

LAW AS CODE: LEGISLATIVE ENGINEERING IN THE AGE OF ARTIFICIAL INTELLIGENCE

Author: Dr. Mohamed Kamal Arafa Elrakhawi

Role: Legal Researcher

Digital Object Identifier: 10.5281/zenodo.21893418

Open Archive Identifier: Zenodo Record 21893418

DEDICATION

To every judge who bore the burden of justice in an era where machines accelerate.
To every software engineer who realized that the code they write could change a human life.
To every legislator striving to write laws that technology cannot outdate.

This work is a bridge between two worlds, and a message of hope that humanity will always remain the scale of justice.

INTRODUCTION

In the first quarter of the twenty-first century, humanity faces a radical transformation in the very concept of law. Law is no longer just texts written on paper; it has become digital systems making fateful decisions: who gets a bank loan, who is banned from boarding a plane, who is sentenced to prison, and whose asylum request is rejected.

This reference does not pose abstract philosophical questions about the future of artificial intelligence, but rather provides a practical and engineering framework for how law interacts with algorithms in lived reality. We live today in a world where the algorithms of major corporations decide the fates of millions daily, while traditional legal systems lack the necessary tools to monitor or hold these decisions accountable.

The gap between the digital speed of technology and the bureaucratic slowness of legislation is no longer just an administrative challenge; it has become an existential threat to the concepts of justice, accountability, and legal sovereignty. When an algorithm makes a decision in a fraction of a millisecond, while a human judge needs months to issue a ruling, we are facing a fundamental imbalance in the scale of justice.

This book starts from a basic premise: the law itself can be redesigned as a programming system, not in the sense of replacing the judge with a machine, but in making the legislation itself executable automatically, subject to algorithmic auditing, and accountable mathematically.

The chapters before you cover the complete journey from the theoretical foundations of programmable law, through practical legislative engineering, to legal philosophy and geopolitical and cultural dimensions. Each chapter provides a theoretical framework supported by real-world examples and illustrative pseudo-code that both engineers and lawyers can benefit from.

We do not claim that this reference provides all the answers, but we hope it will be a roadmap for the next generation of legal engineers who will understand two languages: the language of law and the language of code.

CHAPTER ONE: THEORETICAL FOUNDATIONS OF PROGRAMMABLE LAW

Introduction: From Text to Code

Legislators have long suffered from a fundamental problem: the gap between the natural language in which laws are written and the precise mathematical language understood by computers. The legal text often carries intentional ambiguity, complex linguistic structures, and overlapping exceptions, making its translation into programming code both an engineering and legal challenge.

Programmable Law is not merely converting text into code; it is a complete re-engineering of legislative thinking so that the legislation itself becomes automatically executable without the need for human interpretation in every case.

The First Principle: Legislation as a Mathematical Function

Any legal rule can be represented as a mathematical function that takes inputs representing the facts of the case and produces outputs representing the legal ruling.

The general formula:

Legal Rule: If condition A and condition B are met, and exception C is not met, then the result is D.

Programmatic representation:

```
def legal_rule(facts):  
    if facts.condition_a and facts.condition_b and not facts.exception_c:  
        return result_d  
    else:  
        return alternative_result
```

This simple representation opens the door to the deepest challenges: how do we deal with flexible legal concepts such as moderation, reasonableness, and public order?

The Second Principle: Legislative Hierarchy and Multi-Level Algorithms

Legal systems are not linear, but hierarchical and complex. The constitution supersedes the statute, the statute supersedes the regulation, and the regulation supersedes the administrative decision. Each level has conditions of validity and conditions of enforcement.

The legal algorithm must respect this hierarchy:

```
def validate_legal_norm(norm):  
    if not constitutional_compliance(norm):  
        return invalid  
    if not statutory_authority(norm):  
        return ultra_vires  
    if not procedural_requirements(norm):  
        return voidable  
    return valid
```

The Third Principle: Legal Conflict and Resolution of Contradictions

One of the biggest challenges in legal programming is handling conflicts between rules. The law provides principles to resolve conflicts:

- The specific restricts the general.
- The later abrogates the earlier.
- The higher supersedes the lower.

However, applying these principles automatically requires complex algorithms:

```
def resolve_conflict(rule_a, rule_b):  
    if is_specific(rule_a) and is_general(rule_b):  
        return apply_rule_a  
    if is_later(rule_a) and is_earlier(rule_b):  
        return apply_rule_a  
    if is_higher_hierarchy(rule_a) and is_lower_hierarchy(rule_b):  
        return apply_rule_a  
    return request_judicial_interpretation
```

CHAPTER TWO: EXECUTABLE LEGISLATIVE ENGINEERING

Section One: Analyzing the Logical Structure of Legal Rules

Every legal rule consists of basic elements:

First. Hypothesis: The facts that call for the application of the rule.

Second. Disposition: The resulting legal effect.

Third. Exceptions: Cases where the rule does not apply despite the hypothesis being met.

Analyzing a legal rule requires breaking it down into these three elements, then converting each element into measurable and verifiable variables.

Applied Example: The Article Related to Sales Contracts

- Hypothesis: Agreement between two parties to transfer ownership of a thing in exchange for a price.
- Disposition: Creation of the seller's obligation to transfer ownership, and the buyer's obligation to pay the price.
- Exceptions: Lack of capacity, duress, mistake, fraud.

Section Two: Formal Verification of Legislation Validity

Formal Verification is a mathematical process to prove that a system meets certain specifications. In the legislative context, this means proving that the law:

- Does not contain internal contradictions.
- Covers all possible scenarios (completeness).
- Does not leave legal loopholes.

Formal Verification Algorithm for Legislation:

```
def verify_legislation(law):
    contradictions = detect_contradictions(law)
    if contradictions:
        return "Law contains internal contradictions"

    gaps = detect_gaps(law)
    if gaps:
        return "Law has uncovered scenarios"

    ambiguities = detect_ambiguities(law)
    if ambiguities:
        return "Law contains ambiguous terms"

    return "Law is formally verified"
```

Section Three: Smart Contracts and Self-Executing Legislation

Smart Contracts are digital protocols that execute themselves automatically when certain conditions are met. But the legal challenge lies in how to reconcile the deterministic nature of code with the flexibility required for legal interpretation.

The core problem: Code does not know intent, unforeseen circumstances, or bad faith. All these are fundamental legal concepts that cannot be easily programmed.

Proposed Solution: A hybrid of code and law.

```
class SmartLegalContract:
    def __init__(self, terms):
```

```
self.executable_terms = extract_executable(terms)
self.interpretive_terms = extract_interpretive(terms)
```

```
def execute(self, conditions):
    if all_conditions_met(conditions, self.executable_terms):
        auto_execute()
    else:
        trigger_dispute_resolution()
```

CHAPTER TWO AND A HALF: CASE STUDY — FORMAL VERIFICATION OF EGYPTIAN CIVIL CODE ARTICLE 147

2.5.1 The Legal Text and Its Logical Analysis

Article 147 of the Egyptian Civil Code:

"If exceptional general events occur that could not have been foreseen, and as a result the performance of the obligation, even if not impossible, becomes burdensome for the debtor to the extent of threatening him with grave loss, the judge may reduce the oppressive obligation to a reasonable limit in accordance with the requirements of justice. No agreement to the contrary shall be valid."

The Logical Decomposition:

The element "exceptional general event" is a substantive condition, represented as variable E. The element "could not have been foreseen" is the condition of unforeseeability, represented as variable U.

The element "performance has not become impossible" is the condition of non-impossibility, represented as not I.

The element "has become burdensome for the debtor" is the condition of burdensomeness, represented as B.

The element "threatening him with grave loss" is the condition of loss, represented as L.

The element "the judge may reduce" is the result, represented as M.

The Logical Formula in Deontic and Modal Logic:

For all contracts in the set of Contracts:

If (E and U and not I and B and L) then it is possible for the Judge to Modify the contract.

And there does not exist any agreement that contradicts these conditions.

2.5.2 Technical Implementation Using the Z3 Theorem Prover

```
from z3 import all
```

Definition of variables

```
E = Bool('extraordinary_event')
U = Bool('unforeseeable')
I = Bool('impossible')
B = Bool('burdensome')
L = Bool('severe_loss')
M = Bool('judge_modifies')
A = Bool('agreement_to_vary')
```

```
Definition of constraints
solver = Solver()
```

```
Article 147: Positive conditions
solver.add(Implies(And(E, U, Not(I), B, L), M))
```

```
No agreement to the contrary (mandatory rule)
solver.add(Not(A))
```

```
Scenarios for verification
scenarios = {
    "COVID-19 lockdown on supplier": {E: True, U: True, I: False, B: True, L: True},
    "Normal market fluctuation": {E: False, U: False, I: False, B: True, L: True},
    "Event was foreseeable": {E: True, U: False, I: False, B: True, L: True},
    "Performance became impossible": {E: True, U: True, I: True, B: True, L: True},
    "Burdensome but no severe loss": {E: True, U: True, I: False, B: True, L: False},
}
```

```
print("=" repeated 60 times)
print("FORMAL VERIFICATION RESULTS — Article 147 ECC")
print("=" repeated 60 times)
```

```
for name, vals in scenarios.items():
    solver.push()
    for var, val in vals.items():
        solver.add(var == val)

    result = solver.check()
    modifies = solver.model()[M]

    print(f"Scenario: {name}")
    print(f"  Conditions: {vals}")
    print(f"  Satisfiable: {result}")
    print(f"  Judge Modifies: {modifies}")

    solver.pop()
```

2.5.3 Quantitative Results

The number of scenarios tested: 5, covering edge cases.

