehicles”. In: *2017 10th International Symposium on Advanced Topics in Electrical Engineering (ATEE)*. IEEE. 2017, pp. 673–676.

[115] Samuel Woo, Hyo Jin Jo, and Dong Hoon Lee. “A practical wireless attack on the connected car and security protocol for in-vehicle CAN”. In: *IEEE Transactions on intelligent transportation systems* 16.2 (2014), pp. 993–1006.

[116] Shaoshan Liu et al. “A unified cloud platform for autonomous driving”. In: *Computer* 50.12 (2017), pp. 42–49.

[117] Sadeq Almeaibed et al. “Digital twin analysis to promote safety and security in autonomous vehicles”. In: *IEEE Communications Standards Magazine* 5.1 (2021), pp. 40–46.

[118] Chen Yan, Wenyuan Xu, and Jianhao Liu. “Can you trust autonomous vehicles: Contactless attacks against sensors of self-driving vehicle”. In: *Def Con* 24.8 (2016), p. 109.

[119] Shubha R Shetty and DH Manjaiah. “A Comprehensive Study of Security Attack on VANET”. In: *Data Management, Analytics and Innovation*. Springer, 2022, pp. 407–428.

[120] Mujahid Muhammad and Ghazanfar Ali Safdar. “Survey on existing authentication issues for cellular-assisted V2X communication”. In: *Vehicular Communications* 12 (2018), pp. 50–65.

[121] Bharat K Bhargava et al. “A Systematic Approach for Attack Analysis and Mitigation in V2V Networks.” In: *J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl.* 7.1 (2016), pp. 79–96.

[122] Rushit Dave, ES Boone, and Kaushik Roy. “Efficient Data Privacy and Security in Autonomous Cars”. In: *Journal of Computer Sciences and Applications* 7.1 (2019), pp. 31–36.

[123] Cara Bloom et al. “Self-driving cars and data collection: Privacy perceptions of networked autonomous vehicles”. In: *Thirteenth Symposium on Usable Privacy and Security (SOUPS 2017)*. 2017, pp. 357–375.

[124] Axel Ekdahl and Lídia Nyman. “A Methodology to Validate Compliance to the GDPR”. In: (Nov. 19, 2019). URL: `http://hdl.handle.net/2077/62557`.

[125] Federico Costantini et al. “Autonomous vehicles in a GDPR era: An international comparison”. In: *Advances in Transport Policy and Planning*. Vol. 5. Elsevier, 2020, pp. 191–213.

[126] Auxane Boch, Ellen Hohma, and Rainer Trauth. “Towards an Accountability Framework for AI: Ethical and Legal Considerations”. In: (2022). URL: `https://ieai.sot.tum.de/wp-content/uploads/2022/03/ResearchBrief_March_Boch_Hohma_Trauth_FINAL_V2.pdf`.

[127] María Cristina Gaeta. “Data protection and self-driving cars: The consent to the processing of personal data in compliance with GDPR”. In: *Communications Law* 24.1 (2019), pp. 15–23.

[128] John R Anderson and Christian Lebiere. “Knowledge representation”. In: *The atomic components of thought*. Psychology Press, 2014, pp. 19–55.

[129] Ronald Brachman and Hector Levesque. *Knowledge representation and reasoning*. Morgan Kaufmann, 2004.

[130] Agnieszka Nowak-Brzezińska and Alicja Wakulicz-Deja. “Exploration of rule-based knowledge bases: A knowledge engineer’s support”. In: *Information Sciences* 485 (2019), pp. 301–318.

[131] Khalil AbuDahab, Dong-ling Xu, and Yu-wang Chen. “A new belief rule base knowledge representation scheme and inference methodology using the evidential reasoning rule for evidence combination”. In: *Expert Systems with Applications* 51 (2016), pp. 218–230.

[132] Czesław Horyń and Agnieszka Nowak Brzezińska. “Detecting outliers in rule-based knowledge bases using Self-Organizing Map and Local Outlier Factor algorithms”. In: *Procedia Computer Science* 225 (2023), pp. 2116–2125.

[133] Agnieszka Nowak-Brzezińska and Alicja Wakulicz-Deja. “Exploration of rule-based knowledge bases: A knowledge engineer’s support”. In: *Information Sciences* 485 (2019), pp. 301–318. ISSN: 0020-0255. DOI: `https://doi.org/10.1016/j.ins.2019.02.019`. URL: `https://www.sciencedirect.com/science/article/pii/S002002551930129X`.

