# Explainability in Artificial Intelligence

Lecture Notes

**Nadjib Lazaar**
Université Paris-Saclay
lazaar@lisn.fr

---

**Introduction**

Explainability in artificial intelligence is a crucial field that aims to make the decisions of AI models understandable to humans. With the rise of deep learning systems and complex models, understanding why a decision is made becomes fundamental for ensuring trust, transparency, and fairness in automated systems.
In this course, we will explore various approaches to explainability and introduce fundamental concepts such as minimal explanation sets. We will then focus on the QuickXplain algorithm, an efficient method for extracting a minimal subset of constraints explaining a contradiction in a given problem.

## 1 Introduction

Artificial Intelligence (AI) plays a central role today in many fields such as healthcare, finance, transportation, and security. While the performance of AI-based systems, especially those using machine learning, is impressive, their widespread adoption raises a key question: **can we trust them?**

This is where the growing need for **explainability** emerges. This notion encompasses all methods, techniques, and approaches that help understand, interpret, and justify decisions made by AI systems. Explainability has become a technical, ethical, and legal concern.

The motivations for making AI explainable are numerous:

- **Trust**: A user is more likely to accept a decision if they can understand its justification.
- **Verification and validation**: Experts must be able to analyze model behavior, especially in case of malfunctions.

- **Accountability**: In critical contexts (justice, medicine, autonomous transport), it is necessary to identify the causes of a decision.
- **Regulatory compliance**: Legal texts such as the GDPR or the European AI Act require transparency and control guarantees.

This lecture note aims to present the main approaches to explainability in AI, from general methods to logical foundations based on constraint system consistency. We will cover:

- motivations and legal framework (GDPR, AI Act);
- the major families of explainability methods;
- logic-based and constraint programming approaches: inconsistency, MUS/MCS/MSS, and the QuickXplain algorithm.

# 2 Regulatory Framework: GDPR and AI Act

Explainability in AI is not just a technical need: it is also a legal requirement. Two major regulations currently govern AI usage in Europe: the **General Data Protection Regulation (GDPR)** and the **AI Act**, which is still being formally adopted but already highly influential.

## 2.1 GDPR and the Right to Explanation

The GDPR, in effect since 2018, regulates the automated processing of personal data. Article 22 introduces a significant restriction: an individual cannot be subject to a fully automated decision producing legal effects, except under strict conditions.

One of the most discussed contributions of the GDPR is the idea of a **right to explanation**, although it is not explicitly stated. Article 15(1)(h) provides that the data subject may obtain "meaningful information about the logic involved" in an automated process.

**Implications for AI**:

- Obligation to disclose the purpose and logic used.
- Need to provide safeguards against bias and discrimination.
- Importance of interpretable or explainable methods.

## 2.2 The AI Act: Toward Specific AI Regulation

The **AI Act** is a European regulation under finalization (consolidated version by the end of 2023), aiming to regulate the development and deployment of AI systems in Europe. It introduces a risk-based classification: *minimal, limited, high, unacceptable.*

**For high-risk systems**, strict requirements are imposed, such as:

1. Clear and up-to-date **technical documentation**;
2. High-quality **training data**;
3. **Decision traceability**;
4. **Explainability and transparency** of results;
5. **Human oversight**;
6. **Robustness, accuracy, and cybersecurity**;
7. **Risk management** throughout the lifecycle.

### 2.3 Toward Normative Explainability

Texts like GDPR and the AI Act do not define what constitutes a good explanation but establish requirements of *transparency*, *understandability*, *justification*, and *accountability*. It is up to research to propose suitable methods depending on the context, target audience, and associated risks.

## 3 Explainability Methods in Artificial Intelligence

Explainability approaches aim to make AI models understandable to humans. Two main dimensions can be distinguished: **when** the explanation is produced (before or after learning), and **the level** of the explanation (global or local).

### 3.1 Typology of Methods

**Intrinsic explainability** Some models are inherently interpretable. This is the case for decision trees, logical rules, or linear models. The explanation is directly linked to the model's structure.

**Post-hoc explainability** When a model is complex (deep neural networks, random forests, etc.), explanation methods are needed after training. The goal is to explain a specific decision without modifying the model.