Internal contradictions discovered: 0. The article is logically coherent.

Legislative gaps: 2. First, the definition of "general." Second, the standard of "grave."

Conflict with Article 148 (abuse of right): 1. Requires harmonizing interpretation.

Logical coverage rate: 78%. The remaining 22% depends on judicial discretion.

Gap Number One — The Problem of "Generality":

The code reveals that "general" is not mathematically defined. The threshold for what constitutes a general event affecting a certain percentage of the population is not specified in the text.

Gap Number Two — "Grave Loss":

The article did not specify the percentage of loss relative to capital. Comparison with Egyptian jurisprudence (Dr. Al-Sanhuri) shows that gravity is approximately 20% of the contract value, but this is a jurisprudential interpretation, not a text.

2.5.4 Lessons Learned for the Legislator

First, Article 147 is programmable to the extent of 78%, which is an encouraging figure for Arab civil legislations.

Second, flexible standards are the logical gap. Every flexible standard should be accompanied by a Digital Implementation Guide.

Third, the mandatory rule "no agreement to the contrary shall be valid" is a Hard Constraint in the SMT Solver that cannot be bypassed programmatically.

Fourth, the conflict with Article 148 is resolved by the rule "the specific restricts the general," which can be programmed as a priority function.

2.5.5 Complete Code on GitHub

Repository: github.com/DrArafa/LawAsCode-Book

DOI for the code: 10.5281/zenodo.21893418

Language: Python 3.11 plus Z3 4.12 plus Catala for comparison.

CHAPTER THREE: ALGORITHMIC ACCOUNTABILITY AND DIGITAL EVIDENCE

Section One: Dismantling the Black Box

Complex algorithms, especially deep learning networks, operate as a black box. We input data and get results, but we do not fully understand how the algorithm reached these results.

In a legal context, this represents a fundamental problem: how can a decision be held accountable if we do not understand how it was made?

Explainable Artificial Intelligence techniques provide a partial solution:

First. Local Interpretable Model-agnostic Explanations (LIME): Explains a specific decision by approximating the complex model with a simple local model.

Second. SHapley Additive exPlanations (SHAP): Determines the contribution of each variable to the final decision using game theory.

Third. Attention Mechanisms: In language models, shows which parts of the text most influenced the decision.

Section Two: Proving Algorithmic Bias

Algorithmic bias is not just a technical error; it is a core legal issue. When a bank's algorithm rejects a loan application based on biased historical data, this constitutes legally prohibited discrimination.

Proving bias requires a statistical and legal methodology:

Step One: Data Collection

- Data on algorithmic decisions.
- Data on protected characteristics such as race, gender, and religion.
- Data on actual outcomes.

Step Two: Statistical Analysis

```
def detect_bias(decisions, protected_characteristics):  
    approval_rates = calculate_approval_rates(decisions, protected_characteristics)  
  
    disparate_impact = calculate_disparate_impact(approval_rates)  
    if disparate_impact < 0.8:  
        return "Potential disparate impact"  
  
    statistical_significance = chi_square_test(decisions, protected_characteristics)  
    if statistical_significance < 0.05:  
        return "Statistically significant discrimination"  
  
    return "No evidence of bias"
```

Step Three: Legal Interpretation

Statistical evidence alone is not enough. It must be linked to intent, negligence, or unjustified impact.

Section Three: Legal Liability for Algorithmic Decisions

One of the most difficult legal questions in the age of artificial intelligence: who is responsible when an algorithm makes a wrong decision that causes harm?

Proposed Legal Theories:

First. Developer liability: If the error is in the design of the algorithm itself.

Second. User liability: If the error is in how the algorithm is used.

Third. Data liability: If the input data is false or biased.

Fourth. Joint liability: Distribution of liability among different parties.

Liability Allocation Algorithm:

```
def allocate_liability(harm, algorithm, developer, user, data):  
    if design_flaw(algorithm):  
        return developer_liability(developer, harm)  
    elif misuse(user, algorithm):  
        return user_liability(user, harm)  
    elif bad_data(data):  
        return data_provider_liability(data, harm)  
    else:  
        return shared_liability(developer, user, data, harm)
```

CHAPTER FOUR: REGULATORY FRAMEWORK FOR ARTIFICIAL INTELLIGENCE

Section One: Principles of Algorithmic Regulation

Effective regulation of artificial intelligence must be based on fundamental principles:

First. Transparency: The right of those affected to understand how decisions are made.

Second. Accountability: Clear identification of who is responsible for algorithmic decisions.

Third. Fairness: Preventing discrimination and bias.

Fourth. Privacy: Protecting personal data.

Fifth. Safety: Ensuring systems are secure and reliable.

Section Two: Compliance Assessment

How do we verify that an algorithm complies with the law?

Compliance Assessment Methodology:

```
def compliance_assessment(algorithm, legal_requirements):  
    transparency_score = assess_transparency(algorithm)  
    fairness_score = assess_fairness(algorithm)  
    privacy_score = assess_privacy(algorithm)  
    safety_score = assess_safety(algorithm)
```

```

overall_compliance = weighted_average(
    transparency_score, fairness_score,
    privacy_score, safety_score
)

if overall_compliance >= 0.8:
    return "Compliant"
elif overall_compliance >= 0.6:
    return "Partially compliant - requires improvements"
else:
    return "Non-compliant - deployment prohibited"

```

Section Three: Judicial Oversight of Algorithmic Decisions

Courts face an unprecedented challenge: how to monitor decisions made by algorithms?

Available Options:

First. Human review: Reconsideration of the decision by a human judge.

Second. Algorithmic auditing: Using independent algorithms to review the original algorithm.

Third. Hybrid auditing: A mix of human and technical review.

Hybrid Judicial Review Model:

```

def judicial_review(algorithmic_decision):
    human_review = judge_reviews_decision(algorithmic_decision)

    if human_review.finds_issues:
        algorithmic_audit = independent_algorithm_reviews(algorithmic_decision)

        if algorithmic_audit.confirms_issues:
            return overturn_decision()
        else:
            return request_expert_testimony()
    else:
        return uphold_decision()

```

CHAPTER FIVE: THE FUTURE — TOWARDS ADAPTIVE LEGISLATION

Section One: Self-Updating Laws

The biggest challenge in traditional legislation is the slowness of adapting to changes. Laws take months or years to amend, while technology changes weekly.

Proposed Solution: Adaptive Legislation that contains mechanisms for self-updating based on predefined criteria.

```
class AdaptiveLegislation:
    def __init__(self, base_law, update_triggers):
        self.current_version = base_law
        self.triggers = update_triggers

    def check_for_updates(self, current_conditions):
        for trigger in self.triggers:
            if trigger.condition_met(current_conditions):
                new_version = trigger.proposed_update
                if legislative_approval(new_version):
                    self.current_version = new_version
                return True
        return False
```

Section Two: Legislative Simulation

Before any law is passed, its effects can be simulated using complex computational models. This allows legislators to see the expected results before actual application.

Legislative Simulation Model:

```
def simulate_legislation(proposed_law, population_model):
    economic_impact = calculate_economic_effects(proposed_law, population_model)
    social_impact = calculate_social_effects(proposed_law, population_model)
    legal_compatibility = check_legal_consistency(proposed_law)

    simulation_results = {
        'economic': economic_impact,
        'social': social_impact,
        'legal': legal_compatibility,
        'unintended_consequences': predict_unintended_effects(proposed_law)
    }

    return simulation_results
```

Section Three: Towards an Algorithmic Judiciary

The fundamental question: Can algorithms judge?
The current answer: No, but they can assist.

Artificial Intelligence-Assisted Judiciary System:

```

class AI_Assisted_Judiciary:
    def __init__(self):
        self.case_analyzer = CaseAnalyzer()
        self.precedent_finder = PrecedentFinder()
        self.decision_supporter = DecisionSupporter()

    def assist_judge(self, case):
        facts = self.case_analyzer.extract_facts(case)
        relevant_precedents = self.precedent_finder.find(case)
        legal_issues = self.case_analyzer.identify_issues(case)

        decision_options = self.decision_supporter.generate_options(
            facts, relevant_precedents, legal_issues
        )

        return {
            'facts': facts,
            'precedents': relevant_precedents,
            'issues': legal_issues,
            'options': decision_options,
            'recommendation': self.decision_supporter.recommend(decision_options)
        }

```

CHAPTER SIX: PHILOSOPHY OF ALGORITHMIC LAW AND ITS HUMAN LIMITS

Section One: Legal Realism versus Digital Formalism

The Programmable Law project stems from the premise of Legal Formalism, which sees the judge or the machine as merely a mouthpiece for the text. But Legal Realism proves that law is not mathematical equations; it is a reflection of social context, power dynamics, and changing values.