[134] Donald Nute. “Defeasible logic”. In: *International Conference on Applications of Prolog*. Springer. 2001, pp. 151–169.

[135] Alejandro J García and Guillermo R Simari. “Defeasible logic programming: An argumentative approach”. In: *Theory and practice of logic programming* 4.1-2 (2004), pp. 95–138.

[136] Jaap Hage. “Law and defeasibility”. In: *Studies in legal logic* (2005), pp. 7–32.

[137] Rachel Rudinger et al. “Thinking like a skeptic: Defeasible inference in natural language”. In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. 2020, pp. 4661–4675.

[138] Aman Madaan et al. “Think about it! improving defeasible reasoning by first modeling the question scenario”. In: *arXiv preprint arXiv:2110.12349* (2021).

[139] Guido Governatori et al. “Computing strong and weak permissions in defeasible logic”. In: *Journal of Philosophical Logic* 42 (2013), pp. 799–829.

[140] Xianchang Wang, Jia-Huai You, and Li Yan Yuan. “Logic programming without default negation revisited”. In: *1997 IEEE International Conference on Intelligent Processing Systems (Cat. No. 97TH8335)*. Vol. 2. IEEE. 1997, pp. 1169–1173.

[141] David Poole. “A logical framework for default reasoning”. In: *Artificial intelligence* 36.1 (1988), pp. 27–47.

[142] John L Pollock. “Defeasible reasoning”. In: *Cognitive science* 11.4 (1987), pp. 481–518.

[143] Raymond Reiter. “Nonmonotonic reasoning”. In: *Exploring artificial intelligence*. Elsevier, 1988, pp. 439–481.

[144] Emily Allaway et al. “Penguins Don’t Fly: Reasoning about Generics through Instantiations and Exceptions”. In: *arXiv preprint arXiv:2205.11658* (2022).

[145] Sumithra Bhakthavatsalam, Chloe Anastasiades, and Peter Clark. “Genericskb: A knowledge base of generic statements”. In: *arXiv preprint arXiv:2005.00660* (2020).

[146] Paul N Otto and Annie I Antón. “Addressing legal requirements in requirements engineering”. In: *15th IEEE international requirements engineering conference (RE 2007)*. IEEE. 2007, pp. 5–14.

[147] Benjamin Johnston and Guido Governatori. “Induction of defeasible logic theories in the legal domain”. In: *Proceedings of the 9th international conference on Artificial intelligence and law*. 2003, pp. 204–213.

[148] Grigoris Antoniou, David Billington, and Michael J Maher. “On the analysis of regulations using defeasible rules”. In: *Proceedings of the 32nd Annual Hawaii International Conference on Systems Sciences. 1999. HICSS-32. Abstracts and CD-ROM of Full Papers*. IEEE. 1999, 7–pp.

[149] Mehran Kazemi et al. “Boardgameqa: A dataset for natural language reasoning with contradictory information”. In: *Advances in Neural Information Processing Systems* 36 (2024).

[150] Xin Sun and Livio Robaldo. “On the complexity of input/output logic”. In: *Journal of Applied Logic* 25 (2017), pp. 69–88.

[151] Antonino Rotolo, Guido Governatori, and Giovanni Sartor. “Deontic defeasible reasoning in legal interpretation: two options for modelling interpretive arguments”. In: *Proceedings of the 15th international conference on artificial intelligence and law*. 2015, pp. 99–108.

[152] Livio Robaldo et al. “Formalizing GDPR provisions in reified I/O logic: the DAPRECO knowledge base”. In: *Journal of Logic, Language and Information* 29.4 (2020), pp. 401–449.

[153] Henry Prakken. *Logical tools for modelling legal argument: a study of defeasible reasoning in law*. Vol. 32. Springer Science & Business Media, 2013.

[154] Guido Governatori and Michael J Maher. “Annotated defeasible logic”. In: *Theory and Practice of Logic Programming* 17.5-6 (2017), pp. 819–836.

[155] Grigoris Antoniou et al. “On the modeling and analysis of regulations”. In: (1999).

