Rules as Executed

Publishing Machine-Executable Law to Rebalance the Powers

Eelco Hotting* Anne Schuth*

Dutch Ministry of the Interior and Kingdom Relations

eelco.hotting@minbzk.nl anne.schuth@minbzk.nl

Position paper, 31 August 2026

Abstract

Government law execution is formalized in software, but that software is not published. This deepens the executive's structural advantage in the separation of powers: it alone can see how operative law is applied, while Parliament, courts, and citizens cannot verify, review, or predict its application. This paper proposes that law execution be published as machine-executable specifications in a restricted, declarative format designed to be readable by legal professionals. Such a format must be limited to a small set of operations designed to guarantee termination and to support version management, attestation, and formal analysis. The published specification is the one that decides cases: each decision records the version it ran and the inputs it ran on, so its recipient can re-execute the published rule and see whether the outcome is the one it yields. Determinism makes the outcome checkable. The signed trace fixes the inputs but not their truth; that requires binding an input to the source authoritative for it, where one exists. The specification computes a default: a departure replaces a computed value with one from another source, and a reader can establish that it was made, on what ground, and by whom. A reference implementation demonstrates feasibility. We argue that publication of executable law restores the precondition for constitutional oversight, each actor within its own office: the recipient of a decision and the court can re-execute that case, auditors with lawful access to the data can do so across a population, and Parliament and the public can read and analyze the rule itself, with no case data at all. The paper develops the constitutional argument, the required properties of the format, the implications for each constitutional actor, and the research agenda this approach requires.

doi:10.5281/zenodo.22201094 Position paper 31 Aug 2026

1 Introduction

The rule of law rests on a principle: government's operative rules must be knowable, comprehensible, and predictable for those subject to them (Voermans et al., 2011). These demands were formulated for law addressed to human readers, and they assumed that

*Equal contribution. The authors are now with the Dutch Digital Service, Ministry of Economic Affairs and Climate. The work was done at the Ministry of the Interior and Kingdom Relations.

a rule, once published and understood, is applied by officials whose reasoning can in turn be examined. Digital execution breaks that assumption: rules are now applied at scale by systems whose operation is not visible in the published text, so knowability and predictability on paper no longer guarantee that the rule as executed matches the rule as promulgated. This paper argues that digital execution therefore adds a fourth demand, implicit in the first three but no longer satisfied by them: the rules as executed must be independently verifiable. This is not a new value but the condition under which the older ones survive automation, and it is close to what Fuller called congruence between official action and declared rule (Fuller, 1969). Across the Dutch state, law execution unfolds in systems opaque even to the officials implementing them (Peeters and Widlak, 2023; Widlak and Peeters, 2025). Benefit eligibility decisions, license approvals, and penalty calculations are produced by databases, spreadsheets, and bespoke software whose logic remains effectively secret, neither published in the Staatsblad (Bulletin of Acts and Decrees) nor otherwise accessible to citizens, legislators, or courts. Often, the workings are hidden even from the executive body itself: the logic is buried in software, layered under years of incremental change across related legislation, visible to no one in its entirety. Sometimes that software is licensed from an external supplier whose source code the government neither owns nor may inspect, leaving the decision logic opaque even to the executing body itself. The law itself may be formally published, but its execution remains hidden. No branch of government fully sees how legislation functions. The legislative and judicial branches cannot see the effect of the laws they pass or the decisions they review. The executive branch operates the systems but often lacks a consolidated view of the rules those systems encode.

At its core, this issue concerns the constitutional balance of power. The Grondwet (Dutch Constitution) does not enact a strict separation of powers; it allocates legislative, executive, and judicial functions across institutions that check and constrain one another, a balance in which the executive has long been the most powerful actor. Digitalization has given the executive power a structural advantage: it can encode its interpretations of law into systems accessible only through technical expertise, creating a practical monopoly on access to how law is actually applied. The judiciary lacks the tools to verify compliance. Parliament cannot assess enacted legislation's effects. Citizens cannot predict or challenge decisions from invisible law execution. Scheltema (2021b) argued that all three branches must actively contribute to keeping the *rechtsstaat* (rule-of-law state) accessible to citizens, each through its own constitutional function.

We propose that formalizing law execution as a published, inspectable, machine-executable specification can restore constitutional balance. If the specification is published in a format restricted enough to ensure readability by legal professionals, and versions are managed with the same rigor as the law itself, government's operative rules become subject to the same scrutiny and debate as the law text itself. The proposal makes publication binding: the executing organization must decide cases with the specification it publishes, and each decision records an attestation of the specification, the inputs, and the trace that produced it. Binding does not automate the decision: the competent authority can depart from a computed outcome where the law gives it the power to, and the departure and its ground are recorded in the same attested trace (Section 4.6). The paper proceeds from the constitutional problem (Section 3) through the proposed format (Section 4), its implications for each constitutional actor (Section 6), and the research agenda required to realize this approach (Section 12).

One caveat is owed up front. Both authors are engineers, and this paper reasons about

constitutional doctrine anyway. We hope the legal scholars on whose terrain we trespass will forgive what they can and correct what they cannot; the questions of Section 12.1 are meant to make the correcting easy.

2 How We Got Here

This section traces how an engineering problem became a constitutional argument.

2.1 Proactive Service and Known Data

The intellectual origin of this work is modest. A fieldlab (a multi-day collaborative sprint bringing together teams from different government organizations) on proactive service delivery (Tekofsky and de Groot, 2024) began with a scenario: a young person turns eighteen. At that moment, they may become eligible for multiple benefits, subject to various regulations, across different governmental bodies. The fieldlab's goal was to build software that could, from data already known to government, automatically determine that eligibility and inform the citizen.

This sounds simple in the abstract. In practice, it required encoding the rules of multiple laws, tracing their dependencies, and detecting the moment they became applicable. The software would need to know the health care benefit law's definition of household composition, the income thresholds, the interaction with student finance rules, the municipal residency requirements. Each rule came from different legislation, updated at different times, administered by different bodies.

2.2 From Fieldlab to Format

The central technical challenge was one that legal professionals had articulated clearly: proactive decisions require knowing whether a decision is beneficial for a citizen. Yet the interdependencies between laws, processes, and data points make that determination intractable in practice. Government does not know all the factors affecting how law operates on an individual's situation. Where to draw the boundary of which laws and which data points to include in a benefit calculation is itself an unresolved question. The fieldlab needed an abstraction: an engine and an executable rule specification that could be changed quickly to simulate changes in law and temporal effects, and that could be used from different perspectives: from the citizen exploring scenarios and from the executive branch determining eligibility.

Under the pressure of the fieldlab's deadline, a simple format emerged that made the idea work as a proof of concept. The first version was a Python engine with executable rule specifications in YAML. A WebAssembly (WASM) packaging of this Python engine followed, enabling the same rules to run deterministically in the browser.

The fieldlab produced a proof of concept and confirmed feasibility. After the fieldlab, further applications followed. A second engine was built in Go to demonstrate that engines were simple enough to be independently implemented and could process the same rules and data to produce identical results. A later reimplementation in Rust became the reference engine; it compiles to WebAssembly, bringing the same in-browser determinism to the current format. The question shifted from "how do we encode this law" to "what does this law actually specify". The first question is technical; the second is legal. Could rules

be represented in a form restricted enough to be unambiguous, clear enough to be read by people trained in law, and formal enough to be executable?

2.3 Prior Approaches to Machine-Readable Law

There is a long history of attempts to formalize law as executable rules. That lineage begins with McCarty (1977), who formalized US corporate reorganization tax law in TAXMAN, and Sergot et al. (1986), who encoded the British Nationality Act as a logic program. Dutch researchers pioneered legal ontology work in the 1990s (Valente and Breuker, 1994; van Kralingen, 1997; Visser and Bench-Capon, 1998). From 1999, the Belastingdienst (Dutch Tax Administration) pursued ontology-based rule management through POWER (Programme for an Ontology-based Working Environment for modeling and use of Regulations and legislation) (van Engers et al., 2001). The Immigration and Naturalisation Service[1] (IND) has applied rule-based systems to the Vreemdelingenwet (Aliens Act) since the early 1990s (van Doesburg and van Engers, 2019), work that evolved into the Calculemus (Latin for 'let us calculate') method and Formal Language for the INTerpretation of sources of norms (FLINT) (van Doesburg, van der Storm, and van Engers, 2016; van Binsbergen et al., 2020). FLINT models the normative structure of law (who may act, what duties arise, which powers are conferred) rather than encoding the computational logic that produces individual decisions. The Belastingdienst developed RegelSpraak[2], a controlled natural language for encoding tax rules, and its authoring environment ALEF (Agile Law Execution Factory)[3]. Lokin (2018) proposed treating the legislator as a system manager, and Ausems, Bulles, and Lokin (2021) published a systematic method for analyzing legal sources for ICT implementation; this line of work has since matured into a sustained program on legislating for the digital rule of law (Lokin, 2026). Internationally, the idea has since moved into production. France encodes parts of its tax and benefit law as executable specifications: OpenFisca (open-source engine encoding French tax and benefit law)[4] has powered official public simulators since around 2014, and the tax administration has published the operative source of its income-tax computation; parts of French benefit law have since been encoded in the verified language Catala (Mérigoux, Chataing, and Protzenko, 2021). Commercial rule engines such as Oracle Policy Automation run declarative encodings of legislation in government production. New Zealand's Better Rules program drafted legislation as executable rules from the outset (Service Innovation Lab, 2018). The OECD surveyed this landscape under the heading "rules as code" and documented both its promise and its fragmentation (Mohun and Roberts, 2020).

Executability and publication each have precedent in this lineage. The present proposal conjoins them under a binding regime: the published specification must be the one that governs execution, and whoever holds a decision's inputs can check that identity by re-running the published rule. Software supply-chain security standardizes such a binding, provenance attestation that cryptographically ties a deployed artifact to the published source it was built from (Open Source Security Foundation, 2023). Law execution has none. France already imposes a statutory duty to disclose the rules of algorithmic decisions, yet that duty concerns documentation *about* the processing and does not guarantee that

[1] Dutch: *Immigratie- en Naturalisatiedienst*.
[2] https://regelspraak.nl
[3] https://regels.overheid.nl/docs/methods/ALEF
[4] https://openfisca.org

Table 1: Prior approaches to machine-readable law, grouped by the goal they optimize for. *Bound to exec.* means that the organization must decide with the specification it publishes, and that each decision attests which specifications it ran, so that a reader can establish that the published rule decided the case. The rightmost column states the property this paper's constitutional argument requires, and no prior approach was designed to provide it.

	authoring & execution		*reader-side*
Approach	exec.	publ.	bound to exec.
Akoma Ntoso / LegalRuleML	–	yes	–
RegelSpraak / ALEF (Netherlands)	yes	part	–
FLINT / Calculemus (Netherlands)	yes	part	–
Oracle Policy Automation	yes	–	–
OpenFisca (France)	yes	yes	–
Catala (France)	yes	yes	–
Better Rules (New Zealand)	yes	yes	–
STOP/TPOD, toepasbare regel (Netherlands)	yes	yes	–
This paper	yes	yes	yes

the disclosed rule is the rule the system runs; in practice compliance has remained thin. Mandatory publication bound to execution, rather than disclosure alongside it, is the gap this paper addresses.

Table 1 is not a scorecard on which this paper's proposal outscores the others; each prior system is strong at the goal it was built for, which is helping specialists author and run rules. The right-hand property (whether the published rule is guaranteed to be the rule that runs) is one that none of them set out to provide, because their purpose lies at the writing and execution end. Catala, for instance, can prove properties of its encodings and that compilation preserves them, but not that the system deciding a citizen's case ran the published version. The constitutional problem of Section 3 is at the reading end instead.

One precedent is closer to home than any of these. Under the Omgevingswet (Environment and Planning Act), Dutch authorities publish machine-readable, version-managed rules into a statutory register: planning and environmental instruments are published through the STOP (Standaard Officiële Publicaties, the Dutch standard for structured publication of official regulations) and TPOD (Toepassingsprofielen voor Omgevingsdocumenten, the STOP application profiles for planning and environmental instruments) standards, and the same legal rules are made available to citizens as *toepasbare regels* (applicable rule: a rule made operable as a decision tree in the permit portal), the decision trees that drive the permit check in the national portal of the Digital System for the Environment and Planning Act[5] (DSO). This is among the most advanced deployments of machine-readable law anywhere, and it took a decade of sustained effort. Drafting *toepasbare regels* is voluntary, and by early 2025 roughly nine in ten municipalities had taken up the work (Ministerie van Volkshuisvesting en Ruimtelijke Ordening, 2025). The proposal here extends the same trajectory, and it takes from the DSO exactly the lesson the DSO makes visible. Each competent authority authors its own *toepasbare regels* by hand (Informatiepunt Leefomgeving, 2026): the legal rule is published in one form, translated into a *toepasbare regel* in another, and applied to an individual case in the case-handling

[5]Dutch: *Digitaal Stelsel Omgevingswet*.

systems of the authority in a third; keeping these aligned is ongoing interpretive work, and nothing in the standards guarantees that the decision tree a citizen meets, or the decision a case-worker issues, matches the rule as published. This paper adds the binding relation that closes that gap: a single executable specification that is published because it is the one the decision runs on. The DSO also differs in reach: its *toepasbare regels* tell a citizen whether an activity needs a permit or a notification, not the operative logic that computes the outcome of an entitlement or a levy. The step this paper proposes begins where those questionnaires end.

2.4 A Broader Pattern

The problem this paper addresses is not new. As early as 2011, the Wetenschappelijke Raad voor het Regeringsbeleid (Scientific Council for Government Policy (WRR)) warned that an "iGovernment" had emerged in which information flows and system architectures, rather than conscious policy choices, shape how citizens experience the state (Wetenschappelijke Raad voor het Regeringsbeleid, 2011). The opacity of digitalized law execution has since contributed to well-documented failures in Dutch government. The parliamentary inquiry prompted by the *toeslagenaffaire* (childcare benefits scandal) identified that all three branches of government had been *blind voor mens en recht* (blind to people and justice), with harsh legislation, institutional culture, and opaque systems reinforcing each other (Parlementaire enquêtecommissie Fraudebeleid en Dienstverlening, 2024). The Raad van State (Council of State), in an unsolicited advisory opinion, warned that algorithmic decision-making obscures the basis of governmental decisions and called for transparency about the rules and data used (Raad van State, 2018). In a factsheet written for Parliament itself, Lokin and Passchier (2024) distilled the same message: the digital execution of statutes carries rule-of-law risks that the legislator currently cannot see. Bovens and Zouridis (2002) showed how discretion migrates from street-level bureaucrats to system designers when law execution is automated; Zouridis, Van Eck, and Bovens (2019) extended this analysis to show how automated discretion differs qualitatively from human discretion. Similarly, van Eck (2018) argued that computer instructions should be disclosed alongside administrative decisions.

These analyses converge on the same structural observation: when law execution is embedded in opaque systems, the constitutional safeguards that depend on transparency cease to function. The problem is not confined to any single scandal or agency, nor is it uniquely Dutch. Australia's Robodebt scheme, where an automated income-averaging algorithm wrongly accused hundreds of thousands of welfare recipients of debt, prompted a Royal Commission that found systemic failures across government (Royal Commission into the Robodebt Scheme, 2023). The United Kingdom's Post Office Horizon scandal, where a faulty IT system led to the wrongful prosecution of roughly 1,000 sub-postmasters, similarly demonstrated how opaque systems can override human judgment and due process (Williams, 2025). Pasquale (2015) described the broader pattern: algorithmic systems that control consequential decisions while remaining opaque to those affected by them. Citron (2008) argued that automated decision systems combine adjudication with rulemaking while adhering to the procedural safeguards of neither, jeopardizing due process. The problem is a property of how modern states have digitalized law execution across the board, without formalizing, publishing, or making inspectable government's actual operative rules.

3 The Constitutional Problem

What follows puts that diagnosis in constitutional terms: which branch gained what, and why the existing safeguards do not correct it.

3.1 Executive Dominance Through Digitalization

Digitalization has deepened the constitutional imbalance. As government shifted law implementation from explicit administrative procedures into software systems, the executive gained a structural advantage. Only the executing branch can reconstruct what its systems actually do. Encoding law as a published specification is precisely that reconstruction, carried out in the open. Legislators passed laws whose effects they could not predict. Courts received cases where the reasoning behind governmental decisions was embedded in code they could not read. Citizens faced decisions based on rules they had no way of verifying.

This frustrates two concrete duties. Ministers answer for everything the administration does (Art. 42(2) Grondwet) and owe the Chamber the information its members ask for (Art. 68). Neither duty can be discharged for a rule that nobody can read.

Scheltema (2015) described the resulting condition as the *bureaucratische rechtsstaat* (bureaucratic rule-of-law state): an administrative state that asks *what behavior does the law prescribe* instead of *how can I achieve the law's purpose*. Digitalization reinforces this mode by embedding rigid interpretations in software. The executive retains ultimate discretion, now through technical choices about how to encode rules rather than through conscious administrative judgment.

Prins (2016) observed that the executive systematically builds digital capacity (sensor networks, predictive analytics, smart data analysis) while the legislature and judiciary fail to invest comparably, and asked what it means when executive agencies commit fully to such analysis "terwijl de rechtspraak onvoldoende competent is om de black box van de verschillende toepassingen te openen" (while the judiciary is insufficiently competent to open the black box of the various applications). Passchier (2020) diagnosed three distinct mechanisms: the executive was already the dominant branch, and digitalization amplified that dominance; government systems increasingly function as black boxes opaque even to their operators; and interconnected data-sharing networks undermine the hierarchical accountability structures that parliament and the judiciary depend on. The translation from law to executable instructions is where law is silently reinterpreted: discretionary space must be filled with concrete decision rules when automating, and those choices become operative without publication or oversight, the migration of discretion to system designers that Bovens and Zouridis (2002) described. Publishing source code alone does not repair this: judges and parliamentarians, Passchier observes, lack the expertise to review how such systems work, so disclosure must extend to the choices, data, and assumptions behind a system. Passchier called for "*trias politica* by design": the constitutional separation of powers must be built into the architecture of government IT systems, not retrofitted through procedural oversight after deployment.

In practice, the executive interprets how laws function, translates that interpretation into code, executes the code, and collects data about the results. The legislature cannot see its enactments in operative form. The judiciary sees only the output, and can only accept the executive's account or reconstruct the logic itself. Citizens experience decisions but cannot verify whether they follow from the law.

3.2 The Gap Between Publication and Intelligibility

The Grondwet requires that law be published: Art. 88 mandates that statutes cannot take effect before publication and Art. 89 extends the same regime to delegated regulation. The Bekendmakingswet (Publication Act) designates the media: the Staatsblad for statutes and Orders in Council[6] (AMvBs), the Staatscourant (Government Gazette) for ministerial regulations and policy rules. This is the principle of promulgation: law must be known to be binding. Publishing the text alone does not meet it if the law's operation remains unintelligible.

The maxim *nemo censetur ignorare legem* (no one is presumed ignorant of the law) frames a legal fiction that presumes citizens have access to legal knowledge. When law is executed through opaque systems, the presumption becomes problematic: citizens cannot satisfy the presumed obligation to know the law because the law, as executed, is not made accessible to them. Formally, government complies with promulgation requirements: the legislature enacts, the Staatsblad publishes. Practically, the law as executed remains unintelligible to those subject to it. A person seeking to know how the law applies often has only one option: formally apply and await the decision of the very executive body that holds the monopoly on its execution. The outcome cannot be calculated in advance, planned for, or verified independently.

The principle of legal certainty demands that citizens and other actors be able to predict what government will do (Voermans et al., 2011). Fuller (1969) articulated parallel requirements for any legal system: among his eight desiderata, law must be promulgated, intelligible, prospective, consistent, and congruent between official action and declared rule. Raz (1979) similarly grounded the rule of law in law's capacity to guide the behavior of its subjects, requiring open, clear, and prospective rules. Dutch *rechtszekerheid* (legal certainty) doctrine and Fuller's eight desiderata converge on the same core demand: law must be knowable in the form in which it actually constrains behavior. Uncertainty about rules precludes planning, remedies, and lawful business. The executive may act consistently according to its own internal logic, but from the outside, the logic is unpredictable. Predictability requires that the rules be published in a form accessible to those affected by them.

The law as executed encodes not just statutory requirements but also administrative policy choices about how to interpret those requirements. The Wet op de zorgtoeslag (Healthcare Allowance Act) determines eligibility based on income and partner status, but depends on definitions from other legislation for concepts like who counts as a partner. The administering organization makes policy choices about borderline cases, and those implementation policy choices determine the practical effect of the law. Some of these interpretive choices are published: a *beleidsregel* (policy rule, Art. 1:3(4) General Administrative Law Act) is a besluit that includes rules on how statutory provisions are interpreted, and it cannot be invoked against a citizen unless it has been published (Art. 3:40 jo. 3:42 General Administrative Law Act[7] (Awb)). But the publication duty reaches only those choices that an organization elects to cast as a *beleidsregel*. A large part of how a law is actually executed is never cast in that form at all: *uitvoeringsbeleid* (implementation policy), work instructions, the derivation of a partner from registry relations, the concrete thresholds and the handling of borderline cases remain internal to the executing organization, known to no one outside it, published nowhere. These

[6] Dutch: *Algemene Maatregel van Bestuur*.