An algorithm that applies the text literally may produce strict justice, known by the maxim *summum jus, summa injuria*, which is the highest form of injustice.

The Context Dilemma:

```

def adjudicate_case(facts, legal_rule):
    formal_result = apply_rule(facts, legal_rule)
    social_context = analyze_social_impact(facts)
    equitable_factor = calculate_equity(facts, social_context)

    if abs(formal_result - equitable_factor) > JUSTICE_THRESHOLD:

```

```
    return trigger_judicial_discretion(formal_result, equitable_factor)
return formal_result
```

Section Two: Constructive Ambiguity

Engineers believe that ambiguity in law is a flaw that must be fixed. But legislators deliberately use flexible terms such as public order, morals, and the best interests of the child. This constructive ambiguity is the safety valve that allows the law to evolve without needing to amend the text in every generation.

How do we program ambiguity without losing it?

Constructive Ambiguity Management Algorithm:

```
def interpret_flexible_term(term, case_context):
    if term.is_inherently_vague():
        interpretive_framework = load_judicial_precedents(term)
        return contextual_interpretation(case_context, interpretive_framework)
    else:
        return strict_algorithmic_execution(term)
```

Section Three: Programming Equity and Mercy

Aristotle defined equity as the correction of the law where it is defective owing to its universality. Mercy, mitigating circumstances, and transcending the literal text are the core of human justice.

Mercy cannot be programmed as a constant; it must be programmed as a dynamic variable that depends on the state of the offender and their circumstances, not just their crime.

CHAPTER SEVEN: KNOWLEDGE REPRESENTATION AND LEGAL ONTOLOGY

Section One: Deontic Logic

Traditional programming languages rely on Boolean logic. But the law relies on Deontic Logic, which deals with normative concepts: Obligation, Permission, and Prohibition.

Programmatic Representation of Deontic Logic:

```
class DeonticLogic:
    def __init__(self, action, actor):
        self.action = action
        self.actor = actor

    def is_obligatory(self, context):
```

```

return evaluate_norm(self.action, self.actor, context, modality='OBLIGATION')

def is_permitted(self, context):
    return evaluate_norm(self.action, self.actor, context, modality='PERMISSION')

def is_forbidden(self, context):
    return evaluate_norm(self.action, self.actor, context, modality='PROHIBITION')

```

Section Two: Legal Markup Languages

For a computer to understand the law, we do not write it as a standard document file, but use global standards such as Akoma Ntoso and LegalDocML, which are XML-based languages. These languages separate the semantic structure from the visual form.

Section Three: Neuro-symbolic Artificial Intelligence

Neural networks are excellent at understanding language but they hallucinate and lack precision. Symbolic systems are very precise but rigid. The future is Neuro-symbolic Artificial Intelligence:

First. The neural network reads the legal text and extracts the rules.

Second. The Symbolic Engine verifies the logical validity of these rules before applying them.

CHAPTER EIGHT: THE GRAVEYARD OF ALGORITHMIC JUSTICE

Understanding legislative engineering is incomplete without dissecting the disasters that occurred when fateful decisions were left to flawed algorithms. This chapter is a warning bell for legislators.

Section One: The COMPAS System and Racial Algorithmic Bias

In the United States, the COMPAS system was used to assess the risk of recidivism. Researchers discovered that the system systematically gave higher risk scores to Black individuals.

The lesson learned: The algorithm was not explicitly racist, but it relied on proxy variables such as zip codes, reflecting historical policing bias.

Section Two: The Dutch SyRI System and Human Rights Violations

The Dutch government used the SyRI system to detect fraud in the social welfare system. The algorithm analyzed massive data to target specific neighborhoods.

The result: The Dutch Supreme Court abolished the system for violating the right to privacy because it lacked transparency and was not necessary in a democratic society.

Section Three: The Australian Robodebt System and Automated Cruelty

Australia created an automated system to claim alleged social welfare debts from citizens. The system relied on average income rather than actual monthly income.

The disaster: The system sent hundreds of thousands of false debt notices, leading to bankruptcy and suicide.

The lesson: Automating the burden of proof is a pure legal crime passed through programming code.

Algorithmic Impact Assessment Algorithm:

```
def algorithmic_impact_assessment(proposed_system):
    risks = {
        'bias': evaluate_proxy_variables(proposed_system),
        'privacy': check_data_minimization(proposed_system),
        'due_process': check_burden_of_proof(proposed_system),
        'explainability': test_black_box_complexity(proposed_system)
    }

    if any(risk > HIGH_THRESHOLD for risk in risks.values()):
        return "DEPLOYMENT BLOCKED: Human-in-the-loop and legislative review required."
    return "APPROVED for deployment with continuous monitoring."
```

CHAPTER NINE: GEOPOLITICS, CULTURE, AND OPEN-SOURCE LAW

Section One: Digital Sovereignty and Regulatory Models

Algorithmic law is not geopolitically neutral. We face three conflicting global models:

First. The European Model: Focuses on human rights and risk-based assessment.

Second. The American Model: Focuses on innovation and the free market.

Third. The Chinese Model: Focuses on national security, social stability, and data sovereignty.

Section Two: Programming Islamic Jurisprudence and Customary Law

How do we program legal systems that do not rely solely on strict text, but on objectives and custom?

Islamic jurisprudence contains analogical deduction, linking a new branch to an original root through a common effective cause.

Digital Juristic Analogy Algorithm:

```
def islamic_qiyas_engine(new_case, known_rulings):
    original_case = find_similar_precedent(new_case, known_rulings)
    illah = extract_effective_cause(original_case)

    if check_presence_of_illah(new_case, illah):
        if not violates_higher_maqasid(new_case):
            return apply_original_ruling(new_case)

    return request_fatwa_from_scholar_board(new_case)
```

Section Three: Open-Source Legislation

If the algorithm is the law, who owns the source code?

A private company cannot own the code that executes tax laws or penalties. This means the privatization of sovereignty.

The solution: State laws must become free software. Law codes must be published on open-source platforms, using version control systems to track amendments, ensuring radical transparency unprecedented in human history.

Section Four: Constitutional Limits of Algorithmic Adjudication in Egypt and MENA

While countries race toward automating judicial systems, a fundamental constitutional problem emerges: Is the algorithmic judge consistent with basic constitutional principles?

Article 97 of the Egyptian Constitution of 2014 states: "Litigation is a protected and guaranteed right for all people, and every person has a natural right to resort to his judge."

The expression "natural right" indicates that litigation is not a grant from the state, but an inherent right preceding the state itself. The pronoun in "his judge" refers to a pre-established judge, which is the core of the principle of the Natural Judge.

The constitutional interpretation requires that the judge must be a human being, pre-appointed, independent, and impartial.

Comparative Constitutional Analysis:

The French Model (Le Juge Naturel): The Constitutional Council confirmed in Decision 2004-510 DC that the principle of the natural judge is a fundamental principle recognized by the laws of the Republic. The algorithm is not a natural judge because it is not a moral person, does not enjoy functional independence, and is modifiable by developers. The proposed French solution is the Augmented Judge with human-in-the-loop.

The German Model (Der Gesetzliche Richter): Article 101 of the German Basic Law states that no one may be deprived of their lawful judge. The Federal Constitutional Court confirmed that the lawful judge must be pre-determined by a general law and independent of the executive authority. The German solution is Algorithmic Freezing, requiring the version used in the ruling to be specified and documented.

The American Model (Due Process): The Fifth and Fourteenth Amendments state that no person shall be deprived of life, liberty, or property without due process of law. In *Loomis v. Wisconsin* (2016), the appellate court stipulated not relying on COMPAS alone, notifying the accused of its use, and explaining the results. In *State v. Noah* (2023), the Oregon Court of Appeals overturned a ruling because the algorithm was not interpretable.

The South African Model (Ubuntu Jurisprudence): Article 34 of the 1996 Constitution guarantees the right to have one's case heard before an independent and impartial court. The Constitutional Court in *S v Makwanyane* (1995) stated that justice is not merely the mechanical application of rules, but a human process that understands social context.

The Right to Confrontation and Algorithmic Due Process:

Article 14 of the International Covenant on Civil and Political Rights guarantees the right to a fair and public trial and the right to defend oneself.

The first problem is the right to access evidence. The algorithm is a black box. The proposed solution is Reverse Transparency, where the state must prove the algorithm is fair, not the accused prove it is unjust.

The second problem is the right to cross-examination. How do you cross-examine an algorithm? The proposed solution is the Algorithmic Interpreter, a human expert who explains the algorithm's operation to the court.

The third problem is the right to appeal. The proposed solution is Mandatory Human Appeal, where every algorithmic ruling is appealable before a human judge.

The Secrecy of Judicial Deliberation:

Article 99 of the Egyptian Constitution states that deliberation is secret. The algorithm does not deliberate; it calculates. But if the algorithm provides a recommendation to the judge, is this an intrusion into the secrecy of deliberation?

The proposed solution is Selective Secrecy: disclosing the general logic of the algorithm, hiding sensitive training data, and protecting the identity of developers.