[156] Mohammad Badiul Islam and Guido Governatori. “RuleRS: a rule-based architecture for decision support systems”. In: *Artificial Intelligence and Law* 26.4 (2018), pp. 315–344.

[157] Guido Governatori and Duy Hoang Pham. “Dr-contract: An architecture for e-contracts in defeasible logic”. In: *International Journal of Business Process Integration and Management* 4.3 (2009), pp. 187–199.

[158] Thomas Skylogiannis et al. “DR-NEGOTIATE–A system for automated agent negotiation with defeasible logic-based strategies”. In: *Data & Knowledge Engineering* 63.2 (2007), pp. 362–380.

[159] Guido Governatori, Zoran Milosevic, and Shazia Sadiq. “Compliance checking between business processes and business contracts”. In: *2006 10th IEEE International Enterprise Distributed Object Computing Conference (EDOC’06)*. IEEE. 2006, pp. 221–232.

[160] Marco Billi et al. “Argumentation and defeasible reasoning in the law”. In: *J* 4.4 (2021), pp. 897–914.

[161] Luciano Caroprese, Eugenio Vocaturo, and Ester Zumpano. “Argumentation approaches for explanaible ai in medical informatics”. In: *Intelligent Systems with Applications* 16 (2022), p. 200109.

[162] Michael J Maher et al. “Efficient defeasible reasoning systems”. In: *International Journal on Artificial Intelligence Tools* 10.04 (2001), pp. 483–501.

[163] Harshvardhan Jitendra Pandit, Declan O’Sullivan, and Dave Lewis. “Towards Knowledge-Based Systems for GDPR Compliance.” In: *CKGSemStats@ ISWC*. 2018.

[164] Mike Surridge et al. “Modelling compliance threats and security analysis of cross border health data exchange”. In: *New Trends in Model and Data Engineering: MEDI 2019 International Workshops, DETECT, DSSGA, TRIDENT, Toulouse, France, October 28–31, 2019, Proceedings 9*. Springer. 2019, pp. 180–189.

[165] Marco Robol, Mattia Salnitri, and Paolo Giorgini. “Toward GDPR-compliant socio-technical systems: modeling language and reasoning framework”. In: *The Practice of Enterprise Modeling: 10th IFIP WG 8.1. Working Conference, PoEM 2017, Leuven, Belgium, November 22-24, 2017, Proceedings 10*. Springer. 2017, pp. 236–250.

[166] Harshvardhan J Pandit, Declan O’Sullivan, and Dave Lewis. “Queryable provenance metadata for GDPR compliance”. In: *Procedia Computer Science* 137 (2018), pp. 262–268.

[167] Harshvardhan J Pandit et al. “GDPRtEXT-GDPR as a linked data resource”. In: *European Semantic Web Conference*. Springer. 2018, pp. 481–495.

[168] Monica Palmirani et al. “Legal ontology for modelling GDPR concepts and norms”. In: *Legal Knowledge and Information Systems*. IOS Press, 2018, pp. 91–100.

[169] Monica Palmirani and et al. “Pronto: Privacy ontology for legal reasoning”. In: *Electronic Government and the Information Systems Perspective: 7th International Conference (EGOVIS)*. Springer. 2018, pp. 139–152.

[170] Francesco Sovrano, Fabio Vitali, and Monica Palmirani. “Modelling GDPR-compliant explanations for trustworthy AI”. In: *Electronic Government and the Information Systems Perspective: 9th International Conference, EGOVIS 2020, Bratislava, Slovakia, September 14–17, 2020, Proceedings 9*. Springer. 2020, pp. 219–233.

[171] Florian Kammueller. “Formal modeling and analysis of data protection for GDPR compliance of IoT healthcare systems”. In: *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE. 2018, pp. 3319–3324.

[172] Forough Arabshahi et al. “Conversational neuro-symbolic commonsense reasoning”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 6. 2021, pp. 4902–4911.

[173] Chandra Bhagavatula et al. “Abductive commonsense reasoning”. In: *arXiv preprint arXiv:1908.05739* (2019).

[174] Zayne Sprague et al. “Natural language deduction with incomplete information”. In: *arXiv preprint arXiv:2211.00614* (2022).

[175] Alon Talmor et al. “Leap-of-thought: Teaching pre-trained models to systematically reason over implicit knowledge”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 20227–20237.