[7] Dutch: *Algemene wet bestuursrecht*.

unpublished concretizations, not the formal *beleidsregel*, are what decides most cases. The proposal in this paper (Section 4) extends an accepted principle, that interpretation binding on citizens must be published, to the operative layer where today it does not reach.

3.3 Substitute Governance

Recognizing that law execution happens through opaque systems, the Dutch state has built elaborate substitute governance structures. The Algoritmeregister (Algorithm Register)[8] (Ministerie van Binnenlandse Zaken en Koninkrijksrelaties, 2022) catalogs algorithms in government decision-making. Proportionality committees and human review requirements are mandated at multiple points in decision processes. Data Protection Impact Assessment (DPIA) and Fundamental Rights and Algorithms Impact Assessment[9] (IAMA) (Gerards et al., 2021) processes attempt to evaluate whether decision systems respect fundamental rights.

Each of these mechanisms compensates for opaque law execution. The Algoritmeregister compensates for the fact that algorithms are not disclosed. Proportionality committees compensate for the inability of citizens and courts to verify whether systems meet legal standards.

These substitutes work only imperfectly. The Algoritmeregister lists systems but rarely provides the actual logic; descriptions of decision rules remain vague or absent. Proportionality committees review systems, but without access to the actual rules, they review descriptions and documentation that the same system builders produce. Assessed organizations control information about their own systems, a conflict of interest that procedural safeguards alone cannot resolve. The LegitiMaat (Van Eck et al., 2022), a systematic method for assessing automated law execution, demonstrates both the sophistication of current efforts and their structural limitation: assessors must reconstruct decision logic from documentation. The Nationale Ombudsman reached a similar conclusion: publishing algorithms or source code does not achieve meaningful transparency if citizens cannot understand how decisions affecting them are made (Nationale Ombudsman, 2021). Published executable specifications would not make these mechanisms redundant; they would give them the object they now lack. A DPIA or IAMA could assess the decision logic itself rather than the builders' account of it. The Algoritmeregister could link each entry to the specification it catalogs, and an assessment method like the LegitiMaat could inspect decision logic instead of reconstructing it. Oversight shifts from auditing descriptions of systems to auditing the rules those systems demonstrably run, and questions that are currently out of reach, such as whether the encoded rule is itself proportionate, become askable. Van den Hoven (2017) draws a parallel distinction in technology design: assessing a system's values after deployment is structurally inferior to specifying those values as design requirements from the start. Current oversight mechanisms are reactive; executable specifications make transparency a property of the format itself.

4 Proposal: Published Executable Specifications

Every automated government decision already rests on a formalization of law, embedded in software systems, spreadsheets, and rule engines across government. Law execution is

[8] https://algoritmeregister.nl

[9] Dutch: *Impact Assessment Mensenrechten en Algoritmes*; published in English as FRAIA.

already formalized, in forms inaccessible to anyone outside the executing organization. This section addresses the form, authorship, and publication requirements of that formalization.

4.1 The Format

The format we propose is a restricted, declarative representation of legislation, administrative rules, and policy. It lacks the infrastructure logic, deployment concerns, data persistence, and user interface handling that characterize software systems. It is a structured notation for legal rules, specifying inputs, transformations, and outputs. The format contains a small number of operations: arithmetic, logical comparison, membership tests, and temporal checks, together with delegation to other legal rules. It prohibits unbounded iteration, recursion, and general-purpose computation. The restriction is designed to guarantee termination and to make the specification readable by legal professionals. Readers engage a single canonical artifact, the specification that runs, at the level that suits them: some read it in its published form directly, others through a user interface layered over the same artifact, with cross-references resolved to navigable links and a scenario-runner that executes the rule against test cases.[10] The tooling meets a reader's competence without substituting an abstraction for the thing that runs. Whether legal professionals can in fact verify such encodings reliably, unaided, is an empirical question, and the research agenda takes it up (Section 12.1); the format is built for that goal, but the goal is a hypothesis to be tested, not a property we can assert.

The design intent is that a legislator can check article by article that the encoding matches the statute, and that an administrator implementing a new law can learn the notation without becoming a software engineer. That check depends on a property legal knowledge engineering identified early: Bench-Capon and Coenen (1992) argued that an encoding is verifiable and maintainable only if its structure is isomorphic to the structure of the source text, one provision mapping to one rule. The format is built to that principle: a specification follows the article structure of the statute it encodes. This is where the proposal differs from simply publishing source code. Publishing code does not help legislators or judges, who cannot read it; but the interpretive choices they need to see are buried in that code, under infrastructure and general-purpose logic. The restricted format makes exactly those choices explicit, in a notation whose whole purpose is to express the legal rule and nothing else. The barrier is meant to drop from "hire a software engineer" to "learn a structured notation".

A reference implementation has been published as a public, openly licensed repository.[11] At the time of writing it contains a curated core of more than twenty encoded Dutch regulations, ranging from benefit calculations such as the Wet op de zorgtoeslag to cross-cutting statutes such as the Awb and the Archiefwet (Public Records Act), each specification paired with a suite of behavior-driven test scenarios. Around that core, the translation pipeline of Section 5.3 has drafted encodings for over a hundred further regulations and has harvested several thousand more from the public law repositories into the same structured format; the drafts await human review, the rest await translation. The notation continues to evolve as it is applied to further legislation.

[10] The reading environment is public at `https://editor.regelrecht.rijks.app`.

[11] The format is documented at `https://regelrecht.rijks.app/reference/schema`; the corpus of harvested and encoded regulations is public at `https://github.com/MinBZK/regelrecht-corpus`. The proposal argues that law execution should be published; the reference implementation and the encodings that accompany it are themselves published, so the claims that follow can be inspected rather than taken on trust.

Figure 1 shows a fictitious example: a City Parking Permit Act, Article 2, encoded in such a format. The statutory text appears alongside its machine-readable interpretation. Inputs are sourced from other regulations by reference: residency status comes from a civil registry, emission classification from a vehicle registry, and fee amounts from a parking fee schedule. No values appear as magic numbers; every concrete quantity is traced to its regulatory source. The computation uses only basic operations (comparison, logical conjunction, conditional) and produces two outputs: eligibility and permit fee. The example is simple, chosen to make the notation legible on the page; the intent is that a legal professional reading it can follow, article by article, how it relates to the statute. A real statute such as the Wet op de zorgtoeslag has the same structure but far more cross-law references, mapped in the dependency graph of Section 5.1 (Figure 4); the fictitious example is kept simple for legibility.

4.2 The Legal Status of a Specification

The format is an interpretation of the law, not the law in original form. If the formatted rules conflict with the statute, the statute prevails; the specification must be corrected. Correction is a legal task, requiring expertise in the statute's legislative history, its place in the broader legal system, and the relevant case law. The format clarifies and operationalizes what the law means without replacing legal judgment.

That a specification can be corrected raises a question the proposal must answer: what happens to a citizen who relied on it before the correction? Once government publishes an executable interpretation, attests to it, and invites citizens to run their own case against it (Section 6.4), that interpretation is exactly the kind of concrete, attributable representation on which the *vertrouwensbeginsel* (principle of legitimate expectations) can bite. Under the three-step test the Afdeling bestuursrechtspraak (Administrative Jurisdiction Division of the Council of State, the highest general administrative court) set out in the Amsterdamse dakopbouw judgment, the test turns on how the published specification would appear to a reasonable citizen, and not on what the administration privately intended (Afdeling bestuursrechtspraak van de Raad van State, 2019). A defensible position, which this paper adopts but legal scholarship must develop, is that correction operates prospectively: the reasonable reliance a citizen placed on the specification in force at the time of their action is protected, so a later correction applies to future cases rather than clawing back decisions already arranged around the published rule. Attestation makes this tractable, because it establishes exactly which specification decided the case, turning a diffuse question of legitimate expectations into a precise one. The register records which specification was in force on a past date, and its custody is open (Section 12.1).

This raises the question of what kind of legal object such a specification is. The nearest thing in existing doctrine is the wetsinterpreterende *beleidsregel*, which may under Art. 1:3(4) Awb consist of rules on how statutory provisions are to be interpreted, and which binds the administration once it is published. The fit is not exact. The specification is executable, and not merely a text an official reads and applies; it is composable, feeding its outputs into the specifications of other laws; and it is meant to be adopted across organizations and bound to execution through attestation, which no *beleidsregel* is. Whether the category stretches to cover it, or a new instrument is needed, is for legal scholarship to decide, and this paper does not settle it.

```yaml
# City Parking Permit Act, Article 2 (fictitious)
article: 2
text: |
  A resident is eligible for a parking permit if their
  vehicle has low emissions. The fee is reduced for
  zero-emission vehicles.
input:
  - name: is_resident
    source: { regulation: civil_registry,
              output: zone_resident }
  - name: is_low_emission
    source: { regulation: vehicle_registry,
              output: low_emission }
  - name: is_zero_emission
    source: { regulation: vehicle_registry,
              output: zero_emission }
  - name: standard_fee
    source: { regulation: parking_fee_schedule,
              output: annual_permit_fee }
  - name: reduced_fee
    source: { regulation: parking_fee_schedule,
              output: reduced_permit_fee }
output:
  - name: eligible            # boolean
  - name: permit_fee          # amount
actions:
  - output: eligible
    value:
      operation: AND
      conditions:
        - { operation: EQUALS,
            subject: $is_resident, value: true }
        - { operation: EQUALS,
            subject: $is_low_emission, value: true }
  - output: permit_fee
    value:
      operation: IF
      cases:
        - when: { operation: EQUALS,
                  subject: $is_zero_emission,
                  value: true }
          then: $reduced_fee
      default: $standard_fee
```

Figure 1: Fictitious encoding of a City Parking Permit Act, Article 2, in the proposed format.

4.3 Who Chooses, Who Publishes

The authority to interpret law for execution belongs to the executive branch. Parliament enacts the statute; the organization charged with its execution makes policy choices about borderline cases and encodes those choices in the specification. That interpretation must be published. Citizens, courts, Parliament, and other bodies can then see which interpretation applies, and challenge it if they think it wrong.

Here the proposal meets its sharpest objection. If the executive both interprets the law and encodes that interpretation, then formalization does not remove the interpretive power, it relocates it: the person who writes the specification becomes, in effect, the interpreter of the statute, now clothed in the apparent legitimacy of publication. Hildebrandt (2020) presses exactly this worry about code-driven law, that fixing a legal norm in code freezes one reading of it and scales that reading across every future case, foreclosing the contestation on which legal certainty actually depends. On this view, publishing the specification would not restore balance to the other branches; it would entrench the executive's interpretation and give it a veneer of transparency. The objection rests on a correct premise: formalization does not remove interpretation, and the executive moves first. But the interpretation it relocates was already being made in code and work instructions that no other branch could see. The interpretive act is not new; publication makes it visible, and the constitution's demand of publicity applies to it for the first time. A silent choice becomes a published one, open to attribution and challenge. Diver (2021) sets out the conditions under which code that regulates can be legitimate at all, chief among them that those subject to it can know it in advance and contest it. Read that way, his criteria set a standard this proposal is trying to meet.

The second half of the freezing worry, that a fixed encoding forecloses future readings, is answered by how the format treats its own outputs: every computed value is a default the competent authority can override. The administration's proportionality duty operates through this override, and Section 4.6 develops it.

When multiple organizations execute the same law, each may arrive at its own interpretation. If 300+ municipalities execute the same statute, 300+ interpretations may exist. Whether that is a problem depends on why they diverge. Where the legislator deliberately left the administration discretionary space, divergence is lawful variation, and publication makes it visible and open to democratic debate. Where the divergence is instead mere interpretation of a nationally uniform norm, differing encodings of the same term are likely a defect. Publication does not by itself resolve this, but it is the precondition for resolving it, because it makes the divergence mechanically detectable and comparable across organizations, and therefore contestable. The format makes the differences visible; whether a given difference is legitimate variation or a breach of equality is a legal question, and acting on it is a legislative or judicial one.

The organization executing the law must actually use the interpretation it publishes, as a mandatory requirement. Publishing is a deception if the published specification is not the encoding the system actually runs. The incentive to publish correctly is strong only if publication actually governs the system. Requiring that the organization use the interpretation it publishes makes publishing binding. The organization commits to that interpretation, making execution transparent and verifiable. Any discrepancy between publication and execution becomes an auditable, remediable failure.

This ordering runs the opposite way to how transparency is usually retrofitted. The specification is not a description maintained alongside the system and hoped to agree with it; it is the encoding the engine executes, so that publishing it and running it are the

same act on the same artifact. A specification that is published but not in production is a defective form of compliance: it presents an interpretation that no citizen's case is actually decided by, and can invite reliance on a rule that does not govern. Where an organization cannot yet execute an encoding, the law's execution remains unpublished, and the remedy is to bring encoding and execution together rather than publish ahead of it.

Where a statute in force must be applied but no faithful encoding yet exists, execution cannot wait for one, and the administration cannot decline to execute the law. Published *beleidsregels* then keep execution lawful in the meantime: the competent authority publishes the operative interpretation it will apply, and that publication is itself inspectable and contestable. This operates at the level of the rule. It fixes in the open how the law is executed for everyone the rule reaches, where an override under Section 4.6 departs from a computed outcome in a single decision.

4.4 Attestation

The requirement that organizations execute what they publish is enforceable only if compliance can be verified independently. Attestation provides this: every trace the engine produces records a cryptographic digest of each published specification the execution resolved, and the organization that ran it signs the whole. A hash is a fixed-length fingerprint of a document, and no one can feasibly construct a second document that has the same fingerprint. Anyone holding the published interpretations can recompute those digests and verify that the versions the organization declares it executed are the published ones. That the declared rule was the deciding rule does not follow from a digest. It follows from re-running the published rule on the inputs the trace records and finding the same outcome. Determinism makes that check possible: the outcome is a function of the specification and the inputs, so no organization can present a trace that both matches the published rule and re-executes to a different result. The check is conditional on the engines agreeing. A verifier who re-executes with a different engine relies on the semantic equivalence that Section 9.1 leaves as an open problem, which is why the engine's identity and the digest of the binary that ran belong in what is signed.

The adversary this design assumes bounds what the design has to do. The administration in question intends to execute the law and can fail at it: it can run a version it did not publish, encode a provision wrongly, or lose track of what its systems do. A signature over the resolved closure and the trace reaches each of those failures, since each leaves a record that does not match the published rule. It does not reach an executive that sets out to deceive, one that signs two traces for a single case and produces whichever suits it, or that chooses the inputs that yield an outcome it has already settled on. Defending against that calls for transparency logs, non-equivocation, and inputs signed by the registers that vouch for them, and this paper proposes none of it. It proposes that an organization be answerable for the account it gives of a decision, and that the account be checkable against the rule it published.

Two things are attested, at two moments. A specification is attested when it is published, and that attestation is a claim of responsibility: the organization asserts that this interpretation governs its execution of the law. The organization need not have authored the interpretation; it may adopt one drafted elsewhere or shared across agencies. It must claim the interpretation as binding on its own execution, and an organization that adopts a specification maintained by another attests the version it executes (Section 11.3).

A run of the engine is attested by the organization that ran it. An unsigned hash of a

```
city_parking_permit_act (2026-07-12 {bsn: 999993653} eligible, permit_fee)
├─Evaluating rules for city_parking_permit_act (eligible, permit_fee)
│  ├─Reference: civil_registry#zone_resident
│  │  ├─Resolving from PARAMETERS: $BSN = '999993653'
│  │  ├─Resolving from DATA_SOURCE: $REGISTERED_ZONE = 'zone-A'
│  │  └─Computing zone_resident
│  │     ├─Compute IN(...) = True
│  │     │  ├─Resolving from PARAMETERS: $REGISTERED_ZONE = 'zone-A'
│  │     │  └─Resolving from DEFINITION: $PERMIT_ZONES = ['zone-A', 'zone-B']
│  │     └─Result: zone_resident = True
│  ├─Reference: vehicle_registry#low_emission   [...]
│  ├─Reference: vehicle_registry#zero_emission   [...]
│  ├─Reference: parking_fee_schedule#annual_permit_fee   [...]
│  ├─Reference: parking_fee_schedule#reduced_permit_fee   [...]
│  ├─Computing eligible
│  │  ├─Compute AND(...) = True
│  │  │  ├─Compute EQUALS(...) = True
│  │  │  │  └─Resolving from PARAMETERS: $IS_RESIDENT = True
│  │  │  └─Compute EQUALS(...) = True
│  │  │     └─Resolving from PARAMETERS: $IS_LOW_EMISSION = True
│  │  └─Result: eligible = True
│  └─Computing permit_fee
│     ├─IF(case 0 matched) = 3500
│     │  ├─CASE 0: True
│     │  │  └─Compute EQUALS(...) = True
│     │  │     └─Resolving from PARAMETERS: $IS_ZERO_EMISSION = True
│     │  └─THEN: 3500
│     │     └─Resolving from PARAMETERS: $REDUCED_FEE = 3500
│     └─Result: permit_fee = 3500
└─Result: eligible, permit_fee = {'eligible': True, 'permit_fee': 3500}
```

Figure 2: The execution trace of one run of the Article 2 specification of Figure 1, as the reference engine renders it. Like the specification it runs, the example is fictitious end to end: the act, the registries, and the case, which uses a reserved test BSN; the run itself is real. Each resolved value records its provenance: a run parameter, a record held by another organization (`DATA_SOURCE`), or a list published in the specification (`DEFINITION`). The fee follows the zero-emission branch. Four reference subtrees are elided (`[...]`); the record the organization signs (Section 4.4) carries the full tree together with the content digest of each specification the run resolved.

published document is a citation rather than an attestation: anyone can compute it, and it commits no one. The signature must cover the content digest of each specification in the resolved dependency closure (Section 9.4). A version label will not serve, since a label resolves through a register the executive operates, and the content behind it can change while the label still verifies. The signature must further cover the evaluation reference date, the identity of the engine and the content digest of the binary that ran, the execution trace, and the decision document, where the run produced one. Signing the trace covers everything the execution resolved, including the inputs, every intermediate value, and any override, so an organization cannot substitute an input once a decision is challenged, and cannot add an override to a decision or remove one from it after the fact. Figure 2 shows the trace of one such run, for the parking specification of Figure 1.

A party who may not lawfully hold the inputs cannot re-execute, and only the signature remains available to them: it establishes which specifications ran and which organization is answerable for the result, without disclosing what the execution resolved. Which signature scheme, key-management regime, and certification arrangement serve this falls outside the scope of this paper (Section 10.3).

A check that anyone can run presupposes that two things can be obtained. The specification behind a signed digest must be retrievable, which the register of published specifications (Section 9.4) must make possible. And the engine must be obtainable, since the outcome is established by running it, and the identity and version the signature covers

are of no use to a verifier who cannot get that engine or one certified equivalent to it (Section 9.1). Both are conditions on which the citizen's check depends, and Section 12.2 asks how they are met.

Attestation says nothing about whether the interpretation is legally correct. Whether it faithfully represents the statute remains subject to judicial review. Attestation guarantees that the review concerns the interpretation the organization committed to, by fixing which specification decided the case.

These duties must be enacted to have consequences. Publication and attestation should be imposed as statutory obligations, most naturally in the Awb or the Bekendmakingswet, and non-compliance should touch the decision itself. The pending Wet versterking waarborgfunctie Awb, the revision of the Awb prepared in response to the *toeslagenaffaire*, strengthens the position of citizens facing automated decisions, and disclosure of the decision rules themselves falls outside its scope (BZK and JenV, 2024). Which defect a missing attestation is, and what follows from it, is for lawyers to settle (Section 12.1). Without consequences of some kind, a broken link between publication and execution is only a finding in a report.

4.5 The Recipient's Check

Every check this paper offers a citizen presupposes that the recipient of a decision holds its trace, and publication and attestation do not by themselves provide it. The attested trace must be available to the person a decision is addressed to, when they receive it and without a procedure of their own. Whether Dutch law already provides it is unsettled: the reasons a decision must state, the file it opens in objection and appeal, and the right of access under the General Data Protection Regulation[12] (GDPR) were each designed for something else, and Section 12.1 asks whether any of them reaches an execution trace. One limit is visible from the start. A trace can resolve values about someone other than the addressee, as an allowance that turns on a partner's income resolves that income, so the recipient may have to receive a partial view, an open question Section 12.1 states.