#### Local vs. Global Explanation

- **Local**: explain a specific prediction (e.g., why did this patient receive this diagnosis?).
- **Global**: understand the model's overall behavior (e.g., which features are most important?).

### 3.2 Some Classic Methods

- **LIME** (Local Interpretable Model-agnostic Explanations): builds a simple local model around the prediction to explain.
- **SHAP** (SHapley Additive exPlanations): uses game theory to assign importance to each feature.
- **Counterfactuals**: shows what would need to change for a different prediction to occur.
- **Saliency maps** (in computer vision): highlight areas of an image that influenced the decision.

### 3.3 Limitations of Classical Approaches

These methods are useful but have limitations:

- Lack of formal rigor;
- Difficulty in verifying explanation validity or completeness;
- Sometimes hard to interpret for non-experts;
- Sensitivity to perturbations (instability).

### 3.4 Toward Logic-Based Explainability

In response to these limitations, another approach is emerging: **logical or symbolic explainability**. It is based on formal systems (propositional logic, constraint programming, ontologies) to produce explanations that are verifiable, auditable, and often more understandable.

These methods can explain **inconsistencies** in a system (e.g., constraints that cannot be satisfied together), by identifying minimal parts responsible for the issue.

This framework includes the concepts of **MUS (Minimal Unsatisfiable Subsets)**, **MCS (Minimal Correction Subsets)**, **MSS (Maximal Satisfiable Subsets)**, and the **QuickXplain** algorithm, which we will explore in the following sections.

## 4 Explanation by Inconsistency: MUS, MCS, and MSS

In the context of explainable AI, especially for symbolic or constraint-based systems, a powerful approach is to explain an observed **inconsistency** in a system. An inconsistency arises when a set of constraints or knowledge is **unsatisfiable**, meaning that there exists no solution that satisfies the entire set.

To understand *why* such an inconsistency exists, we can search for the subsets of constraints responsible. This leads to the key notions of:

- **MUS**: *Minimal Unsatisfiable Subsets*
- **MCS**: *Minimal Correction Subsets*
- **MSS**: *Maximal Satisfiable Subsets*

These objects enable the production of **minimal explanations** of the inconsistency.

## 4.1 Basic Notation

- Let $C$ be a finite set of constraints or formulas.
- $C$ is said to be **unsatisfiable** if there exists no assignment that satisfies all the constraints in $C$.
- We write $\mathsf{SAT}(S)$ to denote that the subset $S \subseteq C$ is satisfiable.

## 4.2 Definition of a MUS

**Definition 1** (MUS)**.** *Let $C$ be a set of constraints such that $C$ is unsatisfiable. A subset $M \subseteq C$ is a **MUS** (Minimal Unsatisfiable Subset) if:*

- *$M$ is unsatisfiable: $\mathsf{SAT}(M) = \mathsf{false}$*
- *$\forall c \in M,\ \mathsf{SAT}(M \setminus \{c\}) = \mathsf{true}$ (minimality)*

**Intuition:** A MUS is a minimal explanation of the inconsistency. Removing even a single constraint is enough to resolve it.

**Usefulness:** Provides the user with the "minimal cause" of a conflict between constraints (e.g., in a recommendation system, diagnosis, planning, etc.)

## 4.3 Definition of a MCS

**Definition 2** (MCS)**.** *A subset $M \subseteq C$ is a **MCS** (Minimal Correction Subset) if:*

- *$\mathsf{SAT}(C \setminus M) = \mathsf{true}$ (correction)*
- *$\forall M' \subset M,\ \mathsf{SAT}(C \setminus M') = \mathsf{false}$ (minimality)*

**Intuition:** A MCS is a minimal set of constraints that must be **removed** to restore consistency.

**Usefulness:** In knowledge engineering or diagnosis, MCSs indicate which assumptions or rules must be removed to fix a conflict.