[176] Alon Talmor et al. “Commonsenseqa: A question answering challenge targeting commonsense knowledge”. In: *arXiv preprint arXiv:1811.00937* (2018).

[177] Mor Geva et al. “Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies”. In: *Transactions of the Association for Computational Linguistics* 9 (2021), pp. 346–361.

[178] Koustuv Sinha et al. “CLUTRR: A diagnostic benchmark for inductive reasoning from text”. In: *arXiv preprint arXiv:1908.06177* (2019).

[179] Uri Katz, Mor Geva, and Jonathan Berant. “Inferring implicit relations in complex questions with language models”. In: *arXiv preprint arXiv:2204.13778* (2022).

[180] Naila Azam et al. “Modelling Technique for GDPR-compliance: Toward a Comprehensive Solution”. In: *GLOBECOM 2023 IEEE Global Communications Conference*. IEEE. 2023, pp. 1–7.

[181] Naila Azam et al. “Data Privacy Threat Modelling for Autonomous Systems: A Survey from the GDPR’s Perspective”. In: *IEEE Transactions on Big Data* (2022).

[182] Harold Boley, Adrian Paschke, and M Omair Shafiq. “RuleML 1.0: The Overarching Specification of Web Rules.” In: *RuleML*. Springer. 2010, pp. 162–178.

[183] Ian Horrocks et al. “SWRL: A semantic web rule language combining OWL and RuleML”. In: *W3C Member submission* 21.79 (2004), pp. 1–31.

[184] Matthias Thimm. “Tweety: A comprehensive collection of java libraries for logical aspects of artificial intelligence and knowledge representation”. In: *Fourteenth International Conference on the Principles of Knowledge Representation and Reasoning*. 2014.

[185] Harshvardhan J Pandit, Declan O’Sullivan, and Dave Lewis. “GDPR data interoperability model”. In: *23rd EURAS Annual Standardisation Conference, Dublin, Ireland*. 2018.

[186] Mohamed Abomhara, Martin Gerdes, and Geir M Køien. “A stride-based threat model for telehealth systems”. In: *Norsk informasjonssikkerhetskonferanse (NISK)* 8.1 (2015), pp. 82–96.

[187] Marcelo C Sosa-Iudicissa. *Internet, Telematics, and Health*. Vol. 36. IOS Press, 1997.

[188] Sandeep Kumar Vashist, E Marion Schneider, and John HT Luong. “Commercial smartphone-based devices and smart applications for personalized healthcare monitoring and management”. In: *Diagnostics* 4.3 (2014), pp. 104–128.

[189] Christina Tikkinen-Piri, Anna Rohunen, and Jouni Markkula. “EU General Data Protection Regulation: Changes and implications for personal data collecting companies”. In: *Computer Law & Security Review* 34.1 (2018), pp. 134–153.

[190] Luciano Héctor Tamargo et al. “Time, defeasible logic and belief revision: pathways to legal dynamics”. In: *Journal of applied logics* (2021).

[191] Rao Faizan Ali et al. “Information security behavior and information security policy compliance: A systematic literature review for identifying the transformation process from noncompliance to compliance”. In: *Applied Sciences* 11.8 (2021), p. 3383.

[192] Céline Brassart Olsen. “To track or not to track? Employees’ data privacy in the age of corporate wellness, mobile health, and GDPR”. In: *International Data Privacy Law* 10.3 (2020), pp. 236–252.

[193] Irene Ioannidou and Nicolas Sklavos. “On general data protection regulation vulnerabilities and privacy issues, for wearable devices and fitness tracking applications”. In: *Cryptography* 5.4 (2021), p. 29.

[194] Marc Bourreau et al. *Google/Fitbit will monetise health data and harm consumers*. Centre for Economic Policy Research, 2020.

[195] Michael J Maher et al. “Rethinking defeasible reasoning: A scalable approach”. In: *Theory and Practice of Logic Programming* 20.4 (2020), pp. 552–586.

[196] Michael J Maher. “Propositional defeasible logic has linear complexity”. In: *Theory and Practice of Logic Programming* 1.6 (2001).

[197] Paul Voigt and Axel Von dem Bussche. “The eu general data protection regulation (gdpr)”. In: *A Practical Guide, 1st Ed., Cham: Springer International Publishing* 10.3152676 (2017), pp. 10–5555.