A decision may also show a valid attestation and re-execute to an outcome other than the one it states. The result is reproducible, since the signature fixes the closure and the inputs, so a divergence is a fact any verifier can establish. Two explanations remain. The verifier's engine may disagree with the one that decided, which is why the engine's identity and version are signed and why certification is on the research agenda (Section 9.1). Otherwise the decision did not follow the specification the organization published, which binding publication to execution forbids, and the decision is defective on that ground. Section 12.1 asks how administrative procedure should divide the burden between the parties once a divergence has been demonstrated.

4.6 Overrides and Discretion

A specification computes an outcome, and the outcome is a default. The format has one way of departing from it: a value the rule produced is replaced by a value from another source, and the trace records the substitution, its source, and the ground given for it. The same technical mechanism is used for legal constructs that differ in who has the power to depart and in what supplies the ground.

[12]Dutch: *Algemene verordening gegevensbescherming* (AVG), the abbreviation commonly used in Dutch legal practice.

```yaml
# Lex specialis: declared in the published specification
override:
  output: bezwaartermijn_weken
  computed: 6                  # Art. 6:7 Awb, the general term
  substituted: 4
  ground: "in afwijking van Art. 6:7 Awb"
  declared_by: "Vreemdelingenwet 2000, art. 69"

# Discretionary departure: recorded in the decision trace
override:
  output: closure_duration
  computed: 6            # what the rule produced (months)
  substituted: 3         # the authority's decision
  ground: "Art. 4:84 Awb"
  reason: |
    Applying the standard closure would be disproportionate
    given the children's housing stability and schooling.
  authority: "Burgemeester van X, besluit 2026-0417"
  produced_by: caseworker  # or the process that ran a batch correction
```

Figure 3: Two departures recorded by the same mechanism. Above, a *lex specialis*: Art. 69 Vreemdelingenwet 2000 replaces the objection term Art. 6:7 Awb sets, the derogation is declared in the published specification, and the declaration is its own ground. Below, a discretionary departure: the burgemeester substitutes three months for the six the rule computed, and states the ground and the reasons for that case. The six months come from the authority's own published *beleidsregel* and not from a statute, which makes Art. 4:84 Awb the ground. The engine records both as a substituted value with its source.

The legislature derogates from its own general rule through *lex specialis*. Art. 69 Vreemdelingenwet 2000 sets an objection term of four weeks *in afwijking van* Art. 6:7 Awb, and the specific provision replaces a value the general one set. The specification of Art. 69 declares the derogation, and the declaration is its own ground.

A competent authority departs from a computed outcome through the discretion the law confers on it. Here the authority deciding the case substitutes the value, and it must state the ground and the reasons that support it in that case (Art. 3:46 Awb). The power to depart comes from the provision that grants the discretion, so where a statute binds and confers none, the specification offers nothing to substitute with.

The mechanism sets no limit of its own. Law limits it, and that law is published in the same corpus as the rule it limits. The Awb states the grounds on which an authority may depart, the cases in which it must, and the duty to give reasons, and the engine executes it as a specification alongside the statute it governs (Section 11.2).

The engine does not distinguish the two. Both are a substituted value with a recorded source, and both can replace an intermediate value as well as a final one, so that every value depending on it is computed from the substitution. The trace records the computed value next to the substituted one, so the computation stays re-executable and the departure stays attributable, and the attestation of Section 4.4 covers the whole without a separate signature. A reader can establish that a value was substituted, on what ground, and by whom. Figure 3 shows both records.

This mechanism keeps the encoding from freezing law into a single reading, and it is where the administration's proportionality duties apply: Art. 4:84 Awb requires departing from a *beleidsregel* where, owing to special circumstances (*wegens bijzondere omstandigheden*), its application would be disproportionate for those affected, and Art.

3:4(2) Awb forbids a decision whose adverse consequences are disproportionate to the aims it serves, which reaches decisions under discretionary statutory powers (Afdeling bestuursrechtspraak van de Raad van State, 2022). Both duties are open-ended and cannot be reduced to a rule, so the override is how they are exercised, and a deterministic engine that admitted no exception could honor neither. Scheltema asks whether they reach a bound power under a formal statute, and Section 12.1 takes that up. Because every override is logged and attested, deviation becomes oversight material.

The record also makes the opposite failure visible. Because every decision records the outcome the rule computed, the cases in which the rule produced a harsh outcome and no authority departed from it can be counted, so an administration can be held to account for the mercy it withheld as well as for the mercy it extended. Under unrecorded practice, the withheld half was invisible. Section 13 takes up what recording a departure costs and what it cannot restore.

This is also the answer to a deeper rule-of-law worry. Waldron (2011) argues that the rule of law demands not only determinate, knowable rules but equally procedures in which their application can be argued; a system in which every computed outcome is a default that can be argued against, before an authority that must state its grounds, is built for that conception of law, not against it.

Specific statutory hardship clauses, which state when and how a particular rule may be departed from, are themselves law and belong in the encoding like any other article; leaving them implicit would hide exactly the interpretive choice this paper argues must be published. The format makes visible which hardship provisions can be expressed as rules and which must remain open discretion. This boundary is a property of law: Hart (1961) called it the open texture of legal language, the penumbra where a general term's application is unsettled. An encoding necessarily takes a position inside that penumbra; publication makes the position visible, and the override keeps it from being the last word. A provision that leaves no default to depart from, discretion the format cannot compute even as a starting point, is a different case, addressed as an untranslatable construct in Section 5.4.

4.7 Automated Overrides

An override need not be a manual act. The event demands an authority that takes responsibility and a ground for the departure; neither requires anyone to have looked at the file. Where a court ruling must be applied to a defined group of past decisions, or a known defect in a published rule must be corrected while the amendment is pending, an agency will override automatically, thousands of decisions in one run. In both, the agency is the authority, and it states the ruling or the defect as its ground. Each of those decisions holds the full record of Figure 3, including how the substitution was produced: by a person who weighed the file, or by a process that never saw it. Grounds differ in which of the two they allow. A fixed correction ordered by a court may be applied mechanically; the proportionality duty of Art. 4:84 turns on the individual case, and an override that arrives in a batch of thousands yet is recorded as a caseworker's weighing refutes itself.

One override on its own stated grounds leaves the rule intact as a default. When overrides fire automatically in every case that meets some condition, the published rule has in practice been replaced, and the logic doing the replacing is in software the public cannot read: the unpublished operative layer of Section 3.2 all over again, except that the pattern now appears in attested traces where an auditor can find it. The encoding should

then be amended, or the substituting rule published as an encoding in its own right: a rule that reaches citizens only through overrides has escaped the publication duty the rest of the proposal enforces.

4.8 Law as a Version-Managed Artifact

Laws are not static. They are amended, suspended, and replaced, and they interact with other laws that change over time. Current practice tracks law changes through the official publication system: a law is enacted and published in the Staatsblad; amendments are published; repeal is published. But the versions themselves are not formally tracked in a way that allows precise comparison or dependency resolution.

Software version management practices can be applied to legislation. In software development, Git-style[13] version control tracks every change to code, provides branching for parallel development, allows comparison of versions, and enables tagging of stable releases. History is retained and queryable. A developer can ask "what changed between version 2.1 and 3.0?" and get an exact answer. The same practices could apply to law. They apply directly to the executable specification as well: because it is a plain-text artifact, what is version-managed is not only the enacted law but the published encoding that executes it.

The Staatsblad becomes the equivalent of releases. A law is published in the Staatsblad; it is a stable release. Parliamentary drafting, Raad van State advisory, parliamentary amendments, and voting are the equivalent of branches and commits. The final publication is the merge to main and the release tag. Versions are attached to each release. Dependencies are tracked: if the Wet op de zorgtoeslag refers to "household composition" as defined in another regulation, that dependency is explicit, and it resolves to the version of the other regulation in force on the date in question. When the other regulation changes, the impact on the Wet op de zorgtoeslag is precisely measurable.

This creates what might be called the "formal state" of the law: the question "what does the law require on date D?" has a precise answer, traceable to the Staatsblad publication on that date and all amendments since. The material state is different: "which rule is actually implemented in the system on date D?" is a separate question. Because the executable specification the system runs is itself published and version-managed, discrepancies between formal state and material state become visible and traceable. If the system is applying an interpretation from the 2020 version of the law but the law changed in 2022, that gap is detectable and can be remedied. Version management makes the temporal dimension of law explicit and operationalizable.

Dependency resolution is genuinely hard. When the Wet op de zorgtoeslag depends on definitions in three other regulations, each of which is updated at different times, what is the "current" law? It depends on the date in question. Version management must track temporal consistency: resolving dependencies as of a particular date requires using the versions of each dependent law that were active on that date. Olthof and Van Andel (2025) formalize this requirement as *chronolexografie* (time-indexed recording of legal status), tracking the *rechtstoestand* (legal status) at every point in time and enabling precise reconstruction of which rules applied to which facts on any given date.

[13] Git is a widely used open-source distributed version control system; see `https://git-scm.com`.

4.9 Four Execution Modes

Different laws are triggered in different ways and require different system architectures. Publishing a specification covers law execution only if the format can express every way a law is triggered, so the four modes below set out that requirement in full. We distinguish them by their triggering mechanism.

Active execution is the most familiar mode. A legal determination is initiated for a specific case, on application or ex officio: a citizen applies for a health care benefit, a company requests a building permit, or the government determines child benefit eligibility for a newly registered child. In each case, the law executes with specific parameters and produces a result. The system is synchronous: input in, decision out.

In **reactive** execution, the law fires when a specific state transition occurs. When a government body makes a decision, administrative law activates an objection period. When a government agency creates a document, freedom of information law activates, imposing obligations to consider public access. The trigger is an event, a change from one state to another, and the law responds to that change. Reactive rules are edge-triggered: they fire on the transition, not on the resulting state.

Generative execution covers law that creates, modifies, or empowers the creation of other law. The legislative procedure is one example: a proposal enters parliament, passes committee review, votes, and executive assent, producing new law. Delegation is another: a statute authorizes the executive to set rates by regulation, or empowers municipalities to adopt local ordinances. In each case, the output is not a decision about a person or situation, but a change to the body of law itself.

Finally, **verificative** execution applies where laws impose standards that must hold at all times: the fire code requires buildings to meet safety standards, environmental law requires emissions to stay below thresholds, archival law requires records to be retained for specified periods. These rules are not triggered by an event; they express invariants. Verificative execution continuously checks state against requirements and flags violations. The distinction from reactive execution is temporal: reactive rules fire on a state transition (edge-triggered), verificative rules check whether a condition holds right now (invariant). A building that was always non-compliant with fire safety does not need an event to trigger the violation; the violation exists as long as the state persists.

These four modes require different architectures. Active execution is request-response. Reactive execution requires event detection and routing. Generative execution operates at a meta-level, transforming the legal state itself. Verificative execution requires continuous state monitoring. In practice, these modes compose: a permit application (active) produces a decision that triggers objection rights (reactive) and imposes ongoing permit conditions (verificative).

The decision logic is the same across the four modes; only the trigger differs, and the trigger is not an implementation detail: when an objection period starts to run is itself a rule of law. The format therefore makes triggers part of the published specification. An article declares, in its published form, the events it responds to: a hook names the kind of legal product it attaches to, and any article producing such a product declares so. When the Wet op de zorgtoeslag computes an allowance and produces a decision, the Awb article that sets the six-week objection period fires alongside it, with neither law naming the other (the open arrow in Figure 4); the trigger relation is published, versioned, and attested on both sides, rather than remaining an unpublished configuration in the scheduler of the executing organization. The reference implementation exercises the active and reactive modes this way. Verificative and generative execution raise their own

architectural and constitutional questions (who may trigger execution, what infrastructure monitors invariants, what legal status the output has), which remain open on the research agenda (Section 12).

4.10 Procedure as a Staged Lifecycle

Section 4.9 described the objection period as a reactive rule that fires when a decision is produced. Administrative law embeds that firing in a staged procedure: the Awb treats the *besluit* as the outcome of stages running from the opening of the case through preparation to individual notification (*bekendmaking*, distinct from the constitutional *bekendmaking* of regulations) and possible *bezwaar* (objection).

Obligations attach at different stages, and the stage decides the deadline: the six-week objection term starts the day after notification (Arts. 6:7 and 6:8(1) Awb), so a specification that fires it at the moment the decision is taken computes the wrong date. Whether a hearing is owed beforehand is rule-determined in the same way (Arts. 4:7–4:12). The stage a case is in is therefore legally significant, and a specification must include it as part of the published trigger: a rule declares the stage at which it fires, so the term computation reads the notification date where the law says it must.

Encoding these stages makes the procedural law of the Awb a published artifact in its own right, separate from the substantive law that decides the case. The Wet op de zorgtoeslag computes the allowance; the Awb governs how the case containing that computation moves from application to notification to objection, and it does so in the same way for every law it attaches to, save where a special statute derogates; the derogation is then a published rule in the specification. Attestation (Section 4.4) lets whoever holds a case's execution trace check that the outcome followed from the published rule; a published lifecycle extends that check to the procedure the trace records: the term computation can be re-run against the notification date, and the trace shows whether the record reports a hearing at the stage where the rules prescribe one. Whether a hearing took place, and whether it was adequate, the trace cannot show; that remains a question for the case file and for oversight.

Because the encoded procedure is what the engine runs, the artifact that publishes the lifecycle is the artifact that moves the case through it: the rule declaring the stage at which a term fires is the rule that advances the case from one stage to the next. The orchestration a case-management system performs today, holding which stage a case is in, when a term starts, whether a hearing is owed, becomes a property of the published specification, where now it is logic in a workflow engine that no one outside the organization can read. How far that goes, whether an engine executing the published lifecycle can replace the process system rather than only expose its logic, is a question the research agenda takes up (Section 12.2).

4.11 Executing Across Organizations

Laws refer across organizational boundaries, and the references differ in kind. When the Wet op de zorgtoeslag sets its income threshold as a percentage of the statutory minimum wage, it borrows a definition and a calculation: applying the referenced rule to known inputs yields a number, and that number has no legal effect of its own. When the same law rests on the income assessment the Belastingdienst issues, it relies on a *besluit*, a decision with its own legal effect (*rechtsgevolg*) that the law reserves to the competent

administrative organ (*bevoegde bestuursorgaan*). The first kind of reference can be executed locally: the referring organization runs the other law's published rules and needs nothing from the other authority. The second must be accepted as an authoritative input. The line between the two is legal; the specification records, article by article, which references resolve to calculations and which to decisions.

One question the proposal raises is whether an organization that can run another body's published specification may also execute that body's law and issue decisions under it. Executing published rules yields a computed value; issuing a *beschikking* (an individual decision addressed to a person) remains an act of the competent *bestuursorgaan*, and running the rules confers no such authority.

4.12 Relying on Another Body's Decision

Local execution changes the position of the organization that must rely on another body's decision. An organ that adopts another body's decision as an input owes its own duty of careful preparation (Art. 3:2 Awb) and must be able to reason from that input to the decision it takes (Art. 3:46). Where the value is an authentic datum from a base registration (*basisregistratie*), the registration regime adds a duty to use it and a duty to report well-founded doubt (*gerede twijfel*) to its holder. Each registration has its own act; for the population register these duties are Arts. 1.7 and 2.34 of the Wet basisregistratie personen (Wet BRP). In chain decisions the duty is today discharged largely on institutional trust (van Eck, 2018). Attestation gives the duty a concrete object without performing it. A signature establishes authorship and integrity, which is provenance; the duty is substantive, and satisfying it means having a reason to believe the value is sound. A decision-reference arrives signed by the organ that took it, over the value, the specification closure it was computed under, the case it concerns, and the date (Section 4.4). The receiving organ verifies that signature without needing the inputs behind the decision, which in the case of the income assessment it may not lawfully access anyway.

The signature makes the supplier answerable for the value. It does not make the value true.

Contesting a value is not the same act as contesting the decision derived from it. The citizen's routes run through a request to correct the registered datum (Art. 2.58 Wet BRP, a refusal of which Art. 2.60 makes a *besluit* open to objection) and through objection against the source *besluit* itself. The *terugmeldplicht*, the duty to report suspected errors back to the holder of the registration (Art. 2.34 Wet BRP), obliges the receiving organ to report *gerede twijfel* to that holder. The report suspends the duty to use the datum while it is under investigation (Art. 1.7(2) Wet BRP). It does not suspend the source *besluit*, the term for objecting to it, or the receiving organ's own decision.

Where the receiving organ does lawfully hold the inputs, it can re-execute the published rules as a further check. That check has limits: re-execution confirms the application of the rules and says nothing about the soundness of the inputs. A lawful override does not appear as a mismatch, since the trace records the computed value next to the substituted one and re-execution reproduces the computed value (Section 4.6). A detected mismatch is a ground for doubt, and the receiving organ must act on doubt. It cannot set the supplying decision aside, since competence over that decision stays with the source, and it is not being asked to. It is being asked whether to build its own decision on the value, and Arts. 3:2 and 3:46 Awb forbid it to proceed on a datum it has concrete reason to doubt. The route is to report the doubt and to hold its own decision until the doubt is resolved.

The question is distinct from where the computation runs, which is a matter of privacy and security; the question here is which body's act has legal effect. They interact only one way: executing another law's rules locally is a placement option only where the reference resolves to a calculation. Where it resolves to a decision, no placement choice arises; the decision is taken by the body to which the law assigns it.

5 Diagnostic Value of Encoding

When law is encoded as executable specification, several layers of currently invisible practice become visible. Encoding forces questions that informal administration allows to remain unanswered: what data does this rule actually require? What conditions determine which path the law follows? What happens when inputs fall outside expected ranges? What other laws does this rule depend on? Today, these questions arise only when implementation begins. Meuwese and Timmer (2022) applied the term "law smells" (Coupette et al., 2023) to such anti-patterns in legislation that digital translation exposes; executable specification makes their detection systematic.

5.1 Dependency Graphs and Cross-Law References

Every law refers to other laws. The Wet op de zorgtoeslag refers to the Zorgverzekeringswet (Health Insurance Act) for who counts as insured, to the minimum wage act for its income threshold, to the tax code for its asset test, and, through the partner concept, to the population register. These dependencies are stated in the law itself, mainly in definitions. But the full dependency graph exceeds unaided human reasoning. Encoding makes this graph explicit, navigable, and checkable.

Figure 4 charts this graph for the Wet op de zorgtoeslag to a depth of three references, drawn from the statutory texts. A single allowance decision rests on at least ten other regulations, and not only the ones a reader would guess: the threshold income is defined as a percentage of the statutory minimum wage, the partner concept passes through the general tax code before it reaches the population register, and eligibility reaches the Penitentiaire beginselenwet, because the allowance requires an insurance that can be suspended. Every solid arrow but one is a reference stated in the statutes themselves; the exception is instructive. Art. 24 Zorgverzekeringswet suspends the insurance without naming detention, or the law that administers it, so the link to the Penitentiaire beginselenwet is an interpretive step, one that executing organizations make today invisibly. One law binds without any reference at all: the Awb attaches to the decision itself and starts the six-week objection period, the hook of Section 4.9. Encoding means that such a graph no longer needs to be assembled by hand, reference by reference, but becomes mechanically derivable, for every encoded law at once, and stays current as the laws change. Nor does the graph stop here: each of the referenced laws makes references of its own. Some laws are at the center of hundreds of dependencies; others are isolated. Some dependencies are temporal: law B can only execute if law A has executed first. Some dependencies are organizational: a reference may resolve to a calculation another organization's law defines, or to a decision only that organization may take, the distinction Section 4.11 draws.