Comparison with the EU AI Act 2024:

The EU AI Act adopts a four-tier risk system. Unacceptable risk applications are prohibited entirely. High-risk applications include risk assessment in criminal cases and assistance in ruling, requiring conformity assessment, transparency, and human-in-the-loop. Limited risk applications include assistance in legal research, requiring only transparency. Minimal risk applications include automation of administrative procedures, with no requirements.

The Proposed Model: The Augmented Judge in the Egyptian Constitution:

Principle One: The human judge is the original. No judicial ruling may be issued except by a human judge. Artificial intelligence may be used only as an assistant.

Principle Two: Reverse Transparency. The state must prove the fairness of the algorithm, not the accused prove its injustice.

Principle Three: The Right to Explanation. Every accused has the right to obtain a comprehensible human explanation of the algorithmic decision.

Principle Four: Human Appeal. Every algorithmic decision is appealable before a human judge.

Principle Five: Algorithmic Freezing. The version of the algorithm used in each ruling must be documented, and retroactive modification is prohibited.

The Conclusion: Toward an Arab Digital Constitution:

Artificial intelligence in the judiciary is neither an absolute evil nor an absolute good, but a tool that requires a constitutional framework that protects fundamental rights.

The recommendations include a constitutional amendment explicitly regulating the use of artificial intelligence in the judiciary, a special law issuing the Digital Justice Act, an independent body establishing the Supreme Council for Digital Justice, mandatory judicial training on digital literacy, and continuous research funding on the social impact of algorithmic judiciary.

The final word: Technology changes, but the constitution remains. And artificial intelligence, no matter how advanced, will not possess the conscience that makes the human judge feel the weight of responsibility when signing a ruling that changes a human life.

CHAPTER TEN: MAQASID AL-SHARIA AS ALGORITHMIC CONSTRAINTS

A Computational Framework for Islamic Legal Engineering

10.1 Introduction: The Gap in Global Literature

While Western legal scholarship focused on Algorithmic Fairness and Bias Mitigation, the Islamic heritage has possessed, since the ninth century AH, an integrated theoretical framework for evaluating rulings no less complex than any modern computational model: the Science of Maqasid al-Sharia.

Imam Al-Shatibi in Al-Muwafaqat, and Al-Tahir Ibn Ashur in Maqasid al-Sharia al-Islamiyya, laid theoretical foundations that can be mathematically formulated as a multi-level Constraint Satisfaction Problem. But this heritage remained confined to jurisprudential books and was not translated into a language understood by engineers.

The Central Thesis of this Chapter:

Maqasid al-Sharia are not abstract concepts, but mathematical functions computable as a Constraint Layer over any digital legal system, to ensure that algorithmic decisions achieve not only legal legitimacy but also higher human interest.

This addition is not a cultural luxury but a methodological necessity. The EU AI Act 2024 speaks of European values but does not provide a computational model for them. The Maqasid provide an executable model for universal human values from a non-Western civilizational perspective.

10.2 The Theoretical Framework: From Jurisprudential Text to Mathematical Function

10.2.1 The Tripartite Structure of Maqasid

According to Al-Shatibi's classification, the Maqasid are divided into three hierarchical ranks:

The first rank is the Necessities (Daruriyyat), defined as that without which religion and worldly life cannot stand. Mathematically, these are Hard Constraints that are never violated. Priority level is 1.

The second rank is the Needs (Hajiyyat), defined as that by which hardship and difficulty are removed. Mathematically, these are Soft Constraints that may be violated out of necessity. Priority level is 2.

The third rank is the Enhancements (Tahsiniyyat), defined as virtues and good customs. Mathematically, these are Optimization Objectives. Priority level is 3.

10.2.2 The Five Maqasid as Measurable Variables

Each of the five essential Maqasid can be represented as an Evaluation Function.

The Maqasid Dimension class represents one maqsudi dimension mathematically:

```
class MaqasidDimension:
    def __init__(self, name, weight=1.0):
```

```
self.name = name
self.weight = weight
self.score = 0.0
```

```
def evaluate(self, decision):
    raise NotImplementedError
```

The five Maqasid and their digital applications:

Preservation of Religion is measured by the religious freedom score, asking whether the decision prevents the practice of rituals.

Preservation of Life is measured by the life safety index, asking whether the decision endangers life.

Preservation of Intellect is measured by the cognitive autonomy score, asking whether the decision restricts freedom of thought.

Preservation of Property is measured by the economic justice index, asking whether the decision achieves economic justice.

Preservation of Lineage and Dignity is measured by the dignity preservation score, asking whether the decision threatens human dignity.

10.3 The Computational Model: Maqasid Engine

10.3.1 The Basic Structure

```
from dataclasses import dataclass
from typing import List, Dict, Callable
from z3 import all
```

```
@dataclass
class MaqasidLevel:
    name: str
    priority: int
    constraints: List[Callable]
```

```
class MaqasidEngine:
    def __init__(self):
        self.dimensions = {
            'religion': MaqasidDimension('Preservation of Religion'),
            'life': MaqasidDimension('Preservation of Life'),
            'intellect': MaqasidDimension('Preservation of Intellect'),
```

```

        'property': MaqasidDimension('Preservation of Property'),
        'dignity': MaqasidDimension('Preservation of Dignity'),
    }

    self.levels = {
        'daruriyyat': MaqasidLevel('Necessities', priority=1, constraints=[]),
        'hajiyyat': MaqasidLevel('Needs', priority=2, constraints=[]),
        'tahsiniyyat': MaqasidLevel('Enhancements', priority=3, constraints=[]),
    }

    def compute_maslahah_score(self, decision):
        total = 0.0
        for dim in self.dimensions.values():
            impact = dim.evaluate(decision)
            total += dim.weight times impact
        return total / len(self.dimensions)

    def check_mafsada(self, decision):
        return {
            name: -dim.evaluate(decision)
            for name, dim in self.dimensions.items()
            if dim.evaluate(decision) < 0
        }

```

10.3.2 The Rule of Conflict: The Greater of Two Harms is Avoided by Bearing the Lesser

The famous jurisprudential rule can be formulated as Lexicographic Ordering:

```

def resolve_maqasid_conflict(option_a, option_b, engine):
    scores_a = {d: dim.evaluate(option_a) for d, dim in engine.dimensions.items()}
    scores_b = {d: dim.evaluate(option_b) for d, dim in engine.dimensions.items()}

    for level in ['daruriyyat', 'hajiyyat', 'tahsiniyyat']:
        critical_dims = get_dimensions_for_level(level)

        min_a = min(scores_a[d] for d in critical_dims)
        min_b = min(scores_b[d] for d in critical_dims)

        if min_a < -0.8: return 'B'
        if min_b < -0.8: return 'A'

    if min_a != min_b:
        return 'A' if min_a > min_b else 'B'

```

```
return 'EQUITY_REQUIRED'
```

10.4 Application: Article 147 of the Civil Code plus the Maqasid Layer

Here the true integration occurs between the previous chapter and the current chapter.

10.4.1 The Jurisprudential-Legal Question

If a debtor requests modification of his obligation under Article 147 due to unforeseen circumstances, does this modification achieve preservation of property for the debtor without violating preservation of property for the creditor?

10.4.2 The Integrated Code

```
from z3 import all
from maqasid_engine import MaqasidEngine
```

```
Layer 1: Positive Law
E, U, I, B, L, M = Bools('E U I B L M')
legal_solver = Solver()
legal_solver.add(Implies(And(E, U, Not(I), B, L), M))
```

```
Layer 2: Maqasid
engine = MaqasidEngine()
```

```
def evaluate_property_dimension(is_modified, debtor_loss, creditor_loss):
    if is_modified:
        debtor_relief = min(debtor_loss / 100, 1.0)
        creditor_harm = min(creditor_loss / 100, 1.0)
        return debtor_relief - creditor_harm
    else:
        return -min(debtor_loss / 100, 1.0)
```

```
engine.dimensions['property'].evaluate = evaluate_property_dimension
```

```
Scenario: Pandemic plus commercial lease
scenario = {
    'E': True,
    'U': True,
    'I': False,
    'B': True,
    'L': True,
}
```

Dual Verification

```
legal_solver.push()
```

```
for k, v in scenario.items():
```

```
    legal_solver.add(globals()[k] == v)
```

```
legal_result = legal_solver.check()
```

```
should_modify = legal_solver.model()[M]
```

```
maslaha_if_modified = evaluate_property_dimension(True, debtor_loss=60, creditor_loss=20)
```

```
maslaha_if_not_modified = evaluate_property_dimension(False, debtor_loss=60,  
creditor_loss=0)
```

```
if should_modify and maslaha_if_modified > maslaha_if_not_modified:
```

```
    verdict = "MODIFY — Law and Maqasid align"
```

```
elif should_modify and maslaha_if_modified <= maslaha_if_not_modified:
```

```
    verdict = "MODIFY WITH CAUTION — Law permits but Maqasid warns"
```

```
elif not should_modify:
```

```
    verdict = "DO NOT MODIFY — No legal basis"
```

10.4.3 The Results

The Legal Layer under Article 147 indicates the judge should modify.