[198] Karen McGregor Richmond et al. “Explainable AI and law: an evidential survey”. In: *Digital Society* 3.1 (2024), p. 1.

[199] Harry Surden. “Machine learning and law: An overview”. In: *Research Handbook on Big Data Law* (2021), pp. 171–184.

[200] Kevin D Ashley and Vern R Walker. “Toward constructing evidence-based legal arguments using legal decision documents and machine learning”. In: *Proceedings of the fourteenth international conference on artificial intelligence and law*. 2013, pp. 176–180.

[201] Grigoris Antoniou et al. “Explainable Reasoning with Legal Big Data: A Layered Framework.” In: *FLAP* 9.4 (2022), pp. 1155–1170.

[202] Kwang Y Lee and Mohamed A El-Sharkawi. *Modern heuristic optimization techniques: theory and applications to power systems*. John Wiley & Sons, 2008.

[203] Donald Nute. *Defeasible deontic logic*. Vol. 263. Springer Science & Business Media, 1997.

[204] Henry Prakken and Gerard Vreeswijk. “Logics for defeasible argumentation”. In: *Handbook of philosophical logic* (2002), pp. 219–318.

[205] Guillermo Ricardo Simari. “On the Properties of the Relation between Argumentation Semantics and Argumentation Inference Operators.” In: *COMMA*. 2014.

[206] Michael J Maher and Guido Governatori. “A semantic decomposition of defeasible logics”. In: *AAAI/IAAI*. 1999, pp. 299–305.

[207] Grigoris Antoniou, Michael J. Maher, and David Billington. “Defeasible logic versus logic programming without negation as failure”. In: *The Journal of Logic Programming* 42.1 (2000), pp. 47–57.