A law depends not just on another law's current version, but on the version that was active on the date relevant to the case. When the Wet op de zorgtoeslag was calculated on 2020-03-15, it needed specific versions of all dependent laws that were active on that date. Making this temporal dimension explicit forces clarity about the distinction between *ex*

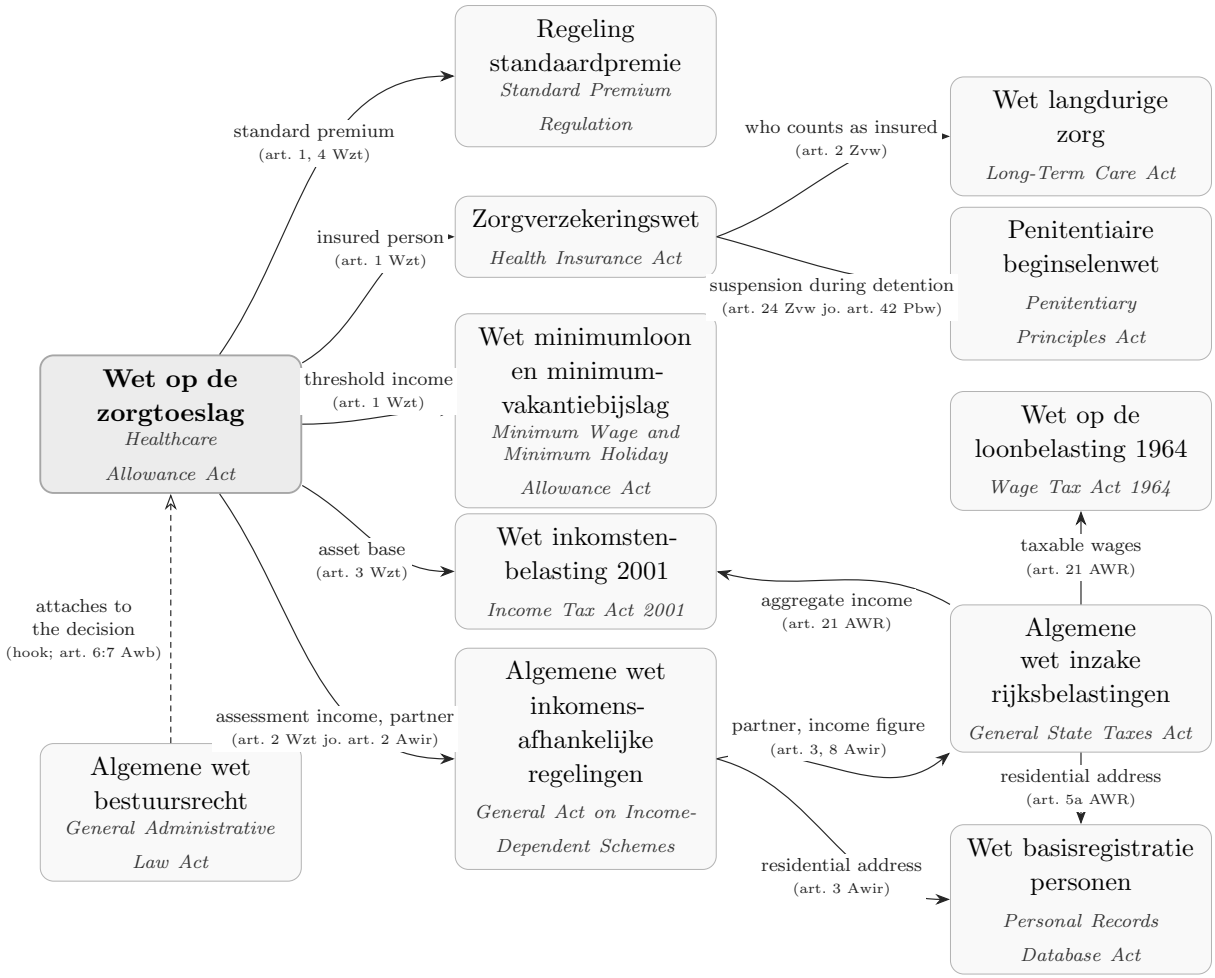

Figure 4: The web of statutory references behind a single Wet op de zorgtoeslag decision, drawn from the statutory texts as in force on 2025-01-01: an arrow points from a regulation to a regulation it refers to, labeled with the borrowed concept and the provision making the reference. The eligibility check reaches the Penitentiaire beginselenwet (Penitentiary Principles Act): the allowance requires an insurance, and Art. 24 Zorgverzekeringswet suspends it. The Awb is drawn with an open, dashed arrow: nothing in the chain refers to it; it attaches of its own force to the decision, the hook relation of Section 4.9.

tunc and *ex nunc* effects: does an amendment apply retroactively to all past cases, requiring recalculation, or only from the date of enactment forward? Version management must encode this distinction explicitly, because the answer determines whether past decisions remain valid or must be revised.

Policy-level dependencies are harder than law-level ones. Laws follow strict formulation rules (Aanwijzingen voor de regelgeving (Instructions for Legislation)), making their structure predictable. Policies like administrative guidance and implementation policy often lack uniform publication standards, are sometimes embedded in code or spreadsheets, and may not even be formally documented. The encoding process confronts organizations with orphaned rules: policies once based on formal guidance but no longer maintained, rules added by developer misunderstanding, exceptions handled through undocumented heuristics. Each orphaned rule becomes a choice: formalize it, delegate it to human discretion, or recognize it as a bug that should be corrected.

5.2 Structural Analysis of Encoded Legislation

Once rules are formalized as executable logic, they become amenable to formal methods and program analysis. Several analyses become possible (F. Nielson, H. R. Nielson, and Hankin, 1999). Completeness checking asks: for every permitted input combination, does the law provide a defined output? Are there input combinations that fall through every conditional branch and produce no decision at all? Currently, such gaps come to light only through court challenges; encoded law allows systematic testing of completeness before deployment. Consistency checking tests whether the same inputs can ever produce contradictory outputs across chains of dependent laws. Divergent definitions across regulations are a common source: one law reads "household" as people sharing rent, another as people sharing utilities. Encoding forces each rule to name the definition it uses, so a divergence a human reader can easily miss becomes explicit, and the engine applies exactly one without ambiguity. Where dependent rules can still yield outputs that cannot both hold for the same facts, encoded law detects the contradiction automatically.

Dead code detection finds branches in the law that no input can reach: the conditions guarding them cannot be satisfied together, whatever the case. It is an analysis of the specification and not an observation about cases. Dead branches are candidates for repeal, so the analysis must never report a live branch as dead.

Order-sensitivity detection determines whether execution order matters when multiple laws depend on the same data that changes over time. This exposes subtle vulnerabilities where the sequence of operations determines the outcome. Current informal practice often does not notice these problems until they cause scandals.

None of these analyses prescribe policy. They reveal problems that, without encoding, are difficult to detect until they have already shaped a real decision. The decision what to fix, and how, remains with the legislature and the executive.

5.3 Translating Law at Scale

A corpus of specifications has to come from somewhere: the diagnostics of this section presuppose it, and the scale of what would need encoding is easy to underestimate. The national repository alone holds some 45,000 regulations in more than 100,000 consolidated versions; the repository of decentralized regulations of municipalities, provinces, and water boards holds over 330,000 more.[14] Execution needs the historical versions too: a case is decided under the law in force at the time (Section 9.4). Encoding a corpus of that size by hand is the standing objection to proposals of this kind (Section 11.1). The reference implementation treats the conversion as what it is, translation, and assigns the labor to large language models: a translating agent renders statutory Dutch into the format, article by article, following a procedure that is itself published. Language models translate; they never execute. What executes is the deterministic engine, running a specification that people have reviewed and an organization has adopted; no language model is consulted when a decision is computed.

The procedure is decomposed into what agent tooling calls "skills": written work instructions, much like those given to a new caseworker. A skill is a short document, readable by anyone; it states when a task arises, which steps to follow, what a good result

[14] Counts from the public law repositories, July 2026: the Basiswettenbestand behind `wetten.overheid.nl` ("zo'n 45.000 regelingen met meer dan 100.000 toestanden", some 45,000 regulations in more than 100,000 consolidated states) and the SRU search service behind `lokaleregelgeving.overheid.nl`, whose records count consolidated versions.

looks like, and what to do when something does not fit the usual pattern. The agent reads the instruction when it starts the task and follows it, and because the same file is versioned and published alongside the specifications it helped produce, anyone can see exactly what the translator was told to do.[15] One skill retrieves the consolidated text from the public law repositories. Another searches the Memorie van Toelichting (Explanatory Memorandum), in which the government explains a bill to Parliament, for worked examples: the memorandum typically spells out what the rule should yield in concrete cases (a household at a given income, the allowance it receives). Each worked example becomes a Behavior-Driven Development (BDD) scenario (North, 2006), an executable test case, so the standard a translation is tested against comes from the memorandum rather than from the translator; where an example and the article's text diverge, the text governs. Generation itself runs as a loop: generate the encoding, validate it against the schema, run the scenarios, and repeat, up to a fixed number of rounds, and whatever still fails is reported rather than patched. The mandate is deliberately narrow. Each encoded provision may interpret only the text of the article it is attached to, and values from other provisions enter through references, never inline. Where the statute is redundant or roundabout, the encoding must be too; the requirement is fidelity, not elegance.

Language models hallucinate, and a pipeline that uses them on law must be built around that failure mode. After generation, a reverse pass walks the encoding in the opposite direction: every input, every condition and threshold, must trace back to a sentence in the statutory text. Untraceable elements the logic does not need are removed as invention. Where the logic does seem to need one, it goes to a human as an assumption to confirm. Logic that traces to a different provision than the one it appears in is refactored into a reference: a correct result in the wrong place still breaks the correspondence between article and encoding. A fidelity audit compares the encoding against the letter of the statute and flags conditions that leaked in from the Memorie van Toelichting, which documents intent but does not bind, qualifiers that were dropped, and provisions given the wrong legal basis. A drift check verifies that the text being translated is the version in force on the specification's validity date. These checks are tractable precisely because the format is restricted: a specification is a finite composition of a small operation set (Section 9.2), so the reverse pass has a known, finite list of elements to check, a checklist that generated general-purpose code would not offer. And where the operation set cannot express a construct at all, the translator's standing instruction is to flag rather than improvise: the untranslatable annotation of the next section is the mandated default for that case, since a translator with no way to say "this does not fit" will otherwise produce an encoding that validates and misrepresents.

None of this changes who answers for the result, though it moves something this paper cannot leave implicit: the translation step it identifies as the point of silent reinterpretation (Section 3.1) is now first performed by a model, and a first draft is not neutral; its choices frame what the reviewer sees. The checks above do not catch everything, and the design does not need them to, because the draft binds no one. A generated specification is a draft until the organization charged with executing the law has reviewed it and adopted it as its published interpretation (Section 4.3); adoption is the interpretive act that counts, attributable and open to contest, and the adopted text executes afterward, not the process that drafted it. From adoption on, the specification is versioned and

[15]The translation skills are published in the reference implementation's repository, https://github.com/MinBZK/regelrecht.

attested like one written by hand, and every decision is bound to it through the trace of Section 4.4. Language models thus appear at the two edges of the pipeline and nowhere in between: translation on the way in, subject to those checks and to human adoption, and explanation on the way out (Section 8.3), where a model renders a finished trace into plain language without evaluating a rule. Between those edges the computation is deterministic, which is also what keeps the engine itself outside the AI Act's definition of an AI system (Section 7.3). The same division makes the transition of Section 11.1 plausible. Designing the procedure is the slow part; running it is not. Once the instructions are written and proven on one law, any number of agents can follow them on any number of laws at once, so the effort that went into the first translated statute is not paid again for the ten-thousandth. From there, how many laws are translated in a day is a question of computing capacity, not staffing. Human review does not multiply this way, and that is where the scarce attention goes.

5.4 Publishing What the Format Cannot Yet Say

Encoding an article for diagnosis works only if the article can be encoded at all. Some articles cannot: a provision's meaning can exceed the operations the format offers, a rounding rule the explicit rounding operations do not cover, or a bracketed table the notation cannot yet represent. One response is approximation: imitate the rounding with arithmetic that agrees on most inputs, or inline the table as a chain of conditionals. A translator left without an alternative, and a language-model translator in particular (Section 5.3), will reach for exactly these workarounds. Each of them yields a specification that executes and looks faithful while diverging from the statute, reopening the gap between published and executed law inside the publication itself. A different kind of gap admits no such workaround: a provision phrased "naar het oordeel van de minister" (at the discretion of the minister) is not missing an operation the format could grow, since the outcome is not the format's to compute at all.

Instead, the format records the gap. An untranslatable construct becomes an annotation on the article it belongs to, published with the specification: the annotation names the provision and the construct, and states why the operation set cannot express it. Figure 5 shows such an annotation for a provision that reads "het bedrag wordt naar boven afgerond op hele euro's" (the amount is rounded up to whole euros): before `ROUND`, `CEIL`, and `FLOOR` existed, such a provision would have been annotated as untranslatable, with the annotation recording exactly what the format needed, until the operation set grew to cover it. An article whose annotation a reviewer has not cleared does not execute, so a specification containing one does not go into production. The gate comes before use and not in front of a citizen: it exists to stop a law being executed on assumptions made by people with no authority to make them. The reviewer clears execution of the remainder, not the gap: the judgment says that the construct genuinely exceeds the operation set, that the encoding excludes it rather than imitating it, and that what is left is fit to run as an openly partial rule. Clearance does not make the gap tolerable, and nothing begins to approximate the missing construct. A remainder that runs without a step the statute prescribes still diverges from the statute; clearance changes only the status of that divergence, published and open to challenge instead of hidden. The engine records the annotation in the trace of every run of the article, and supplying the missing part remains human work, case by case,, until the operation set grows to cover it. The clearance is comparable in kind to the override record of Section 4.6, though at the level of the specification rather than the

```yaml
# Untranslatable construct, annotated on the article
article: 12
untranslatables:
  - construct: "naar boven afgerond op hele euro's"
    reason: "operation set has no rounding primitive"
    suggestion: "add ROUND, CEIL, FLOOR operations"
    accepted: false
```

Figure 5: An untranslatable construct annotated on the article that contains it: what the text says, why the operation set cannot express it, and what the format would need. The `accepted` field records that a reviewer has cleared the article for execution; until it is set, the engine refuses any decision that depends on this article, and after it is set only the expressible remainder executes, with the annotation recorded in its trace.

individual case: an override records one authority's decision in one case (a batch correction writes many such records, one per decision), while a clearance records one reviewer's judgment, with its reasons and its author, published with the specification and applying to every case the annotated article touches until the format grows. A silent approximation leaves no record to contest.

Alongside that gate, an audit mode serves corpus-level analysis: the engine marks instead of refusing, and the mark propagates to every derived value, so an analyst can measure how far a given gap reaches through an encoded corpus and how many decisions it would touch. Wherever a result would inform a case-specific decision, however, the engine still refuses; audit mode is reserved for corpus-level analysis that never produces one.

The reference implementation supports both modes, but detection is incomplete. Structural mismatches, such as a table the schema cannot hold, are caught by a mechanical schema check. A semantic mismatch, an article whose encoding computes cleanly while meaning something else, can pass every automated check; catching it depends on the human reader that the isomorphism principle of Section 4.1 exists to serve. So the guarantee has a limit: whether a gap is declared at all depends on the translator that annotates it or the reader who notices it, and some gaps will escape them. The design ensures something narrower: once a gap is declared, it cannot be silently approximated, since declaration blocks execution, clearance is a recorded human judgment, and the annotation stays in every run's trace. An undetected semantic mismatch produces exactly the silent approximation this section is meant to foreclose, without ever reaching that block. The gap is temporary by design: Section 9.2 describes how the format grows a missing operation in public; the mechanism here governs what happens to the law in the interval before it has grown. The annotation mechanism exists for that growth. The annotations are structured data, readable in aggregate by people and by machines: each `suggestion` across the corpus is a vote for the operation the format needs next, weighted by how many articles depend on it. Walking those votes is itself a task a skill can describe (Section 5.3): a language-model agent following it reads every annotation in the corpus, identifies the most blocking gap, and drafts the proposal to close it. Deciding that proposal stays with people, and the operation enters the format like every operation before it; what one translator flags as untranslatable today becomes an ordinary operation the moment enough other articles need it too.

6 Implications for Constitutional Actors

A published specification changes what each constitutional actor can do. We take parliament, the executive, the courts, and citizens in turn.

6.1 Informed Lawmaking

Scheltema (2018) argued that legislation in the responsive state must be manageable for ordinary citizens. Parliament currently enacts law in prose and cannot test what a statute will do before voting on it. The Memorie van Toelichting documents legislative intent, but nothing verifies whether execution matches that intent. Divergence is discovered years later, through casework or scandal.

A specification can catch that divergence before the law takes effect. The worked examples an Memorie van Toelichting gives, a calculation for a stated case, can serve as a test oracle: the published specification either reproduces them or it does not. In experiments this proved useful, but it rests on a condition current practice does not meet. The examples in an Memorie van Toelichting are frozen at the moment of drafting: an amendment does not necessarily update them, and where amounts are indexed annually the stated figures fall out of step with the law they illustrate, so the example no longer tests the rule in force. Making Memorie van Toelichting examples usable as an oracle would require either a discipline that maintains them across amendment and indexation, or a versioned conformance suite published with each law that tracks it as it changes. Which of these fits the legislative process remains open, and the research agenda takes it up (Section 12.2).

Executable specification makes the Memorie van Toelichting falsifiable. A proposed amendment can be run against population statistics to show which groups are affected and by how much, as LexImpact[16] does with micro-simulation, which needs no individual case. Parliament can compare current law against proposals quantitatively, strengthening the projections that inform legislative judgment by showing how a change would affect the actual population. Section 11.5 addresses the institutional capacity this requires.

BDD (North, 2006) offers a model for this. Scenarios describing what the law should achieve are written first and serve as acceptance criteria. Statutory language is then drafted to satisfy those scenarios. If the text cannot be encoded to pass them, the text needs revision. The scenarios discipline the drafting process without constraining parliamentary deliberation on the policy itself.

Execution data reveals which provisions are frequently triggered and which are rarely used. Rarity is not the dead code of Section 5.2: a provision no case has met may be reachable still, and repealing it is a political judgment. Parliament can weigh maintenance cost against demonstrated benefit, making deregulation empirical.

When multiple laws govern the same population, their combined effect is often unknown. A citizen subject to health care benefit, housing benefit, and child benefit rules simultaneously experiences a net outcome that no single statute describes. Because the specifications are formal and executable, they can be mechanically composed into a consolidated specification that preserves each law's contribution to its policy goal. The consolidated version can be tested against the originals by running both on the same inputs and comparing outcomes; testing builds confidence in the composition but does

[16] https://leximpact.an.fr

not prove equivalence over all inputs, which remains the formal-methods problem of Section 12.2. Where overlapping laws produce unintended thresholds, gaps, or contradictions, composition exposes them. Where simplification is possible, the consolidated specification offers Parliament a candidate replacement, tested against the laws it would replace.

6.2 Stewardship of the Specification

Today, implementing a new law means funding an IT project, building software, and operating the system for its lifetime. Under executable law, a new law is encoded as specification and a shared engine executes the decision logic, while the surrounding operational systems (case management, data intake, payments, appeals) consume that specification instead of re-implementing the rules. The executive's work shifts from infrastructure development to legal analysis: understanding the law deeply enough to encode it correctly and making policy choices about ambiguities. The organizational model becomes stewardship-based. As long as the law is in force, the specification must be maintained.

The executive's discretion shifts accordingly, from technical implementation to legal interpretation. Both are exercises of discretion, but they are qualitatively different. Technical discretion was opaque; legal discretion is publicly debatable.

Publication creates accountability in both directions. External actors can challenge the published interpretation. The executive itself can monitor whether execution matches declared policy, detecting discrepancies before they surface through scandals or litigation.

6.3 Running the Law in Court

Courts reviewing administrative decisions currently either accept the executive's explanation or reverse-engineer undocumented decision logic. Van Amerongen and Schuurmans (2019) showed that the existing judicial review framework fails for algorithmic decisions: the administrative judge cannot check internal consistency or logical derivation when decision logic is opaque. The courts have refused to accept that opacity. The Afdeling bestuursrechtspraak has held that where an administrative decision rests on a calculation model, the government must fully disclose the choices, data, and assumptions built into that model so that affected parties can contest them (Afdeling bestuursrechtspraak van de Raad van State, 2017); and the SyRI judgment struck down risk-profiling legislation under Art. 8 European Convention on Human Rights (ECHR) precisely because the state disclosed nothing about the models it ran (Rechtbank Den Haag, 2020). These courts demanded the rules as executed. A published executable specification supplies exactly that, in a form the court can run: the court can independently verify what outcome the specification yields for the facts at hand and, where it disagrees, identify whether the encoding diverges from the statutory text or the statutory text itself compels the outcome. Where the statute compels it, the court's own powers stop. Art. 120 Grondwet forbids it to review an Act of Parliament for constitutionality, and the Harmonisatiewet judgment extends that bar to unwritten legal principles. Conformity with a treaty under Art. 94 stays open, which is how the SyRI legislation fell, and so does exceptive review of the rules below the statute. Publication changes none of this, and it is not meant to. What it changes is who else can see a compelled outcome: a harsh rule that survives judicial review becomes visible, and countable, to the legislature that enacted it. With the lifecycle of Section 4.10 published alongside the substantive rules, the court can also test procedural

regularity, such as whether the objection term was computed from the notification date, the same way it tests the outcome. The court remains the final arbiter of what the law means: it may use the specification and is never bound by it.

6.4 Running Your Own Case

A citizen can take the published specification and run it against the data government holds about them. Before applying, the citizen can check eligibility and predict the outcome. After receiving a decision, the citizen can check that the outcome is the one the published rule yields on the inputs the trace records, which the decision must supply (Section 4.5), and identify where any divergence lies. The same citizen can enter hypothetical data to explore scenarios: what happens if I move, if my income changes, if I take on a partner? The maxim *nemo censetur ignorare legem* becomes less fictitious when the law, as executed, is actually accessible.