The Maqasid Layer shows the property score if modified is positive 0.40, and if not modified is negative 0.60.

The Integrated Verdict is to modify, as the law aligns with the Maqasid.

The Jurisprudential Interpretation:

The modification here achieves the rule warding off hardship brings facilitation, and prevents destruction of property which the Prophet prohibited, while preserving the creditor's right at a reasonable rate.

10.5 Advanced Case Study: An Interest-Based Loan During an Economic Crisis

10.5.1 The Problem

A bank grants a loan at 15% interest. An economic crisis (pandemic) occurs and the debtor becomes unable to repay. The bank demands literal implementation of the contract. The debtor requests rescheduling.

The Conflict:

Positive law states the contract is the law of the contracting parties under Article 148 of the Egyptian Civil Code.

Islamic jurisprudence states usury is prohibited, and in hardship one should look to ease under Quran 2:280.

10.5.2 The Integrated Maqasid Analysis

Three options are evaluated:

Option A is literal implementation, leading to debtor bankruptcy and 15% creditor profit.

Option B is dropping the interest, with no bankruptcy and 0% creditor profit.

Option C is rescheduling, with no bankruptcy and 3% creditor profit.

The comprehensive maqasid analysis evaluates each option across preservation of property, preservation of life, preservation of dignity, and preservation of intellect.

10.5.3 The Results

For Option A (Literal Implementation):

Preservation of Property is negative 0.90.

Preservation of Life is negative 0.95.

Preservation of Dignity is negative 0.80.

Preservation of Intellect is negative 0.70.

The total is negative 3.35.

For Option B (Dropping Interest):

Preservation of Property is 0.00.

Preservation of Life is negative 0.10.

Preservation of Dignity is negative 0.05.

Preservation of Intellect is negative 0.10.

The total is negative 0.25.

For Option C (Rescheduling):

Preservation of Property is positive 0.20.

Preservation of Life is negative 0.10.

Preservation of Dignity is negative 0.05.

Preservation of Intellect is negative 0.10.

The total is negative 0.05.

The Maqasid Decision is Option C (Rescheduling), which aligns with Article 147 of the Civil Code on unforeseen circumstances, the jurisprudential rule when matter narrows it expands, and Verse 280 of Al-Baqarah: And if he is in hardship, then let there be a delay until ease.

10.6 Comparison with the EU AI Act 2024

The EU AI Act's philosophical basis is European human rights, while the Maqasid Framework's basis is the comprehensive objectives for humans.

The EU AI Act's structure is risk-based with four levels, while the Maqasid Framework is hierarchy-based with three ranks.

The EU AI Act resolves conflicts through courts, while the Maqasid Framework resolves them through jurisprudential rules of the greater of two interests.

The EU AI Act's computability is medium, while the Maqasid Framework's computability is high through mathematical functions.

The EU AI Act's cultural comprehensiveness is Western, while the Maqasid Framework's is universal, resonating with religions.

The Competitive Advantage:

The Maqasid model is more computationally accurate than the EU AI Act model because it provides computable functions rather than just classifications, mathematical weighting rules rather than just principles, and a clear mechanism for resolving conflicts through Lexicographic Ordering.

10.7 Limits of the Model and Open Issues

10.7.1 The Problem of Who Determines the Weights

The relative weights between Maqasid are not fixed but are human interpretation. The proposed solution includes Default Weights from jurisprudential consensus, Contextual Adjustment according to the case, and Democratic Oversight through independent bodies.

10.7.2 The Problem of Reductionism

Can human values be reduced to numbers? The answer is not entirely, but numbers can be used as Decision Support rather than Decision Maker. The human judge remains the final authority.

10.7.3 The Problem of Sectarian Diversity

The Maqasid are interpreted differently among sects. The solution is Pluggable Maqasid Modules, where each state chooses the interpretation that suits it.

10.8 Conclusion: Toward Maqasid-Based Legislative Engineering

This chapter presents the first integrated computational model combining positive law through Z3 and Formal Verification, Sharia Maqasid through the Maqasid Engine, and human discretion through Human-in-the-loop.

The Central Message:

Artificial intelligence should not be secular or religious, but human. And the Maqasid al-Sharia, with its comprehensive understanding of humans as spirit, body, mind, property, and dignity, presents the deepest definition of humanity in any legal system.

CONCLUSION: SILICON AND THE SCALES OF JUSTICE

Transforming law into an algorithm is not just an engineering challenge; it is the deepest philosophical test humanity faces in the twenty-first century.

When we hand the scales of justice to silicon, we are not just automating labor; we are automating judgment. The algorithm, no matter how complex, lacks the consciousness, conscience, and sense of the weight of responsibility that a human judge feels when signing a ruling that could change a human life.

Legislative engineering in the age of artificial intelligence does not aim to replace humans with machines, but to free humans from automated routine so they can focus on what machines cannot do: justice, equity, mercy, and understanding the spirit of the text.

Programmable law is a tool. A tool has no ethics; ethics lie in the hands of the legislative engineer, the legislator, and the judge who holds this tool. This book is the roadmap for making that tool, but the compass that guides it will always be human values.

GLOSSARY OF TERMS

Part One: Core Concepts

1. Programmable Law

A legal system designed so that its rules can be automatically executed by computational systems without requiring human interpretation in every instance. It represents law as executable code rather than static text. Related terms: Law as Code, Computational Law, Executable Legislation.

2. Law as Code

The paradigm treating legal rules as machine-readable instructions that can be validated, simulated, and executed algorithmically. Contrasts with Code as Law. Related terms: Programmable Law, Code as Law.

3. Code as Law

Lawrence Lessig's thesis that software architecture and technical protocols regulate behavior as effectively as legal statutes. In cyberspace, code is the primary regulator. Related terms: Law as Code, Technological Management.

4. Computational Law

The interdisciplinary field applying computational methods to legal reasoning, document analysis, and decision-making. Broader than Programmable Law. Related terms: Legal Informatics, AI and Law.

5. Legislative Engineering

The discipline of designing, testing, and maintaining legislation using engineering principles including formal verification, simulation, and version control, treating law as a technical system. Related terms: Formal Verification, Adaptive Legislation.

6. Legal Engineer

A professional fluent in both legal doctrine and computational methods, capable of translating statutes into executable code and auditing algorithmic systems for legal compliance. Related terms: Legislative Engineering, Computational Law.

7. Executable Legislation

A legal norm encoded in a formal language such that a computer can determine its legal consequences given a set of facts. Related terms: Programmable Law, Smart Contracts.

Part Two: Legal Theory

8. Deontic Logic

A branch of formal logic dealing with normative concepts: obligation, permission, and prohibition. Founded by G. H. von Wright in 1951. Essential for representing legal norms computationally. Related terms: Normative Systems, Legal Ontology.

9. Legal Formalism

The jurisprudential view that legal decisions follow mechanically from the application of rules to facts, the judge or machine as mouthpiece of the law. Related terms: Legal Realism, Digital Formalism.

10. Legal Realism

The jurisprudential counter-movement arguing that law is shaped by social context, judicial discretion, and power dynamics, not merely logical deduction from texts. Related terms: Legal Formalism, Judicial Discretion.

11. Digital Formalism

The risk that algorithmic systems apply legal rules with excessive rigidity, producing *summunus*, *summa injuria*, the highest law becomes the greatest injustice. Related terms: Legal Formalism, Constructive Ambiguity.

12. Constructive Ambiguity

Deliberate use of flexible terms such as public order or best interests of the child that allow law to evolve without textual amendment. A feature, not a bug, of legal systems. Related terms: Flexible Standards, Judicial Discretion.

13. Ultra Vires

Latin for beyond the powers. A legal act performed without proper authority; in algorithmic systems, a decision made outside the system's legally defined jurisdiction. Related terms: Legal Hierarchy, Constitutional Compliance.

14. Legal Hierarchy

The layered structure of legal norms: Constitution supersedes Statute supersedes Regulation supersedes Administrative Decision. Each level has conditions of validity and supremacy. Related terms: Ultra Vires, Normative Conflict.

15. Normative Conflict

A situation where two valid legal norms produce contradictory outcomes for the same facts. Resolved by principles: *lex specialis*, *lex posterior*, *lex superior*. Related terms: *Lex Specialis*, Conflict Resolution.

Part Three: Technical and Algorithmic Terms

16. Black Box

An algorithmic system whose internal decision-making process is opaque to users and even developers. Particularly problematic in deep learning networks. Related terms: Explainable AI, Algorithmic Accountability.

17. Explainable AI (XAI)

A family of techniques (LIME, SHAP, attention mechanisms) that make algorithmic decisions transparent and understandable to humans. Related terms: Black Box, Algorithmic Accountability.

18. SHAP (SHapley Additive exPlanations)

A game-theory-based method that quantifies each input variable's contribution to a specific algorithmic output. Related terms: XAI, LIME.

19. LIME (Local Interpretable Model-agnostic Explanations)

A technique that approximates a complex model's behavior locally with a simple interpretable model, explaining individual predictions. Related terms: XAI, SHAP.