# Appendix A

## 9.4 GConsent Ontology converted into DeLP Knowledge Base

```
r1: ConsentCompliance(X, Y) -< DataSubject(Y), DataController(X),
GivenConsent(Y, X).
r2: ~GivenConsent(Y, X) ←  ~ExplicitlyGiven(Y, X).
r3: ~ExplicitlyGiven(Y, X) ← ~ConsentStatusExplicitlyGiven(Y, X).
r4: ~ConsentStatusExplicitlyGiven(Y, X) ← ~StatusValidForProcessing(X, Y).
r5: ~StatusValidForProcessing(X, Y) ← ~ExplicitlyGiven(Y, X).
r6: ~StatusValidForProcessing(X, Y) ← ~ImplicitlyGiven(Y, X).
r7: ~ImplicitlyGiven(Y, X) ← ~IConsentStatusImplicitlyGiven(Y, X).
r8: ~StatusValidForProcessing(X, Y) ← ~GivenByDelegation(Y, X).
r9: GivenByDelegation(Y, X) ← ConsentStatusGivenByDelegation(Y, X).

%DataSubject
r10: ConsentCompliance(X, Y) -< MinorDataSubject(M), Guardians(G),
GivenConsentByGuardians(G, X).
r11: ~GivenConsentByGuardians(G, X) ← ~ConsentForProcessingPersonalData(X, M).

%Status
r12: ConsentCompliance(X, Y) -< ConsentStatus(X, Y).
r13: ConsentStatus(X, Y) -< StatusValidForProcessing(X, Y).
r14: ConsentStatus(X, Y) -< StatusInvalidForProcessing(X, Y).
r15: ~StatusInvalidForProcessing(X, Y) ← ~ConsentStatusExpired(X, Y).
r16: ~StatusInvalidForProcessing(X, Y) ← ~ConsentStatusInvalidated(X, Y).
r17: ~StatusInvalidForProcessing(X, Y) ← ~ConsentStatusNotGiven(Y, X).
r18: ~StatusInvalidForProcessing(X, Y) ← ~ConsentStatusRefused(Y, X).
r19: ~StatusInvalidForProcessing(X, Y) ← ~ConsentStatusRequested(X, Y).
r20: ~StatusInvalidForProcessing(X, Y) ← ~ConsentStatusUnknown(X, Y).
r21: ~StatusInvalidForProcessing(X, Y) ← ~ConsentStatusWithdrawn(Y, X).

%Purpose
r22: ConsentCompliance(X, Y) -<  PurposeForConsent(X, Y).
r23: ~PurposeForConsent(X, Y) ← ~SpecifySpecificInstanceOfPurpose(X, Y).
r24: ~PurposeForConsent(X, Y) ← ThirdParty(Z),
~SpecifyPurposeAssociatedWithThirdParties(X, Z).
```

```
%ValidConsent
r25: ConsentCompliance(X, Y) -< ValidConsent(X, Y).
r26: ~ValidConsent(X, Y) ← ~FreelyGiven(Y, X).
r27: ~FreelyGiven(Y, X) ← ~FreelyGivenConsentObligation(X, Y).
r28: ~ValidConsent(X, Y) ← ~Informed(X, Y).
r29: ~Informed(X, Y) ← ~InformedConsentObligation(X, Y).
r30: ~ValidConsent(X, Y) ← ~Specific(X, Y).
r31: ~Specific(X, Y) ← SpecificConsentObligation(X, Y).
r32: ~ValidConsent(X, Y) ← ~VoluntaryAndoptIn(X, Y).
r33: ~VoluntaryAndoptIn(X, Y) ← ~VoluntaryOptInConsentObligation(X, Y).
r34: ConsentCompliance(X, Y) -< ConsentActivity(X, Y),
ObligationForObtainingConsent(X, Y).
r35: ~ConsentActivity(X, Y) ← ~ObtainingConsentFromDataSubject(X, Y).
r36: ~ConsentActivity(X, Y) ← ~WithdrawingGivenConsent(Y, X).
r37: ~ObligationForObtainingConsent(X, Y) ←
~CanBeWithdrawnEasilyConsentObligation(Y, X).
r38: ~ObligationForObtainingConsent(X, Y) ←
~ClearExplanationOfProcessingConsentObligation(X, Y).
r39: ~ObligationForObtainingConsent(X, Y) ← ~ShouldBeDemonstrable(X, Y).
r40: ~ObligationForObtainingConsent(X, Y) ←
~ShouldBeDistinguishableFromOtherMatters(X, Y).
r41: ~ObligationForObtainingConsent(X, Y) ←
~NotFromSilenceOrInactivityConsentObligation(X, Y).
r42: ~ObligationForObtainingConsent(X, Y) ← ~ValidConsent(X, Y).

%PersonalData
r43: ConsentCompliance(X, Y) -< PersonalDataForConsent(Y, X).
r44: ~PersonalDataForConsent(Y, X) ← ~hasPersonalData(Y).
r45: hasPersonalData(Y) ← hasName(Y), hasContactInformation(Y).
r46: hasPersonalData(Y) ← SensitivePersonalData(Y).
r47: SensitivePersonalData(Y) ← hasHealthData(Y).

% ConsentContext
r48: ConsentCompliance(X, Y) -< ContextForConsent(X, Y).
r49: ~ContextForConsent(X, Y) ← ~SpecifyConsentLocation(X, Y).
r50: ~ContextForConsent(X, Y) ← ~SpecifyConsentMedium(X, Y).
r51: ~ContextForConsent(X, Y) ← ~SpecifyConsentTime(X, Y).
r52: ~ContextForConsent(X, Y) ← ~SpecifyConsentExpiry(X, Y).

% ConsentForProcessingPersonalData