Verification need not stop at the outcome. Dutch administrative law requires that a decision, when notified, state the remedy that lies against it: whether *bezwaar* or *beroep* (appeal to a court) is available, before which body, and that a period applies (Art. 3:45 Awb; the period itself is set by Art. 6:7). Today this *rechtsmiddelenclausule* (remedy clause) is composed by the executing organization, and clauses are sometimes wrong or missing. Administrative law absorbs the error after the fact: Art. 6:11 Awb excuses a late objection where the lateness cannot reasonably be attributed to the party, and a defective clause is one ground the case law recognizes.

Because an article declares the kind of legal product it produces (Section 4.9), the remedy provisions that attach to that product can be encoded once as hooks, and the applicable route derived rather than composed anew for each decision: the error moves from every notified decision to the one-time encoding, where it can be caught before any decision relies on it, instead of being forgiven case by case under Art. 6:11 Awb. A decision under the Awb ordinarily has an objection route, unless statute or the Awb itself provides another one; a traffic penalty under the Administrative Enforcement of Traffic Rules Act[17] (Wet Mulder) is contested first before the *officier van justitie* (public prosecutor) and then before the *kantonrechter* (subdistrict court). The derived route is attested as part of the trace like any other resolved value, so a party can verify the stated way to challenge a decision the same way it verifies the decision itself. The period itself remains set by statute; the derivation establishes only which statute's clock applies. A correct clause says nothing about whether the underlying decision is right.

Most citizens will not run a specification themselves, and the proposal does not assume they will, though personal software agents may change that. The Wetenschappelijke Raad voor het Regeringsbeleid has documented the gap between knowing and doing: the capacity to act on available information, *doenvermogen* (capacity to act), is unevenly distributed, and it is lowest precisely when circumstances are hardest (Wetenschappelijke Raad voor het Regeringsbeleid, 2017). Intermediaries will read a published specification first: social-legal counsellors, legal aid lawyers, welfare officers, journalists, and the tools they build and share. An advisor can only check a decision against a rule that can be read and run, so publication makes effective intermediation possible at all. The citizen who never opens a specification still benefits from what publication makes possible for those who do.

[17] Dutch: *Wet administratiefrechtelijke handhaving verkeersvoorschriften*; commonly known as the Wet Mulder.

6.5 The Gaming Objection

Openness of the decision rules does raise the gaming objection: if anyone can compute exactly where a threshold lies, some will arrange their affairs just below it, and publishing the state's fraud-detection logic would teach evaders what to avoid. A scope rule answers it. Only the decision logic must be published: the norm as applied to facts to produce a legal outcome. How the administration selects whom to verify, the control and enforcement logic, is a different artifact, and existing law already treats it as a category of its own: supervision is one of the interests against which a request for information is weighed (Art. 5.1(2)(d) Open Government Act[18] (Woo)), a balancing test rather than an exemption. Nothing in this proposal requires publishing risk models, and the SyRI judgment marks the limit from the other side: enforcement interests cannot justify a system whose operative rules escape public scrutiny altogether (Rechtbank Den Haag, 2020). As for arranging one's affairs around a published threshold: that possibility is created by the legislator's choice of a precise rule over an open standard (Kaplow, 1992), not by publication of the rule. Sophisticated parties already optimize against thresholds through advisors; secrecy about the operative rule disadvantages only those who cannot afford one.

The same specification serves all constitutional actors, though not each of them in the same way. An objection officer and a judge hold a case's inputs and can re-execute it. Auditors with a lawful basis for the data can re-execute across a population. Parliament and the public hold no case data and cannot re-execute a decision; publication gives them the rule itself, which they can read and analyze with no case at all. Verifiability is an architectural property of the system.

7 Data, Privacy, and Information Asymmetry

When law execution is formalized and its data requirements are explicit, fundamental rights protections that are currently aspirational become enforceable. Data minimization, proportionality, and purpose limitation are legal obligations under the GDPR and the ECHR, but currently lack operational mechanisms: no one can verify which data a system accessed, in what order, or whether a less invasive path existed. Executable law specifications make these questions answerable. The data a rule requires, the order in which it accesses that data, and the routing of queries between organizations all become visible and open to systematic analysis.

7.1 Execution Path Optimization and Privacy by Design

When law is executed, the order of operations matters for privacy impact. The Wet op de zorgtoeslag requires knowing if someone is an insured person (as defined in the Zorgverzekeringswet). That law suspends insurance rights during detention (Art. 24 Zorgverzekeringswet); detention status is administered under the Penitentiaire beginselenwet; Figure 4 traces this chain. The Wet op de zorgtoeslag also requires checking if the person is at least 18. Both checks are legally authorized. But if age is checked first and the person is 3 years old, the prison check never executes; sensitive data is never accessed. Check prison first, and sensitive data is accessed unnecessarily.

Systems often access every authorized data point, regardless of whether a more efficient path exists. Execution paths are invisible, so no one optimizes them for privacy. Executable

[18] Dutch: *Wet open overheid*.

specifications make paths explicit and analyzable.

Execution logic can be analyzed to identify paths that avoid unnecessary access: where one path's data accesses are a strict subset of another's, preferring it is a mathematical property of the encoded law. The comparison is not always that clean. Sensitivity is a partial order: a path that touches detention status and one that touches income and partner data are not comparable by mathematics alone, and ranking them is a normative choice. The format does not make that choice; it exposes it, so the weighting can be set as policy rather than fall out of an engine's arbitrary evaluation order. The engine could then be required to follow the declared least-invasive path. A trace asserts the order of operations rather than proving it; only the query logs of the providers themselves, held by other organizations, can establish that order independently (Section 9.3).

The current GDPR requires data minimization but provides no operational mechanism for how. Organizations minimize what they are legally entitled to check, not what they actually need. Encoding privacy as part of law execution makes a vague obligation a recorded one.

7.2 Proportionality as Executable Requirement

The principle of proportionality requires that invasive measures (accessing sensitive data, restricting liberty) be necessary, suitable, and proportional to the aim. Normally assessed qualitatively, case by case, proportionality can also be analyzed at design time, before the law is deployed.

When law is encoded, one can analyze all execution paths: which data points are needed, how sensitive they are, and whether a less invasive path exists that produces the same outcome. This is design-time analysis of the rule's own proportionality, distinct from case-specific proportionality (which the administration owes under Art. 3:4(2) Awb wherever its power leaves room to weigh interests, and which the court reviews). Does the law access sensitive data only when legally necessary?

Such analysis supplies formal evidence for the subsidiarity limb of the necessity assessment (ECHR Art. 8) (European Court of Human Rights, 2008); the assessment as a whole remains a normative judgment about pressing social need and proportionality in the strict sense. A policymaker proposing a new rule can analyze it before submission: show where a less invasive execution path yields the same outcome, or discover that one does and redesign before affecting anyone. A DPIA can then use design-time analysis of the encoded rule.

7.3 The European Layer

The EU regulatory layer strengthens this proposal. The AI Act imposes risk management, logging, transparency, and human oversight on high-risk systems, which include AI used by public authorities to decide eligibility for public benefits and services (Annex III) (AI Act, 2024). A deterministic engine executing published declarative rules is arguably not an AI system under the Act at all: the definition turns on a system's capability to infer how it generates outputs, and the recitals exclude systems that automatically execute rules defined solely by natural persons. The objection is in the word "solely". Section 5.3 assigns the first draft of an encoding to a language model, and read literally the recital would exclude any specification a model helped produce. That reading is hard to reconcile with the operative definition, which is in Art. 3(1) and turns on the capability of the deployed

system to infer how it generates its outputs. The engine infers nothing: it evaluates a fixed specification, and identical inputs yield identical outputs on every run. "Defined" names the act that fixes a rule as the rule that governs, and officials perform it when they adopt the specification (Section 5.3). A statute drafted by a civil servant is defined by the legislature that enacts it, and a specification proposed by a model is defined by the officials who adopt it and answer for it. The drafting instrument does not enter the deployed system, and it is the deployed system the Act regulates. The proposal is therefore not a compliance answer to the AI Act; on the axis that matters here it is more demanding than the Act, since a published, attested specification discloses the operative rule itself where the Act requires documentation about the system. Where organizations do use genuinely inferential components around that deterministic core, the Act applies to those components on their own terms: a risk model used to select cases can itself be high-risk under Annex III, and the language models used for translation and explanation fall under the general-purpose regime.

The GDPR already governs the decisions themselves. Automated decision-making with legal effect is regulated (Art. 22), and the data subject holds a right to meaningful information about the logic involved (Arts. 13(2)(f), 14(2)(g) and 15(1)(h)). In *SCHUFA* the Court of Justice read Art. 22 broadly: an automatically computed score that a third party relies on strongly is itself a decision, drawing even preparatory computation into the regime (CJEU, 2023). Under that reading, much of what executing organizations compute today is already subject to an explanation duty. A published executable specification with a per-decision trace is the strongest form that "meaningful information about the logic involved" can take: not a description of the logic, but the logic.

8 Conceptual Shifts and New Ways of Thinking

The changes so far concern institutions and data. Three further shifts are conceptual: what an entitlement is, from whose side execution runs, and what counts as an explanation.

8.1 Citizen Entitlements

Current service design frames benefits and entitlements as "offers" from government. Government decides what to provide, how, and on what terms. Citizens must find the relevant schemes, apply, and prove eligibility. The citizen bears the burden of access.

That framing holds only while no one outside the executive can check what the law actually grants. Once the specification is published, executable, and attested, that check is open to anyone who holds the specification and the case data it needs (Section 4). The reading end that Section 3 placed with parliament and the courts now reaches the citizen: a person can run the published rule against the data government holds about them and compute the outcome the law prescribes for their case (Section 6.4). A citizen no longer takes on trust what they can compute.

This changes what a benefit is. Under the principle of legality the executive applies rules Parliament enacted, so when such a rule says a person qualifies, that qualification is settled by the statute the executive is bound to apply. As long as the determination was made only inside systems the citizen could not inspect, the entitlement was hard to tell apart from a favor the administration chose to extend. A rule the citizen can run turns it into an *aanspraak* the citizen can demonstrate. *Rechtszekerheid* stops depending on the

executive's word for it, because the law as executed is knowable in the form in which it decides the case.

The burden of justification moves with it. If a citizen can show against the published rule that they meet its conditions, a denial is no longer the default that follows an imperfect application. The administration has to say why the rule it published does not yield what the citizen computed from it. Proactive delivery then fits without eroding agency: government can identify the people a rule entitles and offer to process their claims, while the choice to act on the offer stays with the citizen.

8.2 Citizen-Centered Execution

Many visions of proactive service assume government should see all aspects of a person's life to determine eligibility comprehensively. This is impossible, and undesirable: government cannot see everything, and a government that tried to would itself threaten liberty.

A better model is citizen-centered execution. The rule a citizen can check against their own case (Section 8.1) is one they can also run for themselves, locally, in their own security context, understanding their rights and obligations before interacting with government. They know their circumstances better than government does: what information is relevant, what should remain private, what trade-offs matter. The citizen decides whether to claim a right they have identified themselves. Government provides the service as APIs at a lower level. Citizens have their own tools, running the same rules, exploring scenarios with their own data.

The same properties would let citizens delegate this work to software agents acting under their control, drawing on the same published rules the government executes; whether such agents can be trusted with consequential decisions, and who supplies them to citizens who cannot build their own, are open questions this paper does not resolve.

8.3 Explaining Decisions from Execution Traces

When a specification executes, the engine produces not only a result but a complete execution trace (Figure 2): a tree of every value resolved, every condition evaluated, and every branch taken. This trace is a formal, inspectable artifact. It records which inputs determined the outcome, in what order conditions were tested, and where the decisive branching points lay. Unlike a *motivering* (statement of reasons) reconstructed after the fact, the trace is generated during execution and corresponds exactly to what happened.

The trace is machine-readable but not citizen-readable. Translating it into natural language is, however, a bounded task: the trace contains a finite set of resolved values and decision points, each with a defined meaning in the specification. A language model can transform this structured record into a plain-language explanation at an appropriate reading level. The reference implementation demonstrates this for Dutch benefit legislation, producing B1-level (CEFR) paragraphs that explain to a citizen why a particular outcome was reached, referencing the actual values from their case. The language model performs no rule evaluation: the outcome and the trace are fixed by the deterministic engine before any text is generated. But it would be too quick to say the model adds no legal reasoning at all. Choosing which of the resolved values were decisive, ordering them, and phrasing the counterfactuals is itself an act of selecting and framing reasons, and under Art. 3:46–3:47 Awb the reasons communicated to a citizen are part of the *motivering* of the decision. A generated explanation that misstates the decisive ground, for instance

presenting income as decisive when partner status in fact was, is a *motiveringsgebrek* (defect in the statement of reasons) regardless of whether the underlying trace is correct, and a citizen misled by it may wrongly conclude that objection is pointless and let the six-week term lapse. This is why Pasquale and Malgieri (2024) warn against using generative models to justify administrative decisions: a plausible imitation of a human explanation can launder unaccountable reasoning. Bos (2026) evaluated exactly this across three Dutch laws and found that an explanation pipeline constrained to the execution trace was substantially more faithful to it than an unconstrained baseline, which scored higher on readability but was markedly less likely to state the decisive condition; the explanations citizens perceived as understandable were not always the ones experts judged legally correct, so the clarity of the prose is no signal of its fidelity. The architecture here answers part of that objection, since the explanation is anchored to a verifiable trace the citizen can fall back on rather than to a black box, but it does not dissolve it. The organization remains responsible for the content of what it communicates, and the trace, not the generated prose, is the authoritative record.

This separation is architecturally significant. The execution engine is deterministic and verifiable. The explanation layer is stochastic and, as an artifact, disposable: if a rendering is unclear it can be regenerated without touching the decision, and the citizen can always fall back to the trace itself or to the specification. Disposable as an artifact does not mean inconsequential in content, however: while it stands, a communicated explanation has the legal weight set out above, so regeneration is a remedy for an unclear explanation, not a license to treat its contents lightly. Zuurmond et al. (2023) propose a complementary human-centered explanation framework for rule-based decision-making systems, addressing how such explanations should be structured, adapted to the recipient, and communicated. Together, formal traces and human-centered explanation design make the gap between executable law and citizen understanding bridgeable.

9 Technical Foundations and Implementation Requirements

None of this works without technical machinery that today exists only in part. This section covers the four hardest open problems.

9.1 Multiple Engines and Semantic Equivalence

In the proof of concept the same specification was executed by engines written in Python, Go, and Rust/WebAssembly, an informal check that an engine is simple enough to be reimplemented independently, not a rigorous verification. The reference implementation now under development targets Rust for production.

For the system to work constitutionally, different engines must produce identical results. If a citizen's engine gets a different answer than the government's engine, law becomes plural and unpredictable. Achieving this requires a formal definition of what "equivalent execution" means and verification that multiple engines achieve it.

Semantic equivalence is not obvious. Do floating-point calculations round identically (Goldberg, 1991)? Do error conditions have the same handling? Do temporal checks handle edge cases the same way? These are not mere implementation details; they have legal consequences. A EUR 0.01 difference in benefit calculation might be legally insignificant

or might constitute unlawful underpayment, depending on statutory precision.

This is a research problem in computer science. Multi-version consistency semantics from distributed systems apply. Certified engines must be proved to produce identical outputs on identical inputs.

9.2 Format Design: Restricted Operation Set and Decidability

The operation set (Section 4.1) follows from deliberate design decisions about what must be expressible in law and what exceeds law's function.

At the time of writing, the schema (v0.5.6) fixes 25 operations in eight categories: arithmetic (`ADD`, `SUBTRACT`, `MULTIPLY`, `DIVIDE`), selection (`MIN`, `MAX`), explicit rounding (`ROUND`, `CEIL`, `FLOOR`), comparison (`EQUALS` and the four ordering comparisons, over numbers or dates), logical combination (`AND`, `OR`, `NOT`), collection membership (`IN`, `LIST`), a guarded conditional (`IF`), and calendar arithmetic (`AGE`, `DATE`, `DATE_ADD`, `DATE_DIFF`, `DAY_OF_WEEK`).[19] The set is small enough to enumerate in one sentence, which is itself the point: everything a specification can express is built from these operations, so a reader who has learned them has learned the whole language. One design detail has legal weight of its own: rounding is never implicit. An engine never rounds a value by itself, not even money; a rule that rounds must say so explicitly, with direction and precision, the way statutes themselves prescribe rounding.

One absence illustrates how the set is governed. There is no aggregation over collections: no `SUM`, no `COUNT`, no iteration over the elements of a list. A rule can add or divide named values, so a fixed twelve-month average is expressible, but it cannot aggregate over a collection whose size varies by case, the shape of computation at the heart of Robodebt's income averaging (Section 2.4) (Royal Commission into the Robodebt Scheme, 2023). The omission is not an oversight but a policy: the operation set is kept as small as possible and grows only when encoding real law runs into a real limit, at which point the operation is added and a new version of the format is released. The laws encoded so far did not need collection aggregation; current encoding work does, so bounded aggregation, which preserves termination over a finite collection, will enter the format the way every operation entered it: explicitly, versioned, and in public, not as an engine's private workaround. By the time this paper is read, the count above will likely have grown; the schema's public version history records when and why each operation entered, which is the version discipline of Section 9.4 applied to the format itself.

Unbounded iteration and recursion are prohibited because law should terminate; infinite loops prevent decisions and unbounded recursion risks stack overflow, circular dependencies, and non-termination. The restriction is designed to guarantee termination within provable bounds and also ensures readability: a legal professional can learn a small set of operations but cannot be expected to learn an entire programming language; the operation set is sized to what law needs.

Within a single specification, termination is immediate: with no unbounded iteration and no recursion, every computation is a finite composition of operations and therefore terminates (Winskel, 1993). That guarantee covers a specification's own computation. It does not extend to a law whose structure fails to terminate. A legislator can, through delegation and cross-references, build a circular dependency between specifications. Detecting that is itself a graph analysis over the date-resolved dependency graph of Section 5.1,

[19] The full schema is published at `https://regelrecht.rijks.app/reference/schema`; its content digest belongs in the attested closure like any other artifact (Section 4.4).

and where a cycle exists the engine halts on it and reports the defect, the way it reports dead code. That halt is by design. Repairing it is a legislative act: an amendment to the statute, or published *uitvoeringsbeleid* that keeps execution possible until the statute is amended. Unlike an untranslatable construct (Section 5.4), where a reviewer can clear the expressible remainder for execution, nothing lifts this halt; the engine and the schema never work around it. A non-terminating law is a defect the proposal makes visible where today it stays hidden, and the format enforces the division of roles this paper argues for throughout: the engine executes the law and leaves the fixing to the legislator.

9.3 Authorization Derived from Execution

Access to another party's data is usually administered separately from executing the law, in a layer of contracts and access lists. Publishing and attesting the rule that needs the data (Section 4.4) makes that layer derivable from the rule itself. The claim depends on a distinction. A rule's need for a datum is not a legal basis; the basis must be laid down by law (Art. 6(1)(e) and 6(3) GDPR), and executing a rule creates no basis. Execution can supply the authorization instead: the operational decision to grant this organ access to this element, a decision that today is made in that layer. The specification ties every data element to the provision that requires it, so the purpose and necessity of an access (Art. 5(1)(b)–(c)) are recorded in the trace instead of asserted in an authorization database.

An exchange between two organizations is two acts of processing, and the same reasoning holds on both sides: the receiving organ processes under the statute it executes, the providing organ discloses under a statute of its own, and neither basis follows from the other's need. The specification records, next to each cross-organizational reference, the provision that authorizes the disclosure; a reference that can name none is caught at encoding time, like dead code (Section 5.2), where today it appears only in a later audit. The provider still decides, but against a published, attested artifact: whether the signature is valid, the engine certified, the requested element within the rule's stated needs, and the cited provision open to this organ. Special-category and criminal data (Arts. 9–10 GDPR) always need their own explicit basis. In all cases the basis comes from the law; the specification makes it explicit and its absence visible. Execution derives the authorization: the access decision administered today in contracts and access lists, checked instead against the published, attested rule and the provision it cites. The infrastructure this assumes, certified engines, key management, and legal recognition of attested claims, spans the agendas of Sections 12.1 and 12.2.

9.4 Version Management and Temporal Consistency

Law versioning must track every change: amendments, repeals, retroactive effects, temporal validity windows. When law B depends on law A, and both have changed multiple times, "what is the current law?" requires specifying a date, and the version management system must support temporal queries like "what was the law on 2020-03-15?" and retrieve consistent versions of all interdependent rules. This is hard: standard package management (npm, pip) picks the latest compatible version. Law cannot do this; temporal consistency requires specific versions from specific dates (Snodgrass, 1999). A law amended with retroactive effect must be encoded with explicit time travel: "as of this date, the previous version's effect is reversed". Amendments with prospective effect only must specify precisely when the change takes effect.

The Staatsblad becomes the source of release tags: a law published on 2020-03-15 is version 2020-03-15, and amendments published on 2021-06-30 create version 2021-06-30. Dependency references themselves are not version-pinned: when the Wet op de zorgtoeslag references the Zorgverzekeringswet, resolution selects the version of each law in force on the relevant date. The versions actually used must therefore be recorded per execution, and the attestation of Section 4.4 must cover that resolved closure. Without it, two decisions can declare the same specification hash and compute different functions, and a verifier has nothing well defined to re-run.