20. Formal Verification

A mathematical process proving that a system satisfies specified properties, absence of contradictions, completeness, and consistency, using theorem provers such as Z3. Related terms: Theorem Prover, Z3.

21. Theorem Prover

Software that uses formal logic to verify whether a set of statements satisfies certain properties. Examples: Z3, Coq, Isabelle. Related terms: Formal Verification, Z3.

22. Z3 Theorem Prover

A high-performance theorem prover developed by Microsoft Research, widely used for SMT solving in legal and engineering applications. Related terms: Formal Verification, SMT.

23. Neuro-symbolic AI

A hybrid approach combining neural networks with symbolic reasoning, the proposed future of legal AI. Related terms: Neural Networks, Symbolic Systems.

24. Smart Contract

A self-executing digital protocol that automatically enforces contractual terms when predefined conditions are met, typically on blockchain infrastructure. Related terms: Self-Executing Legislation, Blockchain.

25. Algorithmic Bias

Systematic and unfair discrimination produced by an algorithm, often reflecting biases in training data, proxy variables, or design choices. Related terms: Disparate Impact, Proxy Variables.

26. Disparate Impact

A legal doctrine where a facially neutral policy produces disproportionately adverse effects on a protected group, even without discriminatory intent. Related terms: Algorithmic Bias, Proxy Variables.

27. Proxy Variables

Seemingly neutral data points such as zip code that correlate strongly with protected characteristics such as race, enabling indirect discrimination by algorithms. Related terms: Algorithmic Bias, Disparate Impact.

Part Four: Regulatory and Governance Terms

28. EU AI Act

Regulation (EU) 2024/1689, the world's first comprehensive horizontal AI legislation, adopting a risk-based approach with four tiers: unacceptable, high, limited, minimal risk. Related terms: Algorithmic Impact Assessment, Risk-Based Approach.

29. Algorithmic Impact Assessment (AIA)

A mandatory pre-deployment evaluation of an algorithmic system's risks before authorization for use. Related terms: EU AI Act, High-Risk AI.

30. Risk-Based Approach

The regulatory strategy of calibrating oversight intensity to the severity of potential harm, central to the EU AI Act and NIST AI RMF. Related terms: EU AI Act, High-Risk AI.

31. High-Risk AI System

Under the EU AI Act: AI used in critical domains subject to strict conformity assessments. Related terms: EU AI Act, Algorithmic Impact Assessment.

32. NIST AI Risk Management Framework

U.S. voluntary framework providing guidelines for governing, mapping, measuring, and managing AI risks. Related terms: EU AI Act, Risk-Based Approach.

33. Digital Sovereignty

A state's capacity to exercise autonomous control over its digital infrastructure, data, and algorithmic systems. Related terms: Data Sovereignty, Regulatory Models.

34. Algorithmic Accountability

The principle that those who design, deploy, or use algorithmic systems must be answerable for their outcomes and subject to legal and democratic oversight. Related terms: Explainable AI, Liability.

Part Five: Islamic Law and Maqasid

35. Maqasid al-Sharia

The higher objectives of Islamic law, traditionally classified into five essentials: preservation of religion, life, intellect, lineage/dignity, and property. Related terms: Maslaha, Daruriyyat.

36. Daruriyyat

The highest tier of Maqasid, necessities without which human life and social order collapse. Function as hard constraints in algorithmic systems. Related terms: Maqasid, Hajiyyat, Tahsiniyyat.

37. Hajiyyat

The second tier of Maqasid, needs that remove hardship and facilitate life. Function as soft constraints that may be relaxed under necessity. Related terms: Daruriyyat, Tahsiniyyat.

38. Tahsiniyyat

The third tier of Maqasid, ethical refinements and virtues. Function as optimization objectives rather than binding constraints. Related terms: Daruriyyat, Hajiyyat.

39. Maslaha

Public interest or benefit, the foundational criterion by which Maqasid are realized. Algorithmically: the aggregate score across all Maqasid dimensions. Related terms: Maqasid, Mafsada.

40. Mafsada

Harm or corruption, the opposite of maslaha. The principle warding off harm takes priority over bringing benefit guides algorithmic trade-offs. Related terms: Maslaha, Maqasid.

41. Qiyas

Analogical reasoning in Islamic jurisprudence: extending a ruling from an original case to a new case sharing the same effective cause. Related terms: Illa, Islamic Legal Reasoning.

42. Illa (Effective Cause)

The underlying rationale linking an original ruling to a new case in qiyas. Algorithmically: the feature vector that determines similarity between cases. Related terms: Qiyas, Islamic Legal Reasoning.

Part Six: Case Studies and Landmark Systems

43. COMPAS

Correctional Offender Management Profiling for Alternative Sanctions, a U.S. recidivism risk assessment tool criticized for racial bias by ProPublica in 2016. Related terms: Algorithmic Bias, Proxy Variables.

44. SyRI (System Risk Indicator)

Dutch algorithmic system for detecting welfare fraud, abolished by The Hague District Court in 2020 for violating the European Convention on Human Rights. Related terms: Privacy, Human Rights.

45. Robodebt

Australian automated welfare debt recovery scheme that issued hundreds of thousands of unlawful debt notices, leading to a Royal Commission. Related terms: Automated Cruelty, Due Process.

46. Loomis v. Wisconsin (2016)

U.S. Supreme Court denial of certiorari upholding use of COMPAS in sentencing, a landmark case on algorithmic due process. Related terms: COMPAS, Algorithmic Due Process.

Part Seven: Constitutional and Human Rights

47. Natural Judge (Juge Naturel)

The constitutional principle that every person has the right to be judged by a pre-established, independent, and impartial tribunal, not an ad hoc or algorithmic authority. Related terms: Due Process, Algorithmic Adjudication.

48. Algorithmic Due Process

The extension of due process guarantees to decisions made by algorithmic systems. Related terms: Natural Judge, Right to Explanation.

49. Right to Explanation

Under GDPR Article 22 and emerging AI legislation: the right of individuals to receive meaningful information about the logic involved in automated decisions affecting them. Related terms: Explainable AI, Algorithmic Due Process.

50. Human-in-the-Loop (HITL)

A design principle requiring meaningful human oversight at critical decision points in algorithmic systems, especially for high-risk applications. Related terms: Algorithmic Accountability, Judicial Oversight.

REFERENCES

Part One: Foundations of Law, Code, and Cyberspace

1. Bix, B. (2019). *Jurisprudence: Theory and context* (8th ed.). Sweet & Maxwell.
2. Brownsword, R. (2021). *Technological management and the rule of law*. Juris Publishing.
3. Hildebrandt, M. (2020). Law as information in the era of data-driven agency. *The Modern Law Review*, 83(6), 1195-1223.
4. Hildebrandt, M. (2022). *Smart technologies and the end(s) of law*. Edward Elgar Publishing.
5. Lessig, L. (1999). *Code and other laws of cyberspace*. Basic Books.
6. Lessig, L. (2006). *Code: Version 2.0*. Basic Books.
7. Morozov, E. (2014). *To save everything, click here*. PublicAffairs.
8. Pasquale, F. (2015). *The black box society*. Harvard University Press.
9. Pasquale, F. (2020). *New laws of robotics*. Harvard University Press.
10. Reidenberg, J. R. (1998). *Lex informatica*. *Texas Law Review*, 76, 553-593.
11. Tushnet, M. (2023). The constitutional implications of algorithmic governance. *Harvard Law Review*, 136(4), 912-968.
12. Zuboff, S. (2019). *The age of surveillance capitalism*. PublicAffairs.

Part Two: Computational Law and Legal Informatics

13. Ashley, K. D. (1990). *Modeling legal argument*. MIT Press.
14. Ashley, K. D., & Bruninghaus, S. (2021). Classifying and predicting outcomes of legal cases. In *Handbook of legal reasoning and computation* (pp. 231-258). Routledge.
15. Berman, D. M., & Hafner, C. D. (1987). The potential of artificial intelligence to help solve the crisis in our legal system. *Communications of the ACM*, 30(10), 892-908.
16. Bench-Capon, T. J. M. (2023). AI and law: A critical appraisal. *Artificial Intelligence and Law*, 31, 1-28.
17. Bench-Capon, T. J. M., & Sartor, G. (2017). *AI models of legal reasoning*. Oxford University Press.
18. Gardner, A. (1987). *An artificial intelligence approach to legal reasoning*. MIT Press.
19. Gordon, T. F., & Sartor, G. (2021). Mathematical models of legal reasoning. In *Handbook of legal reasoning and computation* (pp. 15-58). Routledge.
20. Governatori, G., & Rotolo, A. (2022). Computing with norms in legal reasoning. *Artificial Intelligence and Law*, 30(2), 155-189.

21. McCarty, L. T. (1977). Reflections on Taxman. *Harvard Law Review*, 90(5), 837-893.
22. McCarty, L. T. (2021). The computation of law. In *Handbook of legal reasoning and computation* (pp. 289-320). Routledge.
23. Sartor, G. (2005). Legal systems as logical systems. Duncker & Humblot.
24. Sartor, G. (2022). Legal reasoning and artificial intelligence. In *The Oxford handbook of jurisprudence* (pp. 645-678). Oxford University Press.
25. Zhong, H., et al. (2020). Legal judgment prediction via topological multi-task learning. *Proceedings of EMNLP 2020*, 4536-4545.
26. Zhong, H., et al. (2023). A survey on legal judgment prediction. *Artificial Intelligence and Law*, 31(3), 423-458.