r53: ConsentCompliance(X, Y) -< ConsentForProcessingPersonalData(X, Y).
r54: ~ConsentForProcessingPersonalData(X, Y) ← ~ConsentForDataAdaptation(X, Y).
r55: ~ConsentForProcessingPersonalData(X, Y) ← ~ConsentForDataAlignment(X, Y).
r56: ~ConsentForProcessingPersonalData(X, Y) ← ~ConsentForDataAlteration(X, Y).
r57: ~ConsentForProcessingPersonalData(X, Y) ← ~ConsentForDataCollection(X, Y).
r58: ~ConsentForProcessingPersonalData(X, Y) ← ~ConsentForDataCombination(X, Y).
```

```
r59: ~ConsentForProcessingPersonalData(X, Y) ← ~ConsentForDataConsultation(X, Y).
r60: ~ConsentForProcessingPersonalData(X, Y) ← ~ConsentForDataDestruction(X, Y).
r61: ~ConsentForProcessingPersonalData(X, Y) ←
~ConsentForDataDisclosurebyTransmission(X, Y).
r62: ~ConsentForProcessingPersonalData(X, Y) ← ~ConsentForDataDissemination(X, Y).
r63: ~ConsentForProcessingPersonalData(X, Y) ← ~DataOrganisation(X, Y).
r64: ~ConsentForProcessingPersonalData(X, Y) ← ~ConsentForDataErasure(X, Y).
r65: ~ConsentForProcessingPersonalData(X, Y) ← ~ConsentForDataRecording(X, Y).
r66: ~ConsentForProcessingPersonalData(X, Y) ← ~ConsentForDataRestriction(X, Y).
r67: ~ConsentForProcessingPersonalData(X, Y) ← ~ConsentForDataRetrieval(X, Y).
r68: ~ConsentForProcessingPersonalData(X, Y) ← ~DataSharingConsent(X, Y).
r69: ~ConsentForProcessingPersonalData(X, Y) ← ~ConsentForDataStorage(X, Y).
r70: ~ConsentForProcessingPersonalData(X, Y) ← ~ConsentForDataStructuring(X, Y).
r71: ~ConsentForProcessingPersonalData(X, Y) ← ~ConsentForDataUse(X, Y).
r72: ~ConsentForProcessingPersonalData(X, Y) ← ThirdParty(Z),
~SpecifyRoleOfThirdPartiesForProcessingData(X, Z).

%Capturing FitbitConsent
r73: ~ConsentForProcessingPersonalData(X, Y) -<
~ConsentForExternalProcessing(X, Y).
r74: ConsentForExternalProcessing(X, Y) -< ThirdParty(Z),
ProcessingForDataTransfer(X, Z).
r75: ~ConsentForExternalProcessing(X, Y) ← ~FreelyGivenConsentObligation(X, Y).
r76: ~FreelyGivenConsentObligation(X, Y) ←
~AllowSeparateConsentToBeGivenForSpecificDataProcessing(X, Y).
r77: ~ConsentForExternalProcessing(X, Y) ← ~DataSharingConsent(X, Y).
r78: ~ConsentForExternalProcessing(X, Y) ←
~SpecifySpecificInstanceOfPurpose(X, Y).
r79: ~ConsentForExternalProcessing(X, Y) ←  RightToDataErasure(X, Y).
r80: RightToDataErasure(X, Y) ← DeletePersonalInformation(X, Y).
r81: RightToDataErasure(X, Y) ← DeletionPeriodUpTo90Days(X, Y).
r82: ~CanBeWithdrawnEasilyConsentObligation(Y, X) ←
DeletePersonalInformation(X, Y), DeletionPeriodUpTo90Days(X, Y).
r83: ~ConsentForExternalProcessing(X, Y) ←
~CanBeWithdrawnEasilyConsentObligation(Y, X).
r84: ~ConsentForExternalProcessing(X, Y) ←
~ConsentForSharingSensitivePersonalData(Y, X).
r85: ConsentForSharingSensitivePersonalData(Y, X) ← hasHealthData(Y).
r86: ~ConsentForExternalProcessing(X, Y) ← ~RightToAccessPersonalData(Y, X).
r87: ~ConsentForExternalProcessing(X, Y) ← ~InformedConsentObligation(X, Y).
r88: ~InformedConsentObligation(X, Y) ← ~InformRecipientsOfPersonalData(X, Y).
r89: ~InformRecipientsOfPersonalData(X, Y) ←
~ProvideDetailsOfTransfersOfPersonalDataToThirdCountries(X, Y).
r90: ConsentCompliance(X,Y) -< DeletePersonalInformation(X, Y),
FreelyGivenConsentObligation(X, Y).
r91: ConsentCompliance(X,Y) -< ~DeletePersonalInformation(X, Y),
~ProcessingForDataTransfer(X, Z).
r92: ConsentCompliance(X,Y) -< FreelyGivenConsentObligation(X, Y),
ProcessingForDataTransfer(X, Z).
r93: ConsentForExternalProcessing(X, Y) -< PersonalData(Y),
ConsentForProcessingPersonalData(X, Y).
r94: ~ConsentForExternalProcessing(X, Y) <- ~RightForAccess(X, Y).
r95 ~RightForAccess(X, Y) <- ~GivesRightToObtainCopyOfPersonalData(X, Y).

%Facts
f1: DataSubject(user).
f2: DataController(fitbit).
f3: DeletionPeriodUpTo90Days(fitbit, user).
```