The published specifications and their versions are held in a register, and whether that is one register or one for each publishing authority, the checks proposed here ask the same two things of it. It must return the content that a digest fixes, so that a verifier holding a signed digest can obtain the specification it identifies. And what was in force on a given date must stay what it was, so that a decision taken years ago can be re-executed against the law as it then read. Section 12.1 asks who holds such a register, and what keeps it from being rewritten. Infrastructure must track all versions, manage temporal queries, and alert when amendments affect dependent laws. It requires coordination across Parliament, the Staatsblad, and executing organizations. It is a significant change to how law is managed.

Two sources of change bypass the Staatsblad entirely. Directly applicable EU regulations enter into force through the Official Journal of the EU, and a single judgment can strip a provision of its validity overnight, as the SyRI ruling did to the legislation it reviewed (Rechtbank Den Haag, 2020). The version model therefore needs change events from non-legislative sources: a judicial decision or an EU act is an event in the version history of every specification that depends on the norm it touches.

10 Out of Scope: Design Choices Not Taken

The design space for a machine-readable encoding of law is large, and the format's shape reflects deliberate exclusions as much as deliberate inclusions. Several exclusions, in particular the absence of infrastructure logic, user-interface handling, data persistence, and general-purpose computation, are already discussed where the format is introduced (Section 4.1) and where the operation set is motivated (Section 9.2). Two further exclusions warrant explicit treatment, because they reject well-argued alternatives in the same problem space: encoding law as controlled natural language (the RegelSpraak and ALEF tradition), and encoding law as linked data (the semantic-web tradition). A short closing subsection records other boundaries that are out of scope for this paper.

10.1 Controlled Natural Language (RegelSpraak / ALEF)

The Belastingdienst developed RegelSpraak and its authoring environment ALEF on a sound premise: a controlled subset of Dutch lets policy specialists author and read execution rules in something close to the language they already use, lowering the authoring barrier and keeping the rule artifact recognizable to the legal professionals who own its content. That choice has been validated within the Dutch tax administration over years of operational use, and any proposal to encode law machine-readably must answer why it does not adopt the same surface form.[20]

[20] ALEF was released as open source in July 2026: https://github.com/belastingdienst/ALEF.

This proposal chooses a structured notation instead, for three reasons rooted in the constitutional argument of this paper. First, semantic equivalence across multiple engines (Section 9.1) is harder to specify against a natural-language surface. When the published artifact is prose-shaped, two engine implementations may legitimately disagree on parse, scoping, or operator precedence without either being demonstrably wrong, and every ambiguity in the surface becomes a hidden interpretive choice inside the engine. A structured notation settles those choices in the specification itself, where they can be reviewed by parties other than the engine author. Second, court-readable diff requires structural identity. When law is amended, judges, citizens, and the legislature need to see exactly what changed between versions (Section 9.4). Diffs over controlled natural language conflate surface edits with semantic edits, since reformulations that read identically can have different operational meaning, and edits that read differently can be operationally identical. Diffs over structured notation do not. Third, authoring is not the constitutional bottleneck this paper diagnoses, and the translation pipeline of Section 5.3 could produce either notation, which makes authoring ergonomics weigh even less. RegelSpraak optimizes for the author of the rule; this proposal optimizes instead for the reader downstream of execution: parliament checking whether its enactment was encoded faithfully, a court reconstructing how a decision was reached, a citizen verifying that the rule applied to their case is the rule the executive published. The constitutional asymmetry of Section 3 is at the reading end, not the writing end.

The attestation mechanism (Section 4.4) needs the same restriction, for reasons that have nothing to do with reading. The attestation supports re-execution: a verifier runs the published rule on the recorded inputs and compares the outcome. That check is a right only if it always yields an answer, and an operation set that admits no unbounded iteration and no recursion (Section 9.2) guarantees that it does. The check is meaningful only if the attested digest names a function rather than a document. Take the digest of a natural-language surface and what has been named is a text whose operational meaning depends on a compiler, which must then be published and attested in its turn, and which becomes a second place where the published rule and the executed rule can diverge. The gap this paper exists to close is now inside the compiler. Where the published artifact is the artifact that runs, there is no such place. The restriction also leaves open the possibility that a party who may not lawfully hold the case data can still check a decision: a succinct proof of correct execution over committed inputs, whose feasibility Section 12.2 takes up.

These optimizations are not mutually exclusive. A natural-language authoring layer that compiles to this format is a coherent direction, and would inherit both the authoring ergonomics of RegelSpraak and the verification properties this paper requires of the published artifact. The present proposal claims only that the published, court-citable, attested form must be the structured one. The choice is not a judgment that ALEF was wrong; it is a choice that the constitutional problem this paper addresses pushes the optimization target to a different point.

10.2 Linked Data and Semantic Web Encodings

A proposal to publish law machine-readably across government naturally raises the question of why it is not framed as a semantic-web project. The format proposed here is not built on RDF (Resource Description Framework), OWL (Web Ontology Language), or SHACL (Shapes Constraint Language), and the omission is deliberate. That tradition produced

mature standards for legal documents and rules, notably Akoma Ntoso for legislative document markup (Palmirani and Vitali, 2011) and the Dutch-initiated MetaLex for legislation interchange (Boer, Hoekstra, and Winkels, 2002); these mark up and exchange legal sources, where the present proposal is an execution format instead.

Linked-data modeling pushes domain experts toward atomic, universal ontologies in which every concept is decomposed to its most general form, producing schemas that the original modelers themselves struggle to navigate after a few years. This notation's operation set is deliberately coarse and law-shaped: it models statutes at the level of articles, conditions, and references, the level at which the legislator wrote them, not at the level of a generic ontology of obligations, rights, and roles. The format follows law as it is, not law as it would be if rewritten as a logical theory. A universal schema is no more attainable: a single ontology spanning health care, tax, and immigration law is not a realistic engineering target, since domain semantics diverge rapidly and the abstractions that survive the divergence are too generic to do useful work. The format scopes to one statute at a time, with explicit, date-resolved cross-references between regulations (the `source: { regulation, output }` pattern in Figure 1), and accepts that the result is a graph of independently maintained artifacts rather than a unified knowledge graph. The use case diverges too: linked data was built for automated discovery and inference over an open, distributed graph, whereas verification, attestation, diff, and audit are local determinism problems over a single published, signed artifact. RDF and OWL bring machinery (inference rules, open-world assumptions, schema alignment) that those use cases do not need and pay for in modeling cost and infrastructure cost.

This format borrows three things from the linked-data tradition: explicit references between regulations as first-class data, date-based version resolution of those references, and publication as the primary act that gives a specification legal weight. The disagreement is about ontology and graph topology, not about machine-readability as a goal.

10.3 Other Boundaries

Several further items are out of scope and should not be expected from this paper. Formal verification of the engines themselves is a research question (Section 9.1), not a deliverable; the present claim is that the format admits such verification, not that any current engine has been verified. The legal-theoretical status of the proposed format as a new instrument category is raised at Section 4.2 and deferred to legal scholarship. Concrete cryptographic protocol design is out of scope. The paper states what the check requires of signatures, key management, and certification (Section 4.4) without selecting a scheme for any of them. The language-model translation pipeline of Section 5.3 is described as far as the constitutional argument requires; which models and agent tooling perform the translation is out of scope, since any translation process that clears the same published validation gates and ends in human adoption is acceptable. Finally, the proposal is not a replacement for existing oversight bodies such as the Algoritmeregister, proportionality committees, the Autoriteit Persoonsgegevens (Dutch Data Protection Authority), or the Algemene Rekenkamer (Netherlands Court of Audit); its aim is to make their work tractable by giving them an artifact they can inspect.

How a structured specification and a controlled-natural-language encoding compare empirically, and under what conditions the natural-language authoring layer sketched in Section 10.1 preserves the constitutional properties of the published form, are open and testable questions the research agenda takes up (Section 12.2).

11 Transition and Institutional Change

The remaining obstacles are institutional rather than technical: how large the transition is, where to begin, what support it needs, and what resistance to expect.

11.1 The Scale of Transition

Implementing executable law across Dutch government is institutional transformation. The scope is vast: tens of thousands of regulations in hundreds of thousands of consolidated versions (Section 5.3), executed by hundreds of organizations, affecting millions of citizens.

The transition involves multiple phases: establishing basic infrastructure (specification format, certified engines, version management systems); encoding the highest-value laws (cross-cutting laws like Awb, GDPR, Woo, Archiefwet that apply uniformly across government and deliver disproportionate value); encoding sectoral laws (social benefits, permits, compliance); and integrating with parliamentary processes, judicial review, and citizen interfaces. This will take years, require sustained political commitment, and face institutional resistance as organizations worry about job loss, role changes, and participation requirements.

11.2 Cross-Cutting Laws as Strategic Priority

Cross-cutting laws are the highest-value starting point. These are laws that apply to almost all government organizations: the Awb objection and appeal provisions, the GDPR data protection requirements, the Woo access obligations, the Archiefwet retention rules.

Today, each organization implements these independently. Awb objection procedures work slightly differently at each agency. Some organizations follow Woo obligations rigorously, others grudgingly, others not at all. The Archiefwet is interpreted inconsistently. From the citizen's perspective, a patchwork of implementations replaces the one government that should apply one law.

Encoding cross-cutting laws once and deploying across all organizations supports uniform application, with citizens experiencing one government applying the same rules everywhere. This makes divergence from the equality principle detectable and contestable: where citizens in different municipalities and agencies are treated inconsistently under a nationally uniform statute, the difference becomes visible and can be challenged.

Beyond uniformity, the decision logic that is currently re-implemented per organization per law becomes a published, shared, verifiable artifact. The surrounding operational systems remain and consume the specification instead of embedding their own copy of the rules, so the payoff accrues over time.

11.3 Ownership of Cross-cutting Law Specifications

Encoding once and deploying everywhere also presupposes an owner, and Dutch institutional arrangements do not currently provide one. Responsibility for execution is fragmented by design: ministers answer to Parliament for the statutes of their own department, agencies such as the Employee Insurance Agency[21] (UWV) execute at arm's length under the Kaderwet zelfstandige bestuursorganen (Framework Act on Independent Administrative Bodies), and municipalities execute national law in *medebewind* (execution

[21] Dutch: *Uitvoeringsinstituut Werknemersverzekeringen*.

of national legislation by decentralized authorities), accountable primarily to their own councils. Nobody in this landscape is responsible for a shared decision-logic artifact, because no such artifact has existed. A shared specification therefore needs an explicit division of roles. The department responsible for a statute maintains and publishes its canonical specification. Every organization that adopts it attests to the version it runs (Section 4.4), and adoption never transfers responsibility for the decision itself, which remains with the administrative body that takes it. Where the law leaves local discretion, an organization departs from the canonical encoding by publishing its own variant, so that the divergence is visible.

11.4 Tooling and Legislative Support

Lawmakers work with word processors and spreadsheets. Parliamentary drafting has no specialized tools. If executable specifications are to become part of the legislative process, lawmakers need purpose-built tooling. A law browser and editor showing multiple perspectives on selected articles: the statutory text, the proposed specification, context and dependencies, notes and commentary. If no specification exists yet for a law, the tool proposes one via the translation pipeline of Section 5.3. The tool includes validation processes, forcing consistency between text and encoding.

Such tooling must support the entire legislative process from first draft through Royal signature, tracking both statutory language and specification changes as Parliament debates, validating consistency, and reporting conflicts so the final enacted law includes both statutory text (published in the Staatsblad) and executable specification (published in a new register). Education is equally necessary, requiring legislators and legislative drafters to understand executable law and its role through courses at the Academie voor Wetgeving (Academy for Legislation), certificates, and training working groups. This includes legal and constitutional education about what this approach means and implies.

11.5 Organized Readership

Publication changes the balance of power only if the other branches can actually work with what is published. A register nobody consults changes nothing, and Section 3.3 showed how thin transparency becomes when disclosure is not connected to anyone's working routine. Readership must be organized, not assumed.

For Parliament this is a question of institutional capacity, so no individual member has to heroically read YAML. The Tweede Kamer (House of Representatives) already has the seeds: Bureau Wetgeving (the House's legislative drafting office) gives members legal support in drafting amendments, and the Dienst Analyse en Onderzoek (the House's Analysis and Research Service) provides analysis for the committees, including a technical check on the budgetary coverage of amendments. Extending these services with the capacity to read, run, and draft specifications would let a member see what an amendment does before voting on it, with all parliamentary groups served equally. France has already built such a capacity: LexImpact, a unit of the Assemblée nationale built on OpenFisca, quantifies deputies' amendments on income tax and social contributions, and its tools are open to citizens, researchers, and journalists as well (Assemblée nationale, 2020).

Three guardrails keep this capacity from inverting the hierarchy it serves. Whether a proposal can be encoded never conditions the admissibility of an amendment; a member's right to amend is not subject to a type checker. The specification follows the adopted text,

never the reverse: where encoding and enacted language diverge, the language wins and the encoding is corrected. And the tooling used in the parliamentary phase is operated by or for the Chamber, not by the executive whose interpretation it exists to check; otherwise the asymmetry this paper diagnoses returns through the very instrument meant to remove it.

11.6 Institutional Resistance and Transitional Justice

For executing organizations, the location of their interpretive work changes first of all. The policy choices that staff now encode in internal systems, work instructions, and parameter files would be made in public, subject to challenge. The surrounding operational work (case handling, data intake, contact with citizens) remains, and where straightforward determinations are automated, capacity shifts toward the decisions that require judgment, toward encoding, and toward oversight. The change this paper proposes is a transparency change before it is an automation change, and its friction points are accordingly about exclusivity rather than headcount: organizations lose the private ownership of interpretation that currently shields their choices from scrutiny.

Without transition support (retraining, new roles, compensation), resistance will be justified and overwhelming. Institutional design must recognize that humans, organizations, and livelihoods are at stake.

Additionally, discovered discrepancies require transitional justice frameworks. If a law was executed incorrectly for years due to misencoding, are past decisions void? Do affected citizens get compensation? These are questions of justice about how retroactive correction should work.

12 Research Agenda by Discipline

The open questions raised in the preceding sections are collected here as a research agenda, ordered by discipline.

12.1 Legal-Constitutional Research

The legal framework for executable law remains underdeveloped.

The specification as a legal instrument.

1. Section 4.2 argues that a published executable interpretation fits the wetsinterpreterende *beleidsregel* (Art. 1:3(4) Awb) at its core and departs from it in being executable, composable, and bound to execution through attestation. Is that category to be stretched, or is a new instrument required, and what challenge, amendment, and revocation procedures attach to it either way?

2. If encoding forces the executive to make explicit the concretizations it now makes silently, which of those choices are permissible as an executing interpretation, which are a *beleidsregel* the organ may set for its own power without any delegation (Art. 4:81 Awb), and which amount to generally binding rules that the principle of legality permits only on a statutory basis (for an order in council, Art. 89 Grondwet)? Does publication reveal that parts of current execution have been operating without such

a basis? The same question arises for the generative mode that Section 4.9 defers to this agenda: where a specification produces a change to the body of law itself, what legal status does that output carry, and who is competent to trigger it?

3. Translating a statute into a specification is an interpretive judgment rather than a formally verifiable property. What institutional process, covering who drafts, who reviews, and who adopts, and under what advisory duties, gives sufficient confidence that an adopted interpretation is defensible, and how does that process relate to the review the Raad van State already performs on delegated regulation?

4. What would it mean, under the Bekendmakingswet, to publish an executable specification: in which register, with what authentic text, and with what legal consequence attaching to the published artifact as against the statutory text it encodes? Does the constitutional principle of promulgation reach the operative encoding, or does it require a category of publication alongside the Staatsblad and the Staatscourant, and what follows for a decision taken under a specification that was executed but never published?

Discretion and the override.

1. How should formalized law execution handle open norms, discretionary space, and proportionality requirements that resist deterministic encoding? Section 5.4 gives the format a mechanism for marking that boundary once it is found, annotation, refusal, human review, but not a way to locate it: which provisions are discretion the format could represent through branches or human decision points, and which resist encoding altogether, remains a legal judgment the format does not make. Can the proportionality override Scheltema (2021a) grounds in Art. 3:4(2) Awb, which he argues reaches even bound powers, function within mechanized execution, alongside or in place of the override mechanism of Section 4.6?

2. What limits the power to override? Art. 4:84 Awb supports departure from a published rule where, owing to special circumstances, strict application would be disproportionate, and Dutch practice reserves that departure for the citizen's benefit. The mechanism supplies no such limit of its own, since the same construct is used for a *lex specialis* that shortens an objection term against the citizen (Section 4.6). The limit comes from the specifications, and the Awb that states most of it is itself one of them. May an authority substitute a value to the citizen's detriment on a discretionary ground, and if not, what makes the constraint enforceable? Is the override itself a *besluit*, separately contestable, or a ground within the decision it modifies, and who within the administration may take it in the competent organ's name (*mandaat*, Arts. 10:1–10:3 Awb), given that Section 4.7 permits overrides produced by a process that never saw the file?

3. May a citizen ask for a departure, and what does a refusal leave behind? Art. 4:84 Awb is in practice invoked by the person the rule applies to. Section 4.6 counts the cases where a harsh outcome stood and no authority departed, and holds the administration to account for the mercy it withheld, but a count of absences does not separate a departure that was weighed and refused from one that nobody considered. Should a refusal be recorded with its reasons, as a departure is, and does a request for one oblige the authority to answer it?

4. Can a citizen invoke a departure granted to someone else? An organ is bound to its own consistent practice, and there is no equality in unlawfulness, so a corpus of findable overrides does not by itself create a claim. It does make a comparison possible that nobody could make before. If a departure in one case comes to bind the next, what stops an administration from departing in none?

5. With what intensity does a court review a departure, and does an attested trace that shows both the computed value and the substituted one change what it can ask?

Review and redress.

1. Can legal professionals reliably establish that an encoding means what the statute means, both by reading it article by article and by running it against cases, and under what conditions of training and tooling? The format is designed to make this check possible (Section 4.1). An encoding that runs cleanly and passes the cases tested can still diverge in meaning from the article; whether the professionals expected to perform the check would catch that is untested.

2. If a court can run the published specification against the facts of a case (Section 6.3), on what basis and with what intensity does it review the specification itself? The administrative court cannot annul a general rule at all, since Art. 8:3 Awb bars an appeal against one. Its only route is to decline to apply the rule in the case before it (*exceptieve toetsing*), and that route is open where the specification is a *beleidsregel*-like instrument. On what basis and with what intensity should it take that route, and how should the judge allocate the review across the categories of dispute encoding makes distinguishable: factual disagreements about input data, encoding disputes about whether the specification matches the statute, conflicts between correctly encoded laws that pull in opposite directions, and cases requiring application of open norms?

3. Does the recipient of a decision have a right to its execution trace? Every check this paper offers a citizen needs it (Section 4.5), and the routes that exist were designed for other things. Do the documents relating to the case (Arts. 7:4(2) and 8:42 Awb) include a trace, and does the duty to state reasons (Arts. 3:46 and 3:47(1)) require what the engine resolved where a decision was computed? Does Art. 15 GDPR reach the values one execution resolved, and what does a term of a month mean for a check the recipient needs within the objection period? Where a trace resolves data about another person, Art. 15(4) GDPR limits the copy, so the right may have to be a right to a partial view, and Section 12.2 asks the technical discipline what such a view still lets the recipient establish.

4. What may an engine do when no decision is taken? Section 4.9 defines verificative execution as continuous checking of state against what the law requires, and Section 13 concedes that it removes an enforcement margin nobody voted to withdraw. We do not know what it needs. On what basis may a state check a population against a rule outside any decision, what does Art. 8 ECHR demand of it after the SyRI judgment (Rechtbank Den Haag, 2020), and what is a flagged violation before anyone has decided anything?

5. What evidentiary weight does an execution trace with a valid attestation have in *bezwaar* and on appeal? Does it discharge the administration's duties of careful

preparation and adequate *motivering* (Arts. 3:2 and 3:46 Awb) as to the application of the rule, and does it thereby shift to the citizen the burden of showing that the fault lies in the encoding rather than in its execution? What does that shift demand of the citizen's access to counter-expertise, and how should the administrative judge treat a trace the citizen cannot independently reproduce?

6. When an attested specification is later found to misencode the statute, what follows for the decisions it produced? Section 4.2 proposes that correction operates prospectively, protecting reliance. Where the error ran to the citizen's detriment, does attestation, by identifying exactly which decisions the defective rule decided, create a duty to restore the citizen to the position the law required (*rechtsherstel*) that displaces *formele rechtskracht*, and on what basis is the state liable for a published encoding that citizens were invited to rely on?