## 4.4 Definition of a MSS

**Definition 3** (MSS)**.** *A subset $S \subseteq C$ is a **MSS** (Maximal Satisfiable Subset) if:*

- $\mathsf{SAT}(S) = \mathsf{true}$
- $\forall c \in C \setminus S,\ \mathsf{SAT}(S \cup \{c\}) = \mathsf{false}$ *(maximality)*

**Intuition:** An MSS is a maximal subset of $C$ that is still satisfiable. Adding any remaining constraint causes inconsistency.

**Usefulness:** MSSs are useful for exploring maximally consistent configurations, for example in constraint-based planning or consistent suggestions in a decision support system.

## 4.5 Relationships Between MUS, MCS, and MSS

There is a strong relationship between these three notions:

- Every **MUS** is the complement of at least one **MCS**: if $M$ is an MCS, then $C \setminus M$ is an MSS, and its complement contains a MUS.
- Conversely, every **MSS** is the complement of a MCS.
- In practice, MUSs can be generated from MCSs (and vice versa), although this can be costly.

## 4.6 Simple Example

Let $C = \{c_1, c_2, c_3\}$ with:

$$c_1 : x > 0$$
$$c_2 : x < 5$$
$$c_3 : x < -1$$

The set $C$ is unsatisfiable. We can identify:

- $MUS = \{c_1, c_3\}$: this subset is minimally unsatisfiable.
- $MCS = \{c_3\}$: removing $c_3$ restores consistency.
- $MSS = \{c_1, c_2\}$: a maximal consistent subset.

### 4.7 Role in Explainability

MUS, MCS, and MSS allow us to:

- produce **minimal and understandable explanations** of a conflict,
- identify the responsible constraints (MUS),
- suggest solutions (MCS to remove or MSS to keep),
- formalize explanation algorithms such as QuickXplain.

## 5 The QuickXplain Algorithm

To efficiently identify a MUS, the **QuickXplain** algorithm computes a minimal subset of constraints responsible for the inconsistency, without enumerating all possible subsets.

### 5.1 General Principle

QuickXplain relies on a **divide-and-conquer** approach that:

- avoids testing all combinations of constraints;
- uses a **satisfiability oracle** to check the consistency of a set;
- returns a **minimal subset** $X' \subseteq X$ such that $X' \cup B$ is unsatisfiable.

### 5.2 Assumptions

- The set $C = X \cup B$ is unsatisfiable.
- $B$ is a set of constraints assumed to already be consistent.
- $X$ contains the constraints suspected of being responsible for the inconsistency.

## 5.3 Algorithm Pseudocode

**Algorithm 1** QuickXplain Algorithm

**Input**: $X$ (set of constraints), $B$ (set of already verified constraints)
**Output**: $X'$ (minimal subset of $X$ such that $X' \cup B$ is inconsistent)
**if** $\neg\mathsf{SAT}(B)$ **then**
  **return** $\emptyset$ {If $B$ is already inconsistent, return empty set}
**end if**
**if** $|X| = 1$ **then**
  **return** $X$ {If $X$ is atomic, return $X$}
**else**
  $X_1, X_2 \leftarrow \text{split}(X)$ {Split $X$ into two subsets $X_1$ and $X_2$}
  $X_1' \leftarrow \text{QuickXplain}(X_1,\ B \cup X_2)$
  $X_2' \leftarrow \text{QuickXplain}(X_2,\ B \cup X_1')$
  **return** $X_1' \cup X_2'$
**end if**

## 5.4 How It Works

The algorithm works recursively by reducing the problem:

- It starts by checking whether $B$ is already inconsistent;
- If there is only one constraint left, it constitutes the minimal explanation;
- Otherwise, $X$ is split into two parts, and the inconsistency is explained by testing each half in the context of the other.

## 5.5 Complexity

The complexity depends on the cost of satisfiability oracle calls. In the worst case, QuickXplain makes $O(n \log n)$ calls, which is much more efficient than exhaustive enumeration.

## 5.6 Advantages

- Does not require generating all MUSs;
- Produces a minimal explanation consistent with the original set;
- Well-suited for user interaction, as the explanations are progressive and understandable.

## 5.7 Applications

- Knowledge debugging (ontologies, rules, expert systems);
- System planning or configuration;
- Diagnosis, explaining conflicts in decision systems.