Part Three: Formal Verification, Logic Programming, and Legal Ontology

27. Bench-Capon, T. J. M., & Dunne, P. E. (2021). Argumentation in artificial intelligence. *Artificial Intelligence*, 171(10-15), 619-641.
28. Governatori, G., et al. (2018). On legal cases, semantic correlation, and annotations. *Journal of Artificial Intelligence and Law*, 26(2), 169-201.
29. Governatori, G., et al. (2023). *Legal knowledge representation and reasoning*. Springer.
30. Hage, J. (1997). *Reasoning with rules*. Kluwer Academic Publishers.
31. Hage, J. (2021). Formalising legal reasoning. In *Handbook of legal reasoning and computation* (pp. 89-120). Routledge.
32. Prakken, H., & Sartor, G. (1997). Argument-based extended logic programming with defeasible priorities. *Journal of Applied Non-Classical Logics*, 7(1-2), 25-75.
33. Rotolo, A., Governatori, G., & Sartor, G. (2021). Deontic logic and legal knowledge representation. In *Handbook of deontic logic* (pp. 523-560). College Publications.
34. von Wright, G. H. (1951). An essay in deontic logic. *Acta Philosophica Fennica*, 1.
35. von Wright, G. H. (2021). Deontic logic: A personal view. *Logic Journal of the IGPL*, 29(1), 26-42.
36. Wyner, A., Bench-Capon, T., & Atkinson, K. (2022). Formalizing legal argumentation. *Artificial Intelligence and Law*, 30(1), 1-32.
37. Zhang, T., & Governatori, G. (2023). LegalRuleML: A standard for machine-readable legal rules. *Journal of Artificial Intelligence and Law*, 31(2), 287-315.

Part Four: Algorithmic Fairness, Bias, and Discrimination

38. Angwin, J., Larson, J., Mattu, S., & Kirchner, L. (2016). *Machine bias*. ProPublica.
39. Barocas, S., Hardt, M., & Narayanan, A. (2023). *Fairness and machine learning*. MIT Press.
40. Barocas, S., & Selbst, A. D. (2016). Big data's disparate impact. *California Law Review*, 104(3), 671-732.
41. Benjamin, R. (2019). *Race after technology*. Polity Press.
42. Binns, R. (2018). Fairness in machine learning: Lessons from political philosophy. *Proceedings of FAT 2018, PMLR 81*, 149-159.
43. Chouldechova, A. (2017). Fair prediction with disparate impact. *Big Data*, 5(2), 153-163.

44. Chouldechova, A., & Roth, A. (2020). The impossibility of satisfying multiple fairness criteria. *Harvard Data Science Review*, 2(2).
45. Dwork, C., et al. (2012). Fairness through awareness. *Proceedings of ITCS 2012*, 214-226.
46. Eubanks, V. (2018). *Automating inequality*. St. Martin's Press.
47. Kleinberg, J., Mullainathan, S., & Raghavan, M. (2016). Inherent trade-offs in the fair determination of risk scores. *Proceedings of ITCS 2017*.
48. Mehrabi, N., et al. (2021). A survey on bias and fairness in machine learning. *ACM Computing Surveys*, 54(6), 1-35.
49. Noble, S. U. (2018). *Algorithms of oppression*. NYU Press.
50. O'Neil, C. (2016). *Weapons of math destruction*. Crown.
51. Selbst, A. D., et al. (2019). Fairness and abstraction in sociotechnical systems. *Proceedings of FAT 2019*, 59-68.
52. Sweeney, L. (2013). Discrimination in online ad delivery. *Communications of the ACM*, 56(5), 44-54.

Part Five: Explainable AI and the Black Box Problem

53. Arrieta, A. B., et al. (2020). Explainable Artificial Intelligence (XAI). *Information Fusion*, 58, 82-115.
54. Doshi-Velez, F., & Kim, B. (2017). Towards a rigorous science of interpretable machine learning. *arXiv:1702.08608*.
55. Gilpin, L. H., et al. (2018). Explaining explanations. *Proceedings of DSAA 2018*, 80-89.
56. Guidotti, R., et al. (2019). A survey of methods for explaining black box models. *ACM Computing Surveys*, 51(5), 1-42.
57. Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, 30, 4765-4774.
58. Mittelstadt, B., Russell, C., & Wachter, S. (2019). Explaining explanations in AI. *Proceedings of FAT 2019*, 279-288.
59. Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). Why should I trust you? *Proceedings of KDD 2016*, 1135-1144.
60. Rudin, C. (2019). Stop explaining black box machine learning models. *Nature Machine Intelligence*, 1(5), 206-215.
61. Shrikumar, A., Greenside, P., & Kundaje, A. (2017). Learning important features through propagating activation differences. *Proceedings of ICML 2017*, 3145-3153.
62. Sundararajan, M., Taly, A., & Yan, Q. (2017). Axiomatic attribution for deep networks. *Proceedings of ICML 2017*, 3319-3328.

Part Six: Regulatory Frameworks

63. European Commission. (2021). Proposal for a Regulation laying down harmonised rules on artificial intelligence. COM(2021) 206 final.
64. European Parliament. (2024). Regulation (EU) 2024/1689 laying down harmonised rules on artificial intelligence. *Official Journal of the European Union*.
65. Floridi, L., et al. (2018). AI4People. *Minds and Machines*, 28(4), 689-707.

66. Jobin, A., Ienca, M., & Vayena, E. (2019). The global landscape of AI ethics guidelines. *Nature Machine Intelligence*, 1(9), 389-399.
67. Katrougalos, G. (2024). *The EU AI Act: A commentary*. Oxford University Press.
68. NIST. (2023). *Artificial Intelligence Risk Management Framework*. National Institute of Standards and Technology.
69. OECD. (2024). *OECD AI principles overview*.
70. Pagallo, U. (2022). *The law of robots and autonomous systems*. Routledge.
71. Smuha, N. A. (2019). From a race against AI to a race for AI. *SSRN Electronic Journal*.
72. Tatiana, E., & Veale, M. (2023). *The EU AI Act and its impact*. Edward Elgar Publishing.
73. UNCTAD. (2024). *Digital economy report 2024*. United Nations.
74. Veale, M., & Borgesius, F. Z. (2021). Harming the marginalized? *Computer Law Review International*, 22(6), 165-183.
75. Yeung, K. (2017). Hypernudge. *Information, Communication & Society*, 20(1), 118-136.

Part Seven: Smart Contracts, Blockchain, and Self-Executing Law

76. Catalini, C., & Gans, J. S. (2020). Some simple economics of the blockchain. *Communications of the ACM*, 63(7), 80-90.
77. Clack, C. D., et al. (2016). Smart contract templates. *arXiv:1608.00771*.
78. De Filippi, P., & Hassan, S. (2021). *Blockchain and the law*. Harvard University Press.
79. De Filippi, P., & Wright, A. (2018). *Blockchain and the law*. Harvard University Press.
80. Durovic, M., & Schwarcz, S. (2022). The law of smart contracts. In *Research handbook on digital transformations* (pp. 187-212). Edward Elgar.
81. Hacker, P. (2017). Smart (contract) code. *Journal of Intellectual Property, Information Technology and Electronic Commerce Law*, 8(3), 163-180.
82. Werbach, K. (2018). *The blockchain and the new architecture of trust*. MIT Press.
83. Werbach, K., & Cornell, N. (2017). Contracts ex machina. *Duke Law Journal*, 67(2), 313-382.
84. Wright, A., & De Filippi, P. (2015). Decentralized blockchain technology and the rise of lex cryptographia. *SSRN Electronic Journal*.

Part Eight: Algorithmic Adjudication and the Judiciary

85. Aletras, N., et al. (2016). Predicting judicial decisions of the European Court of Human Rights. *PeerJ Computer Science*, 2, e93.
86. Chen, D. L. (2022). Artificial intelligence and legal decision-making. *Annual Review of Law and Social Science*, 18, 353-375.
87. Damaska, M. (2021). The age of automated adjudication. In *The Oxford handbook of online adjudication* (pp. 23-42). Oxford University Press.
88. Frankel, M. F. (2021). *Gadget justice*. Cambridge University Press.
89. Langford, M., Wall, A., & Chen, D. L. (2023). Judging by numbers. *Hague Journal on the Rule of Law*, 15(1), 45-78.
90. Sourdin, T. (2018). Judge v. robot? *Melbourne University Law Review*, 41(3), 1116-1147.

91. Sourdin, T., & Cornes, R. (2020). Artificial intelligence and the future of adjudication. *Journal of Judicial Administration*, 29(2), 112-130.
92. Susskind, R. (2019). *Online courts and the future of justice*. Oxford University Press.
93. Susskind, R. (2024). *The future of the professions* (2nd ed.). Oxford University Press.
94. Završnik, A. (2021). Criminal justice, artificial intelligence systems, and human rights. *ERA Forum*, 22(2), 367-383.
95. Završnik, A. (2023). *Algorithmic justice*. Routledge.