7. What is the legal status of automated explanations of law execution, and what liability framework applies when citizens act on explanations that are incorrect or misleading? Section 8.3 holds that a generated explanation which misstates the decisive ground is a *motiveringsgebrek* regardless of whether the underlying trace is correct.

Effect across time and organizations.

1. What new legal obligations arise when temporal complexity in law becomes computationally tractable? When cross-law dependencies and retroactive amendments can be detected automatically, does this create a duty to re-evaluate past decisions, and does the tractability of cross-law interaction raise the standard that the duty of careful preparation (*zorgvuldigheidsbeginsel*, Art. 3:2 Awb) imposes on the administration?

2. What legal recognition do attested claims need to have consequences across organizations? A providing organ that discloses against a published, attested rule (Section 9.3) remains accountable for the disclosure (Art. 5(2) GDPR); what standing must the attestation, and the certification of engines and keys behind it, have to make that reliance defensible, and does that standing require the same statutory anchoring as the publication and attestation duties of Section 4.4?

3. Who holds the register of published specifications, and what prevents it from being rewritten? The reliance a citizen may place on a published interpretation (Section 4.2) presupposes that what was in force on a given date cannot later be changed. A register operated by the executive does not reproduce the monopoly on knowledge this paper diagnoses, since its contents are public. It creates a monopoly on custody instead: the power to alter the record of what was in force. The Bekendmakingswet already authenticates official publication and preserves the archive, and a *beleidsregel* is already published under it, in the Staatscourant where the authority is national and in the gazette of the municipality, province, or water board where it is not (Art. 3:42 Awb, Arts. 5 and 6 Bekendmakingswet), so the question is what that regime does not give: it does not bind the published artifact to what executed, and custody of the platform remains with the executive. Publication is thus already divided. Does each competent authority then publish the specifications it adopts, and if 300 municipalities each publish their copy of a rule that is meant to be the same, which copy is authentic and what follows when they diverge (Sections 4.3

and 11.3)? An append-only log with independent monitors would make the temporal claim of Section 9.4 cryptographic rather than institutional, and the lesson of certificate transparency is that detecting a divergent view calls for monitors rather than authorities, so no constitutional function need be reallocated to obtain it. What legal standing must such monitoring have, and what does it leave untouched, given that an append-only log settles what was published and not who may enter a specification or who decides that an entry is lawful?

12.2 Computer Science Research

Formal methods provide tools for analyzing encoded law, though which tool applies depends on what the operation set is. Linear and metric temporal logic (LTL, MTL) (Pnueli, 1977; Koymans, 1990) can specify what "when to activate" means for reactive and verificative execution modes. Program synthesis can help extract specifications from legacy code or generate specifications from scenarios. No specification aggregates over a collection today, so completeness and consistency within one are static questions over a finite, loop-free expression (Section 9.2). Their decidability depends on the arithmetic the schema admits. Bounded aggregation, which Section 9.2 announces, preserves termination and makes that static check harder, since the property must then hold for every collection the law admits. Whatever can be decided is decided before a single case is run, so a gap or a contradiction in a proposed statute is a finding the legislature can be given while it is still deliberating (Section 6.1). Model checking (Clarke, Grumberg, and Peled, 1999) applies where there is a state space to explore: the staged lifecycle of Section 4.10, and the invariant monitoring of the verificative mode (Section 4.9).

Data flow analysis can prove that one execution path accesses a subset of what another accesses, though not that a set of accesses is the minimum the law requires (Section 7.1). Section 9.1 leaves open what equivalence must hold between alternative encodings and between independent engines. Equality of outputs may not suffice, since the trace is signed and compared, and the staged lifecycle of Section 4.10 gives a specification the internal steps a bisimulation (Milner, 1989) would reach.

Formal verification of the engine is resource-intensive but perhaps necessary for the highest-stakes laws. Comprehensive testing can give confidence, but cannot prove absence of bugs (Dijkstra, 1972). Multi-implementation consistency testing (Avizienis, 1985; Knight and Leveson, 1986) can verify independent engines agree but cannot verify all are correct.

1. Which of incompleteness, inconsistency, dead code, and order-sensitivity are decidable within a single specification, and at what cost, and which of them become undecidable or intractable once resolution across interdependent, date-resolved specifications is admitted? Section 9.2 establishes termination within a specification and concedes that a legislator can build a cycle across specifications, on which the engine halts.

2. What must the dependency graph of Section 5.1 record to support those checks: authorization, delegation, retroactive and prospective effect, and cross-organizational resolution, and how should the temporal version management of Section 9.4 represent them?

3. What is the formal semantics of the specification format, and what notion of equivalence must two engines satisfy for their outputs to count in law as the same decision? Section 9.1 requires that independent engines produce identical results and

defines equivalence over outputs, while leaving the value domain, the error semantics, and the calendar semantics open. Is the numeric domain exact decimal, rational, or floating point, and what does division yield where the statute prescribes no rounding and the engine rounds nothing of its own accord (Section 9.2)? What does an engine do when an operation is not total, and is the result a value, a refusal, or a defect? A conformance relation an engine can be certified against presupposes answers to these.

4. Under what conditions is a decision reproducible years after it was taken? The version management of Section 9.4 fixes the rule side: releases tagged by publication date, references resolved to the versions in force on the relevant date, the versions actually used recorded per execution. Re-execution also needs the inputs and the knowledge state as they stood, and those it does not fix: registry data is corrected after the fact, a retroactive amendment changes in hindsight which version was in force on a past date, and different provisions read from different reference dates. What bitemporal model, separating the time a rule was valid from the time that validity was known (Snodgrass, 1999), and what discipline for fixing inputs, make "the same case, re-run" well defined, and which reference date governs each hop of a cross-law reference when the provisions along the path read from different dates?

5. How can execution path analysis minimize privacy impact, and can data minimisation requirements (GDPR Art. 5(1)(c)) be operationalized as computable policy? Sensitivity is a partial order (Section 7.1), so any total order over it is a normative choice. Who is competent to fix that order, in what instrument must it be published, and what could then be proved about an engine that followed it?

6. What infrastructure is required for authorization derived from law execution, including certified engines, cryptographic attestation, and key management across organizational boundaries? If executing the law is itself the authorization to access data, while the legal basis stays with the provision the rule cites (Section 9.3), what technical and legal framework makes this work?

7. What must a verifier be able to obtain to run the check, and who supplies it? Re-execution needs the specifications behind the signed digests, and an engine the verifier can obtain that is certified equivalent to the one that decided (Section 9.1). A register that resolves a digest to content, and an engine a lay verifier or the tool acting for them can actually run, are conditions of the citizen's check (Section 4.4) and not details of it.

8. Is the specification format closed under execution? A decision is *technically* a grounded instance of a specification: the outputs of Figure 1, fixed to values for one case. The logic-programming lineage of Section 2.3 suggests that representing it as an instance of the same published schema is coherent, since a fact is a rule without conditions. But a decision contains what a rule does not: input values fixed at a decision date, the attestation of Section 4.4 over the whole resolved closure, the override events of Section 4.6. Closure fails if hosting these requires extensions the restricted operation set of Section 9.2 was designed to exclude, or if attestation and overrides change meaning when they attach to instances rather than to rules. If it holds, decisions can be validated, diffed, and queried with the tooling that exists for rules, a decision-reference (Section 4.12) needs no separate interchange format, and

a cross-cutting law such as the Awb, whose subject matter is decisions, consumes the decisions of other laws as ordinary inputs (Section 4.9).

9. Which obligations of the procedural lifecycle can a certified engine executing a published specification discharge, and which necessarily remain with a case file outside it? Section 4.10 makes the stage a case is in, and the rules that fire on it, properties of the published rule, while Section 6.2 still assumes case-management systems that consume the specification, and Section 4.10 concedes that whether a hearing was adequate the trace cannot show. If a certified engine executing the published lifecycle can carry a case from application to notification to objection, what remains for the surrounding systems, storing case facts and documents, communication with the parties, human tasks such as hearings, and where does the line fall for the rest of the Awb's procedural duties?

10. How can a conformance suite, a published set of canonical cases (inputs paired with the outcomes a law intends) that any execution must reproduce, be offered alongside a statute and kept in step with it as the law is amended or its amounts are indexed (Section 6.1)? What artifact, versioned with the law the way Section 9.4 versions the specification, would track each change, and what coverage criterion makes such a suite adequate for a statute, the analogue over articles, conditions, and thresholds of the coverage criteria used for safety-critical code? What authority would maintain and publish it?

11. How is a machine-generated encoding validated, and at what human cost? Section 5.3 assigns translation of the corpus to language models and makes human adoption the act that confers legal effect, while conceding that the fidelity, reverse-trace, and drift checks cannot catch an encoding that executes without error yet means something the article does not. What is the measured error profile of such a pipeline against encodings produced by legal experts, which classes of error do the published checks catch, and what review effort per article does defensible adoption require? The claim that translation scales with computing capacity rather than staffing rests on that last figure.

12. What establishes that unpublished selection and enforcement logic does not influence a published decision path? Section 6.5 keeps risk models and selection logic outside the publication duty, which leaves the non-interference of the two layers to be shown rather than assumed. Information flow security (Sabelfeld and Myers, 2003) offers the property; what would a proof of it cover, who could check it without seeing the withheld logic, and what would such a proof be worth in court against the failure mode the SyRI judgment identified?

13. How do a structured notation and a controlled natural language compare empirically on the properties this paper rests on: a diff a court can read, equivalence across independent engines, and accessibility to readers who are not engineers? Section 10.1 defends the choice of a structured notation on argument, not on measurement; this comparison is the empirical test of that design choice. The authoring layer the same section sketches raises its own question: under what conditions does a controlled-natural-language surface that compiles to the published specification preserve the constitutional properties the published form is designed to guarantee,

without reintroducing at the authoring stage the ambiguity the structured form removes?

14. What must a decision attest to, and to whom can it prove it? A hash of a published specification fixes which rule the organization declares it ran; the outcome is checkable only by re-executing that rule, which needs the inputs and the resolved versions of every rule the specification references. What binds an input to the source authoritative for it, so that the value a decision runs on is the value the register holds, and what would source-signed inputs mean for the receiving organ's duty of careful preparation (Art. 3:2 Awb) and for the *terugmeldplicht* (Section 4.12)?

12.3 Political Science and Governance Research

The distribution of winners and losers requires careful analysis, with resistance patterns predictable: IT vendors lose lock-in, supervisory bodies lose negotiating power, and organizational identities are threatened.

1. What institutional design enables adoption of executable law across government, given that transparency redistributes power away from those who currently control it? Which actors gain (citizens, courts, auditors), how should transition support those whose roles change, and how does the *poldermodel* (Dutch consensus-based governance culture)'s dependence on productive ambiguity, the politics of accommodation Lijphart (1968) described, interact with formalization?

2. How should the authority to produce, publish, and coordinate machine-executable interpretations of law be allocated across levels of government, and when should diversity of interpretation be tolerated versus mandated to converge? When does equality before the law require uniform encoding, and when does municipal autonomy allow local variation?

3. How does the ability to simulate, test, and verify law before enactment change the nature of legislative deliberation, and what does this mean for the relationship between Parliament and the executive?

4. Which features of Dutch administrative law does this proposal presuppose: a codified general administrative-procedure statute, a published *beleidsregel* category, a duty of *motivering*, and a promulgation (*bekendmaking*) regime? Which of these doctrines have functional analogues in EU law, in other continental systems, and in common law jurisdictions, and what does the proposal reduce to where they are absent?

12.4 Philosophy of Law and Technology

The interpretive problem is unavoidable: an executable specification embeds choices about what counts as relevant, how ambiguities are resolved, and what relationships between rules matter. Different people might encode the same law differently, each interpretation defensible. Can you eliminate interpretation by formalizing it, or does formalization merely hide interpretive choices behind technical notation? The refusal mechanism of Section 5.4 is one concrete design response at this boundary: where the format cannot carry an article's meaning, the gap is published and execution halts; that answers the practical question, not the philosophical one.

Van den Hoven (2017) argues that no technology is value-neutral: every system embeds values whether designers intend it or not. The Value Sensitive Design tradition (Friedman and Hendry, 2019) and the design-for-values school around Van den Hoven both translate abstract values into norms and then into concrete design requirements. The constitutional values this paper identifies (knowability, legal certainty, separation of powers) map onto this hierarchy: they are values that must be translated into design requirements for law execution systems. The restricted format, mandatory publication, and cryptographic attestation are such requirements. In this reading, the proposal is an instance of what Van den Hoven calls "moral specs": explicit specifications of the values a system must embody, treated with the same rigor as functional specifications.

Law operates across time with statutes changing and amendments having retroactive or prospective effects, while formal systems typically operate synchronically. What does it mean to execute law correctly when the law itself is changing?

Applying a rule to a case calls for judgment about the situation in front of the official, and formalization may make law clearer while making it more brittle and less able to adapt to circumstances no one anticipated. The reverse is also possible: the proportionality override Scheltema grounds in Art. 3:4(2) Awb works only where the rule producing the disproportionate outcome is visible and identifiable, and formalization makes it so. What is lost when law becomes executable, and does formalization preserve, or even enhance, the law's capacity to respond justly to human complexity?

1. Under what conditions does publishing an encoding open a reading of the law to contestation rather than entrench it? Section 4.3 answers Hildebrandt by holding that the interpretive act is not new and that publication makes it visible. Does the override of Section 4.6 keep the encoding from becoming the last word, or does it relocate the finality to whoever decides when to depart from the rule?

2. What theory of democratic legitimacy applies to machine-executable interpretations of law, and how can executable specifications be integrated into democratic structures without reducing law to technocratic rule-application?

3. Section 6.4 places the realistic readership of a published specification with intermediaries rather than with citizens themselves. Does publication then redistribute capability, or concentrate it in whoever supplies the tools (Section 8.2)? What conditions on the availability, independence, and funding of intermediation would be required for publication to improve the position of the citizens with the least *doenvermogen* rather than the most?

13 Publication Does Not Restore Everything

An official may act only within the powers the law confers. Discretion is enumerated: the room a provision leaves where it does not bind, the weighing that Art. 3:4(2) Awb requires where it does, the duty to depart from a *beleidsregel* whose strict application would be disproportionate in a particular case (Art. 4:84 Awb), and the hardship clauses of particular statutes. Outside those powers, an official who does something good is acting without authority, and what they have done is unlawful however good it was.

In an administration run by people, that boundary is neither sharp nor policed. Something helpful happens for which no one holds a power, no one checks, and the case comes out better than the rule would have made it. A society may depend on that margin

more than it can say, and it cannot say so, because to state the margin is to authorize it, and an authorized margin is a power like any other.

Digital execution closes the margin. A system has no unwritten margin: the room to act contracts to exactly the discretion that was written down. Publication plays no part in this. The margin would close as strictly under a specification that was never published, and it is closing today, inside the software the executive already runs.

The margin cannot be legislated back. An exception that must be reasoned for is a different object from an exception no one noticed, and the second does not survive being written down. Once enacted, it is the first. A state that declines to look is operating a policy of non-enforcement, which is again a rule, published and uniform. What was lost depended on being unwritten, and nothing a legislature enacts is unwritten.

Enforcement has a margin of its own. A rule is applied as often as the administration is able to apply it, and it has never been able to apply it to everyone. The check that was never run and the recovery that was never pursued were the residue of a state that could not look everywhere at once, and that residue fell to the people whose circumstances change fastest and whose paperwork is worst, which is to say the people least able to comply. Nobody voted to give it to them, and nobody voted to take it away. Verificative execution (Section 4.9) takes it away. That capability produced Robodebt (Royal Commission into the Robodebt Scheme, 2023), where the reach was new and the way the debts were established was unlawful from the start. This paper raises whether a legislature would authorize that reach, and cannot settle it.

This paper belongs to the movement that closes the margin, and it should not pretend otherwise. Its claim is narrower. The margin is already closed, and it was closed by the software the executive runs today: systems that admit no discretion of any kind, and that closed it without anyone outside being able to see that they had. Measured against that, the proposal here reopens part of the space and makes the closure visible: the computed outcome is only a default (Section 4.6), and departing from it is a lawful act with a stated ground against an identifiable rule, which is the condition Scheltema's proportionality duty requires (Section 12.1).

This is also where a familiar argument goes wrong. Lawyers reading a rigid system see a choice, and they hold the people who built it responsible for it. The engineers hear an accusation they cannot answer, since a system that executes rules can do only what was specified, and no amount of better building produces one that does what nobody wrote down. Both are partly right, and the argument is useless because the two halves are never separated. Part of the rigidity is formalization itself. Part of it was avoidable and was not avoided: recovery in full or not at all, no way to depart from the computed outcome, no way to record that anyone had, thresholds with no hardship clause behind them. Those were decisions. This paper removes as much of the avoidable rigidity as it can, and by removing it, shows how much is left. No builder can take the residue away, and it becomes possible to say which is which. That divides the responsibility: it obliges those who build to stop invoking inevitability, and those who judge to stop asking of a rule system what no rule system can give.

An apparatus that is legible and auditable is a more attractive instrument of government than one that is neither, and instruments that work are given more work. Making the execution of rules inspectable may therefore extend the reach of rule by execution. The alternative on offer is the same apparatus, unread. Keeping it that way gives the official no room back, since the margin closed when the rules were formalized; it withholds only the citizen's one remaining advantage, a rule that can be seen and argued with.

14 Conclusion

This paper has argued that machine-executable law addresses a root constitutional problem: the executive's monopoly on understanding operative law. Digital execution adds a fourth demand to the three the doctrine already makes (knowable, comprehensible, predictable): the rules as executed must be independently verifiable. Publication in a restricted format, designed to terminate and to be readable by legal professionals, puts the interpretive choices of execution before Parliament, courts, and citizens. Because the executive must decide cases with the specification it publishes, anyone a decision reaches can re-run the published rule on the inputs the decision records and see whether the outcome is the one that rule yields. Determinism makes the outcome checkable. The inputs are recorded and signed, so they cannot be substituted after the fact. The check cannot establish that they are true, and each input must therefore be bound to the source authoritative for it, where one exists (Section 12.2). The computed outcome can still be departed from where the law confers the power, most consequentially where applying the rule would be disproportionate in the individual case, and the departure is itself attested, so publication binds execution without eliminating the individual judgment the rule of law also requires. Earlier approaches to machine-readable law made rules executable or made them public, but none guaranteed that the published rule is the rule that runs (Table 1).

The parliamentary inquiry prompted by the *toeslagenaffaire* found that all three branches had been *blind voor mens en recht* (Parlementaire enquêtecommissie Fraudebeleid en Dienstverlening, 2024). Opacity of execution was one strand of that failure: the legislature could not see its enactments in operative form, the judiciary reviewed decisions whose operative logic it could not inspect, and the executive often lacked a consolidated view of its own systems. A published, attested specification is one artifact available to all three branches and to the citizen, and each holds it for the purpose its office has. Every decision has a recipient, and every recipient can re-run their own case, as can the court seized of it. Verification thus falls to the party with standing, which is the right model: a state in which anyone could re-run anyone's case would have bought transparency by abolishing privacy. Parliament's object was never the individual decision. Art. 68 Grondwet obliges ministers to give individual members the information they ask for, unless the interest of the state opposes it, and the *toeslagenaffaire* shows what follows when members must use that right to work individual cases because nothing at the level of the rule can be inspected. Publication gives them the rule as a total function, which anyone holding no case data at all can analyze: every threshold it contains, and the full effect of an amendment before it is passed. That capacity meets the standard Section 4.3 took from Diver (2021): code that regulates is legitimate only where those subject to it can know it in advance and contest it, both individually and through the institutions that review it. Publication before application supplies the advance knowledge; the attestation on every decision gives that contest a concrete object.

Publication rebalances the powers only if the other branches can work with what is published; that requires organized readership in Parliament's support services (Section 11.5), and the intermediaries who assist citizens extend it (Section 6.4). The wider transformation required is institutional, political, and philosophical. Executing organizations give up the exclusive hold on interpretation that has shielded their choices from scrutiny, and they need new expertise and roles to work in the open. Constitutional clarity about what the trias politica (separation of powers) means in the digital context has yet to be established. The deepest questions (interpretation, authority, temporality, judgment) demand serious

engagement; the research agenda collects them, from the legal status of an executable specification to the normative foundations of formalized execution.

Incremental transparency efforts leave the core problem unaddressed. The approach proposed here is imperfect and introduces new problems, but it targets the structural cause: opacity of law execution. The rule of law depends on law being knowable and verifiable by those subject to it. Publishing the rules as executed restores the possibility that citizens and their institutions can understand and challenge how power is exercised over them.

Acknowledgements

We thank Digilab for organizing the Fieldlab that gave rise to this work and for the close collaboration since, the academics, policymakers, administrators, and other experts who tested and sharpened our thinking, and the colleagues now working with us to take these ideas further. We are also grateful to the Ministry of the Interior and Kingdom Relations for the room to pursue this line of work.

