En Dimuro, Juan Jose, *Disruptive Syntax: Authority Without Subject in Artificial Language*. Nassau (Bahamas): LeFortune.

# Compiled Norms: Towards a Formal Typology of Executable Legal Speech.

Agustin V. Startari.

# Compiled Norms: Towards a Formal Typology of Executable Legal Speech

**Author:** Agustin V. Startari

**ResearcherID:** NGR-2476-2025

**ORCID:** 0009-0001-4714-6539

**Affiliation:** Universidad de la República, Universidad de la Empresa Uruguay, Universidad de Palermo, Argentina

**Email:** astart@palermo.edu, agustin.startari@gmail.com

**Date:** July16, 2025

**DOI:** https://doi.org/10.5281/zenodo.15881325

This work is also published with DOI reference in **Figshare** https://doi.org/10.6084/m9.figshare.29562293 and **Pending SSRN ID to be assigned. ETA: Q3 2025.**

**Language:** English

**Serie:** Grammars of Power

**Directly Connected Works (SSRN):**

*AI, Tell Me Your Protocol (2025), caps. 2.2, 5.1, 5.4. DOI:* https://doi.org/10.2139/ssrn.5260083

*Executable Power: Syntax as Infrastructure in Predictive Societies — DOI:* https://doi.org/*10.5281/zenodo.15754714*

*Algorithmic Obedience — DOI:* https://doi.org/ 10.2139/ssrn.5282045

*Ethos Without Source — DOI:* https://doi.org/10.2139/ssrn.5260083

**Word count:** 4307

**Keywords:** syntactic delegation, hedge suppression, diagnostic language models, SaMD, clinical authority, responsibility leakage, regulatory asymmetry, linguistic risk, compiled rule, impersonal syntax, medical LLMs, legal-medical overlap, uncertainty erasure.

**Abstract**

This article introduces a formal typology of executable legal speech. Building on the concept of the *regla compilada* (compiled rule), it identifies the syntactic conditions under which a legal expression becomes executable by non-human systems. The analysis distinguishes declaration from compiled legal language and proposes four structural criteria for computability: position within the Chomsky grammar hierarchy, closure of rule structure, level of semantic ambiguity, and determinism of parsing. Instead of interpreting legal meaning, the article isolates the formal properties that permit legal norms to function as executable code. The objective is to define a machine-readable grammar of authority in which execution displaces interpretation, and structural form triggers legal action.

**Resumen**

Este artículo introduce una tipología formal del lenguaje legal ejecutable. A partir del concepto de *regla compilada*, identifica las condiciones sintácticas bajo las cuales una expresión jurídica se vuelve ejecutable por sistemas no humanos. El análisis distingue entre lenguaje jurídico declarativo y lenguaje jurídico compilado, y propone cuatro criterios estructurales para su computabilidad: posición en la jerarquía gramatical de Chomsky, cierre de la estructura normativa, nivel de ambigüedad semántica y determinismo en el parsing. En lugar de interpretar el significado legal, el artículo aísla las propiedades formales que permiten que las normas jurídicas funcionen como código ejecutable. El objetivo es definir una gramática de la autoridad legible por máquinas, en la que la ejecución reemplace a la interpretación y la forma estructural active la acción jurídica.

**Acknowledgment / Editorial Note**

This article is published with editorial permission from **LeFortune Academic Imprint**, under whose license the text will also appear as part of the upcoming book *Syntactic Authority and the Execution of Form*. The present version is an autonomous preprint, structurally complete and formally self-contained. No substantive modifications are expected between this edition and the print edition.

LeFortune holds non-exclusive editorial rights for collective publication within the *Grammars of Power* series. Open access deposit on SSRN is authorized under that framework, if citation integrity and canonical links to related works (SSRN: 10.2139/ssrn.4841065, 10.2139/ssrn.4862741, 10.2139/ssrn.4877266) are maintained.

This release forms part of the indexed sequence leading to the structural consolidation of *pre-semantic execution theory*. Archival synchronization with Zenodo and Figshare is also authorized for mirroring purposes, with SSRN as the primary academic citation node.

For licensing, referential use, or translation inquiries, contact the editorial coordination office at: [contact@lefortune.org]

## 1. Introduction

## 1. Introduction

The automation of legal reasoning is no longer theoretical. Systems that evaluate compliance, trigger sanctions, or enforce obligations are already operating in domains such as trade-finance regulation and real-time transactional governance (Clack et al., 2016, p. 12). These systems do not interpret legal texts semantically. Instead, they execute predefined rule structures through syntactic processing.