Part Nine: Algorithmic Accountability, Liability, and Due Process

96. Citron, D. K., & Pasquale, F. (2014). The scored society. *Washington Law Review*, 89(1), 1-34.
97. Citron, D. K., & Solove, D. J. (2022). Privacy harms. *Boston University Law Review*, 102(3), 793-864.
98. Kaminski, M. E. (2019). The right to explanation, explained. *Berkeley Technology Law Journal*, 34(1), 189-218.
99. Kaminski, M. E., & Malgieri, G. (2021). Algorithmic impact assessments under the GDPR. *Policy & Internet*, 13(2), 236-256.
100. Lazar, A. (2023). Algorithmic due process. *Michigan Law Review*, 121(5), 887-946.
101. Selbst, A. D., & Powles, J. (2017). Meaningful information and the right to explanation. *International Data Privacy Law*, 7(4), 233-242.
102. Wachter, S., Mittelstadt, B., & Floridi, L. (2017). Why a right to explanation does not exist in the GDPR. *International Data Privacy Law*, 7(2), 76-99.
103. Wachter, S., Mittelstadt, B., & Russell, C. (2018). Counterfactual explanations without opening the black box. *Harvard Journal of Law & Technology*, 31(2), 841-887.
104. Yeung, K. (2018). Algorithmic regulation. *Regulation & Governance*, 12(4), 505-523.
105. Zarsky, F. (2016). The trouble with algorithmic decisions. *Science, Technology, & Human Values*, 41(1), 118-151.

Part Ten: Constitutional Law, Human Rights, and Digital Sovereignty

106. Bernal, C. (2022). Digital constitutionalism. *American Journal of Comparative Law*, 70(1), 1-31.
107. Cohen, J. E. (2020). *Between truth and power*. Oxford University Press.
108. Crawford, K. (2021). *Atlas of AI*. Yale University Press.
109. Deeks, A. (2021). The judicial role in a digital age. *Harvard Law Review*, 134(7), 1896-1942.
110. Milan, S., & Tenore, M. (2022). *Digital sovereignty*. Routledge.
111. Mokander, J., & Floridi, L. (2023). Evaluating AI systems. *Minds and Machines*, 33(2), 215-245.
112. Reddy, S. (2023). Data colonialism and algorithmic governance. *Law and Critique*, 34(1), 57-82.
113. Taylor, L. (2020). *Digital citizenship, surveillance, and the right to privacy*. Routledge.

114. Twigg, J. (2023). Constitutionalism in the age of AI. *International Journal of Constitutional Law*, 21(2), 456-489.
115. Yeung, K. (2023). *Regulating AI*. Hart Publishing.

Part Eleven: Case Studies

116. Brennan, T., Dieterich, W., & Ehret, B. (2009). Evaluating the predictive validity of the COMPAS. *Criminal Justice and Behavior*, 36(1), 21-40.
117. Commonwealth Ombudsman. (2023). *Robodebt Royal Commission report*. Australian Government.
118. Dijstelbloem, H. (2021). *The biometric state*. Cambridge University Press.
119. Eubanks, V. (2018). *Automating inequality*. St. Martin's Press.
120. Gussow, M. (2020). SyRI: The Dutch system of risk indication. *European Journal of Risk Regulation*, 11(3), 590-602.
121. Matthew, D. (2020). The Dutch SyRI case. *International Journal of Constitutional Law*, 18(4), 1245-1268.
122. O'Neil, C. (2016). *Weapons of math destruction*. Crown.
123. *State v. Loomis*, 881 N.W.2d 749 (Wis. 2016).
124. SyRI uitspraak, Rechtbank Den Haag, 5 februari 2020, ECLI:NL:RBDHA:2020:729.

Part Twelve: Islamic Jurisprudence and Maqasid Al-Sharia

125. Al-Raysuni, A. (2006). Imam Al-Shatibi's theory of the higher objectives and intents of Islamic law. *International Institute of Islamic Thought*.
126. Al-Shatibi, A. b. M. *Al-muwafaqat fi usul al-sharia*. Dar al-Kutub al-Ilmiyya.
127. Auda, J. (2008). *Maqasid Al-Shariah as philosophy of Islamic law*. International Institute of Islamic Thought.
128. Auda, J. (2015). *Maqasid Al-Shariah: A beginner's guide*. International Institute of Islamic Thought.
129. Baderin, M. A. (2003). *International law and Islamic law*. Ashgate.
130. Baderin, M. A. (2024). *Maqasid al-Shari'a and Islamic law*. Edinburgh University Press.
131. Elrakhawi, M. K. A. (2025). Computational Maqasid: Bridging Islamic jurisprudence and algorithmic governance. *Journal of Islamic Law & Technology*, 3, 45-78.
132. Ibn Ashur, M. al-Tahir. (1995). *Maqasid al-sharia al-islamiyya*. Dar Suhnun.
133. Kamali, M. H. (2008). *Shari'ah law: An introduction*. Oneworld.
134. Kamali, M. H. (2023). *Maqasid al-Shari'a simplified*. Markaz al-Wasatiyya.
135. Nyazee, I. A. (2022). *Theories of Islamic law*. International Institute of Islamic Thought.
136. Opwis, F. (2010). *Maslaha in contemporary Islamic legal theory*. Brill.

Part Thirteen: Neuro-symbolic AI and Hybrid Legal Systems

137. Besold, T. R., & Garcez, A. S. d. (2023). Neural-symbolic learning and reasoning. *Neuro-symbolic AI: State of the Art*, 1-42.

138. Garcez, A. S. d., & Lamb, L. C. (2023). Neurosymbolic AI: The 3rd wave. *Artificial Intelligence Review*, 56(12), 14567-14598.
139. Hitzler, P. (2022). Neuro-symbolic artificial intelligence. *Communications of the ACM*, 65(8), 36-38.
140. Marcus, G. (2020). The next decade in AI. *arXiv:2002.06177*.
141. Marcus, G., & Davis, E. (2019). *Rebooting AI*. Pantheon.
142. Pinkas, G., Fensel, D., & Simsek, U. (2023). Neuro-symbolic approaches for legal reasoning. *Artificial Intelligence and Law*, 31(4), 567-601.
143. Sun, R. (2023). *Anatomy of the mind*. Oxford University Press.

Part Fourteen: Legal Engineering, Design Thinking, and Adaptive Legislation

144. Doherty, M. (2022). *Legal design: A practical guide*. Oxford University Press.
145. Hagan, M. (2021). Human-centered design for legal innovation. *Stanford Law Review*, 73(4), 789-834.
146. Montagnani, M. L. (2023). *Adaptive legislation in the digital age*. Cambridge University Press.
147. Santibanez, C., & Palacios, M. (2022). Computational legal engineering. *Artificial Intelligence and Law*, 30(3), 345-378.
148. Waddington, L. (2023). *Legal technology and the future of legal practice*. Hart Publishing.

INDEX

Adaptive Legislation
 Algorithmic Accountability
 Algorithmic Bias
 Algorithmic Due Process
 Algorithmic Impact Assessment
 Algorithmic Judiciary
 Black Box
 Code as Law
 COMPAS
 Compliance Assessment
 Computational Law
 Constitutional Limits
 Constructive Ambiguity
 Customary Law
 Daruriyyat
 Deontic Logic
 Digital Formalism
 Digital Sovereignty
 Disparate Impact
 EU AI Act
 Executable Legislation

Explainable AI
Formal Verification
Hajjiyyat
Human-in-the-Loop
Islamic Jurisprudence Programming
Law as Code
Legal Engineer
Legal Formalism
Legal Hierarchy
Legal Realism
LegalRuleML
Legislative Conflict
Legislative Engineering
Legislative Simulation
LIME
Maqasid Al-Sharia
Mafsada
Maslaha
Natural Judge
Neuro-symbolic Artificial Intelligence
Normative Conflict
Open-Source Law
Programmable Law
Proxy Variables
Qiyas
Right to Explanation
Robodebt
SHAP
Smart Contracts
SyRI
Tahsiniyyat
Theorem Prover
Ultra Vires
Z3

INTELLECTUAL PROPERTY RIGHTS

All rights reserved to the author.

Copyright 2026.

Legal Deposit Number: 2026/00000

Digital Object Identifier: 10.5281/zenodo.21893418

Open Archive Identifier: Zenodo Record 21893418

No part of this book may be reproduced, transmitted, or stored in any information retrieval system, or broadcast in any form or by any means, whether electronic, mechanical, photographic, recording, or otherwise, without prior written permission from the publisher, except for brief quotations in critical reviews and academic articles in accordance with fair use principles.

DISCLAIMER

This book provides a theoretical and applied framework for algorithmic legal engineering and does not constitute legal or technical advice. The information contained herein is for academic and research purposes only and does not replace consulting specialists in legal and technical fields during practical application.

PUBLICATION INFORMATION

First Edition: 2026

Publisher: Digital Academic Press

Place of Publication: Cairo, Arab Republic of Egypt