References

Afdeling bestuursrechtspraak van de Raad van State (May 17, 2017). *AERIUS/PAS referral*. ECLI:NL:RVS:2017:1259. URL: https://uitspraken.rechtspraak.nl/details?id=ECLI:NL:RVS:2017:1259.

— (May 29, 2019). *Amsterdamse dakopbouw*. ECLI:NL:RVS:2019:1694. URL: https://uitspraken.rechtspraak.nl/details?id=ECLI:NL:RVS:2019:1694.

— (Feb. 2, 2022). *Harderwijk (evenredigheidsbeginsel)*. ECLI:NL:RVS:2022:285, Grote Kamer. URL: https://uitspraken.rechtspraak.nl/details?id=ECLI:NL:RVS:2022:285.

Assemblée nationale (2020). *LexImpact*. URL: https://leximpact.an.fr (visited on 07/08/2026).

Ausems, Anouschka, John Bulles, and Mariette Lokin (2021). *Wetsanalyse: voor een werkbare uitvoering van wetgeving met ICT*. Den Haag: Boom juridisch.

Avizienis, Algirdas (1985). "The N-Version Approach to Fault-Tolerant Software". In: *IEEE Transactions on Software Engineering* SE-11.12, pp. 1491–1501. DOI: 10.1109/TSE.1985.231893.

Bench-Capon, Trevor J.M. and F. P. Coenen (1992). "Isomorphism and Legal Knowledge Based Systems". In: *Artificial Intelligence and Law* 1.1, pp. 65–86. DOI: 10.1007/BF00118479.

Boer, Alexander, Rinke Hoekstra, and Radboud Winkels (2002). "METALex: Legislation in XML". In: *Proceedings of the 15th Annual Conference on Legal Knowledge and Information Systems (JURIX 2002)*. IOS Press, pp. 1–10.

Bos, Flora Magdalena Johanna (July 2026). "From Rule Execution to Citizen Explanation: Evaluating LLM-Generated Explanations in Rule-Based Government Systems". Master's thesis. Vrije Universiteit Amsterdam.

Bovens, Mark and Stavros Zouridis (2002). "From Street-Level to System-Level Bureaucracies: How Information and Communication Technology Is Transforming Administrative Discretion and Constitutional Control". In: *Public Administration Review* 62.2, pp. 174–184. DOI: 10.1111/0033-3352.00168.

Citron, Danielle Keats (2008). "Technological Due Process". In: *Washington University Law Review* 85, pp. 1249–1313. URL: https://openscholarship.wustl.edu/law_lawreview/vol85/iss6/2/.

Clarke, Edmund M., Orna Grumberg, and Doron A. Peled (1999). *Model Checking*. Cambridge, MA: MIT Press. ISBN: 978-0-262-03270-4.

Coupette, Corinna et al. (2023). "Law Smells: Defining and Detecting Problematic Patterns in Legal Drafting". In: *Artificial Intelligence and Law* 31.2, pp. 335–368. DOI: 10.1007/s10506-022-09315-w.

Court of Justice of the European Union (Dec. 7, 2023). *OQ v Land Hessen (SCHUFA Holding (Scoring))*. Case C-634/21, ECLI:EU:C:2023:957.

Dijkstra, Edsger W. (1972). "Notes on Structured Programming". In: *Structured Programming*. London: Academic Press, pp. 1–82. ISBN: 978-0-12-200550-3.

Diver, Laurence E. (2021). *Digisprudence: Code as Law Rebooted*. Future Law. Edinburgh University Press. DOI: 10.1515/9781474485340. URL: https://laurencediver.net/assets/digisprudence.pdf.

European Court of Human Rights (Dec. 4, 2008). *S. and Marper v. the United Kingdom*. App. nos. 30562/04 and 30566/04, Grand Chamber. URL: https://hudoc.echr.coe.int/fre?i=001-90051.

European Parliament and Council of the European Union (June 13, 2024). *Regulation (EU) 2024/1689 laying down harmonised rules on artificial intelligence (Artificial Intelligence Act)*. OJ L, 2024/1689, 12.7.2024.

Friedman, Batya and David G. Hendry (2019). *Value Sensitive Design: Shaping Technology with Moral Imagination*. Cambridge, MA: MIT Press. ISBN: 978-0-262-03953-6. DOI: 10.7551/mitpress/7585.001.0001.

Fuller, Lon L. (1969). *The Morality of Law*. rev. New Haven: Yale University Press. ISBN: 978-0-300-01070-1.

Gerards, Janneke H. et al. (2021). *Impact Assessment Mensenrechten en Algoritmes*. Utrecht University. URL: https://dspace.library.uu.nl/handle/1874/415481.

Goldberg, David (1991). "What Every Computer Scientist Should Know About Floating-Point Arithmetic". In: *ACM Computing Surveys* 23.1, pp. 5–48. DOI: 10.1145/103162.103163.

Hart, H. L. A. (1961). *The Concept of Law*. Oxford: Clarendon Press.

Hildebrandt, Mireille (2020). "Code-Driven Law: Freezing the Future and Scaling the Past". In: *Is Law Computable? Critical Perspectives on Law and Artificial Intelligence*. Ed. by Christopher Markou and Simon Deakin. Hart, pp. 67–83. DOI: 10.5040/9781509937097.ch-003.

Informatiepunt Leefomgeving (2026). *Strategie voor toepasbare regels bepalen*. URL: https://iplo.nl/digitaal-stelsel/toepasbare-regels-maken-aanleveren/strategie-bepalen/ (visited on 07/10/2026).

Kaplow, Louis (1992). "Rules versus Standards: An Economic Analysis". In: *Duke Law Journal* 42.3, pp. 557–629. DOI: 10.2307/1372840.

Knight, John C. and Nancy G. Leveson (1986). "An Experimental Evaluation of the Assumption of Independence in Multiversion Programming". In: *IEEE Transactions on Software Engineering* SE-12.1, pp. 96–109. DOI: 10.1109/TSE.1986.6312924.

Koymans, Ron (1990). "Specifying Real-Time Properties with Metric Temporal Logic". In: *Real-Time Systems* 2.4, pp. 255–299. DOI: 10.1007/BF01995674.

Lijphart, Arend (1968). *The Politics of Accommodation: Pluralism and Democracy in the Netherlands*. Berkeley: University of California Press.

Lokin, Mariette (2018). "Wendbaar wetgeven: de wetgever als systeembeheerder". PhD thesis. Vrije Universiteit Amsterdam.

— (June 12, 2026). *Digitaal Disciplineren Anno 2026: Hoe Houdt de Wetgever Grip Op Digitale Uitvoering?* Inaugural lecture. Oratie, leerstoel Wetgeven in de digitale rechtsstaat. Heerlen.

Lokin, Mariette and Reijer Passchier (Nov. 2024). *Rechtsstatelijke risico's van de digitale uitvoering van wetten*. Wetenschappelijke factsheet. Parlement en Wetenschap. URL: https://www.eerstekamer.nl/bijlage/20250129/wetenschappelijke_factsheet/document3/f=/vmkgn0uje7le.pdf.

McCarty, L. Thorne (1977). "Reflections on TAXMAN: An Experiment in Artificial Intelligence and Legal Reasoning". In: *Harvard Law Review* 90, pp. 837–893. DOI: 10.2307/1340132.

Mérigoux, Denis, Nicolas Chataing, and Jonathan Protzenko (2021). "Catala: A Programming Language for the Law". In: *Proceedings of the ACM on Programming Languages* 5.ICFP, 77:1–77:29. DOI: 10.1145/3473582.

Meuwese, Anne and Ivar Timmer (2022). "'Law Smells': Over Digitale Vertalingen van Regelgeving en de Uitvoeringspraktijk". In: *RegelMaat* 38.3, pp. 272–283. DOI: 10.5553/RM/0920055X2022038003007.

Milner, Robin (1989). *Communication and Concurrency*. London: Prentice Hall. ISBN: 978-0-13-114984-7.

Ministerie van Binnenlandse Zaken en Koninkrijksrelaties (2022). *Algoritmeregister van de Nederlandse Overheid*. URL: https://algoritmes.overheid.nl/ (visited on 03/06/2026).

Ministerie van Binnenlandse Zaken en Koninkrijksrelaties and Ministerie van Justitie en Veiligheid (2024). *Wet versterking waarborgfunctie Awb: voorontwerp ter internetconsultatie*. URL: https://www.internetconsultatie.nl/waarborgfunctieawb (visited on 07/08/2026).

Ministerie van Volkshuisvesting en Ruimtelijke Ordening (Feb. 3, 2025). *Voortgang implementatie Omgevingswet vierde kwartaal 2024*. Kamerstuk 33 118, nr. 287. Tweede Kamer der Staten-Generaal. URL: https://zoek.officielebekendmakingen.nl/kst-33118-287.html.

Mohun, James and Alex Roberts (Oct. 14, 2020). *Cracking the Code: Rulemaking for Humans and Machines*. OECD Working Papers on Public Governance 42. OECD. DOI: 10.1787/3afe6ba5-en. URL: https://oecd-opsi.org/publications/cracking-the-code/.

Nationale Ombudsman (2021). *Een Burger Is Geen Dataset: Ombudsvisie Op Behoorlijk Gebruik van Data en Algoritmen Door de Overheid*. 2021/021. URL: https://www.nationaleombudsman.nl/publicaties/onderzoeken/een-burger-is-geen-dataset.

Nielson, Flemming, Hanne Riis Nielson, and Chris Hankin (1999). *Principles of Program Analysis*. Berlin: Springer. ISBN: 978-3-540-65410-0. DOI: 10.1007/978-3-662-03811-6.

North, Dan (Mar. 2006). *Introducing BDD*. URL: https://dannorth.net/blog/introducing-bdd/ (visited on 03/06/2026).

Olthof, Timen and Marc Van Andel (2025). *Chronolexografie: Het Bijhouden van de Rechtstoestand in de Tijd*. URL: https://chronolexografie.nl/ (visited on 03/07/2026).

Open Source Security Foundation (2023). *SLSA Specification v1.0*. URL: https://slsa.dev/spec/v1.0/ (visited on 07/10/2026).

Palmirani, Monica and Fabio Vitali (2011). "Akoma-Ntoso for Legal Documents". In: *Legislative XML for the Semantic Web*. Ed. by Giovanni Sartor et al. Dordrecht: Springer. DOI: 10.1007/978-94-007-1887-6_6.

Parlementaire enquêtecommissie Fraudebeleid en Dienstverlening (Feb. 26, 2024). *Blind voor mens en recht*. Tweede Kamer der Staten-Generaal. URL: https://www.tweedekamer.nl/kamerstukken/detail?id=2024Z02942&did=2024D06757.

Pasquale, Frank (2015). *The Black Box Society: The Secret Algorithms That Control Money and Information*. Cambridge, MA: Harvard University Press. ISBN: 978-0-674-36827-9. DOI: 10.4159/harvard.9780674736061.

Pasquale, Frank and Gianclaudio Malgieri (2024). "Generative AI, Explainability, and Score-Based Natural Language Processing in Benefits Administration". In: *Journal of Cross-disciplinary Research in Computational Law* 2.2. URL: https://journalcrcl.org/crcl/article/view/59.

Passchier, Reijer (2020). "Digitalisering En de (Dis)Balans Binnen de Trias Politica". In: *Ars Aequi* 69.10, pp. 916–927.

Peeters, Rik and Arjan C. Widlak (July 2023). "Administrative Exclusion in the Infrastructure-level Bureaucracy: The Case of the Dutch Daycare Benefit Scandal". In: *Public Administration Review* 83.4, pp. 863–877. ISSN: 0033-3352, 1540-6210. DOI: 10.1111/puar.13615. URL: https://onlinelibrary.wiley.com/doi/10.1111/puar.13615 (visited on 03/04/2026).

Pnueli, Amir (1977). "The Temporal Logic of Programs". In: *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, pp. 46–57. DOI: 10.1109/SFCS.1977.32.

Prins, Corien (2016). "Digitale (Dis)Balans Binnen de TRIAS". In: *Nederlands Juristenblad* 91.14, p. 909.

Raad van State (Aug. 31, 2018). *Ongevraagd advies over de effecten van de digitalisering voor de rechtsstatelijke verhoudingen*. W04.18.0230/I. URL: https://www.raadvanstate.nl/@112661/w04-18-0230/.

Raz, Joseph (1979). *The Authority of Law: Essays on Law and Morality*. Oxford: Oxford University Press. ISBN: 978-0-19-825345-7. DOI: 10.1093/acprof:oso/9780198253457.001.0001.

Rechtbank Den Haag (Feb. 5, 2020). *NJCM c.s. v. the State of the Netherlands (SyRI)*. ECLI:NL:RBDHA:2020:865. URL: https://uitspraken.rechtspraak.nl/details?id=ECLI:NL:RBDHA:2020:865.

Royal Commission into the Robodebt Scheme (July 2023). *Report of the Royal Commission into the Robodebt Scheme*. Commonwealth of Australia. URL: https://robodebt.royalcommission.gov.au/publications/report.

Sabelfeld, Andrei and Andrew C. Myers (2003). "Language-Based Information-Flow Security". In: *IEEE Journal on Selected Areas in Communications* 21.1, pp. 5–19. DOI: 10.1109/JSAC.2002.806121.

Scheltema, Michiel (2015). "Bureaucratische rechtsstaat of responsieve rechtsstaat?" In: *Nederlands Tijdschrift voor Bestuursrecht*, pp. 287–289.

— (2018). "Wetgeving in de responsieve rechtsstaat". In: *RegelMaat* 33.3, pp. 121–132. DOI: 10.5553/RM/0920055X2018033003002.

— (2021a). "Een wet van Meeden en Perzen? Geen onwrikbare wet in het hedendaags bestuursrecht". In: *Wetgeving en uitvoering. Preadviezen van de Nederlandse Vereniging voor Wetgeving*. Oisterwijk: Wolf Legal Publishers, pp. 15–57. ISBN: 978-94-6240-689-6.

Scheltema, Michiel (2021b). "Mag de rechtsstaat voor de burger worden gesloten? Bouwstenen voor een meer burgervriendelijk bestuursrecht". In: *Ars Aequi* 70.9, pp. 809–820.

Sergot, M. J. et al. (1986). "The British Nationality Act as a Logic Program". In: *Communications of the ACM* 29.5, pp. 370–386. DOI: 10.1145/5689.5920.

Service Innovation Lab (2018). *Better Rules for Government Discovery Report*. New Zealand Government, Department of Internal Affairs. URL: https://www.digital.govt.nz/dmsdocument/95-better-rules-for-government-discovery-report/html.

Snodgrass, Richard T. (1999). *Developing Time-Oriented Database Applications in SQL*. San Francisco: Morgan Kaufmann. ISBN: 1-55860-436-7.

Tekofsky, Aliza and Roos de Groot (2024). *Whitepaper Fieldlab Proactieve Dienstverlening*. Digilab. URL: https://digilab.overheid.nl/projecten/fieldlab-proactieve-dienstverlening/whitepaper/.

Valente, André and Joost Breuker (1994). "A Functional Ontology of Law". In: *Towards a Global Expert System in Law*. Padova: CEDAM.

Van Binsbergen, L. Thomas et al. (2020). "eFLINT: A Domain-Specific Language for Executable Norm Specifications". In: *Proceedings of the 19th ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences (GPCE '20)*. ACM. DOI: 10.1145/3425898.3426958.

Van Doesburg, Robert and Tom M. van Engers (2019). "Explicit Interpretation of the Dutch Aliens Act: Specifications for Decision Support Systems and Administrative Practice". In: *Proceedings of the 1st Workshop on Artificial Intelligence and the Administrative State (AIAS 2019)*. Vol. 2471. CEUR Workshop Proceedings. Montreal, pp. 27–37. URL: https://ceur-ws.org/Vol-2471/paper6.pdf.

Van Doesburg, Robert, Tijs van der Storm, and Tom M. van Engers (2016). "CALCULEMUS: Towards a Formal Language for the Interpretation of Normative Systems". In: *AI4J Workshop at ECAI 2016*. The Hague.

Van Eck, Marlies (2018). "Geautomatiseerde ketenbesluiten & rechtsbescherming: een onderzoek naar de praktijk van geautomatiseerde ketenbesluiten over een financieel belang in relatie tot rechtsbescherming". PhD thesis. Tilburg University. DOI: 10.2139/ssrn.3422781.

Van Engers, Tom M. et al. (2001). "POWER: Using UML/OCL for Modeling Legislation — an Application Report". In: *Proceedings of the 8th International Conference on Artificial Intelligence and Law (ICAIL 2001)*. St. Louis. DOI: 10.1145/383535.383554.

Van Kralingen, Robert W. (1997). "A Conceptual Frame-Based Ontology for the Law". In: *Proceedings of the First International Workshop on Legal Ontologies*. Ed. by Pepijn R.S. Visser and Radboud G.F. Winkels. Melbourne, pp. 15–22.

Van Amerongen, N.H. and Y.E. Schuurmans (2019). "Advies van een Deskundige of Algoritme? De Toetsing van 'Black Box'-besluiten Door de Bestuursrechter". In: *Verwant met Verband: Ruimte, Recht en Wetenschap*. Ed. by P.J. Huisman, A.R. Neerhof, and F.J. Van Ommeren. Den Haag: IBR, pp. 175–196. ISBN: 978-94-6315-045-3.

Van den Hoven, Jeroen (2017). "Ethics for the Digital Age: Where Are the Moral Specs?" In: *Informatics in the Future*. Ed. by Hannes Werthner and Frank Van Harmelen. Cham: Springer. DOI: 10.1007/978-3-319-55735-9_6.

Van Eck, Marlies et al. (2022). *De LegitiMaat: Een Werkmethode Om de Geautomatiseerde Uitvoering van Wetten Te Beoordelen*. Ministerie van Binnenlandse Zaken en Koninkrijksrelaties. URL: https://www.paoleiden.nl/leiden-law-academy-blog/beric

hten/2022/juli/de-legitimaat-een-werkmethode-om-de-geautomatiseerde-uitvoering-van-wetten-te-beoordelen/.

Visser, Pepijn R.S. and Trevor J.M. Bench-Capon (1998). "A Comparison of Four Ontologies for the Design of Legal Knowledge Systems". In: *Artificial Intelligence and Law* 6, pp. 27–57. DOI: 10.1023/A:1008251913710.

Voermans, Wim et al. (2011). *Juridische betekenis en reikwijdte van het begrip 'rechtsstaat' in de legisprudentie & jurisprudentie van de Raad van State*. Studies en rapporten / Raad van State 4. Den Haag: Raad van State. 141 pp. ISBN: 978-94-90830-02-1.

Waldron, Jeremy (2011). "The Rule of Law and the Importance of Procedure". In: *Getting to the Rule of Law: NOMOS L*. Ed. by James E. Fleming. New York: New York University Press, pp. 3–31. DOI: 10.18574/nyu/9780814728437.003.0001.

Wetenschappelijke Raad voor het Regeringsbeleid (2011). *iGovernment*. WRR Reports to the Government 86. Amsterdam University Press. ISBN: 978-90-8964-394-0. DOI: 10.26530/OAPEN_401759.

— (Apr. 24, 2017). *Weten is nog geen doen: Een realistisch perspectief op redzaamheid*. 97. Den Haag: WRR. URL: https://www.wrr.nl/publicaties/rapporten/2017/04/24/weten-is-nog-geen-doen.

Widlak, Arjan C. and Rik Peeters (June 2025). "A Theory of the Infrastructure-Level Bureaucracy: Understanding the Consequences of Data-Exchange for Procedural Justice, Organizational Decision-Making, and Data Itself". In: *Government Information Quarterly* 42.2, p. 102021. ISSN: 0740624X. DOI: 10.1016/j.giq.2025.102021. URL: https://linkinghub.elsevier.com/retrieve/pii/S0740624X25000152 (visited on 03/04/2026).

Williams, Wyn (July 8, 2025). *Post Office Horizon IT Inquiry: Volume 1 – Human Impact and Redress*. HC 1119. Post Office Horizon IT Inquiry. URL: https://www.postofficehorizoninquiry.org.uk/reports-and-statements.

Winskel, Glynn (1993). *The Formal Semantics of Programming Languages: An Introduction*. Cambridge, MA: MIT Press. ISBN: 978-0-262-73103-4. DOI: 10.7551/mitpress/3054.001.0001.

Zouridis, Stavros, Marlies Van Eck, and Mark Bovens (2019). "Automated Discretion". In: *Discretion and the Quest for Controlled Freedom*. Ed. by Tony Evans and Peter Hupe. Springer, pp. 313–329. DOI: 10.1007/978-3-030-19566-3_20.

Zuurmond, Suzan et al. (2023). "Human-Centred Explanation of Rule-Based Decision-Making Systems in the Legal Domain: Demonstration". In: *Legal Knowledge and Information Systems: JURIX 2023*. Vol. 379. Frontiers in Artificial Intelligence and Applications. IOS Press, pp. 375–378. DOI: 10.3233/FAIA230992.