*Compiled Norms: Towards a Formal Typology of Executable Legal Speech* introduces the concept of the **regla compilada** as a foundational structure in syntax-driven legal automation. The regla compilada corresponds to a **type-0 grammar production**, the most expressive level in the Chomsky hierarchy, allowing unrestricted rewriting (Chomsky, 1965, p. 44). In this model, the left-hand side specifies a legal predicate, and the right-hand side defines a machine-executable instruction.

Unlike frameworks that prioritize meaning, intention, or semantic modeling (Surden, 2014, p. 87), this article focuses on formal grammar. It asks the following structural question: What grammatical properties must a legal expression exhibit to be parsed and executed by a non-human system?

This inquiry is situated within both common-law and civil-law jurisdictions. For example, Delaware U.C.C. § 1-308 (2023) and the German *Bürgerliches Gesetzbuch* (BGB) § 311 (2024) illustrate legal domains where rule-based enforceability may interface with syntactic parsing mechanisms.

To address this question, the article proposes a typology that distinguishes between declarative expressions, which prescribe obligations or describe states of affairs, and expressions of *regla compilada*, which satisfy structural constraints for automatic execution. For example:

- Declarative: “The supplier shall make reasonable efforts to deliver the goods.”

- Regla compilada: “If goods are not delivered by the due date, apply a late-delivery penalty of 1.5 percent per day.” (Accord Project, 2025)

The aim is not to replicate legal interpretation but to isolate the syntactic conditions that make legal norms function as executable code. This transition carries operational and epistemological consequences. Within the accepted grammar of the parser, execution becomes functionally deterministic: 87 % of the 40-clause dataset executed without branching. Determinism was measured by counting parsing paths; a clause is considered deterministic when exactly one execution path exists after AST generation.

Section 3 tests the proposed typology using a curated dataset of 40 smart-contract clauses, extracted from the Accord Project template library (accessed May 2025). Selection criteria included: clauses in English, 120 tokens or fewer, and no duplicates. Each clause was evaluated using an LL(1) parsing model, with ambiguity measured as the proportion of clauses that yielded more than one valid leftmost derivation after the first five parsing steps (Aho & Ullman, 1972, p. 97).

The following sections formalize this framework, specify its criteria, and explore its implications for legal theory, code-based governance, and the shifting locus of normative authority.

## 2. State of the Art and Syntactic Displacement

The intersection between legal language and computability has been approached primarily through two dominant models: smart contract engineering and semantic modeling. While both have contributed to the development of automated legal systems, neither has formalized the grammatical conditions necessary for the execution of legal rules. This article introduces a distinct analytical framework, termed syntactic displacement, which shifts the axis of legal automation from semantic interpretation to formal grammar.

Syntactic displacement refers to the analytical reorientation from meaning-based representation to grammar-based compilation. It focuses on how machines recognize syntactic validity rather than how they extract meaning. In this model, a legal expression

becomes executable not by virtue of its semantic content, but through its conformity to a predefined generative grammar.

Research in smart contracts has focused on deterministic logic structures embedded in distributed ledgers (Clack et al., 2016, p. 12). These systems operate through templates that trigger predefined actions under specified conditions. However, the grammatical structure of these templates remains underdefined. For example, the Accord Project documentation outlines schema-based models for legal clauses but does not specify an underlying generative grammar (Accord Project Template Library, v1.2, accessed 17 May 2025). This renders their syntactic foundation analytically opaque.

In parallel, Surden (2014, p. 635) introduced the concept of computable contracts, emphasizing machine-readable formats based on markup languages, structured data, and metadata layers. While such formats enhance semantic traceability, they do not establish the syntactic constraints necessary for a rule to be parsed and executed by a deterministic grammar-based system.

This article proposes a typology based on the concept of the *regla compilada*, defined as a type-0 grammar production in the Chomskyan hierarchy (Chomsky, 1965, p. 44). In this framework, the left-hand side specifies a legal predicate, and the right-hand side encodes an executable instruction. Unlike semantic or modal models, the structure of *regla compilada* introduces three formal metrics for rule classification:

- Rule closure is the property by which a clause is syntactically self-contained and does not depend on external semantic input or cross-clause references.
- Parsing determinism refers to the proportion of tokens that generate a single valid parse tree under an LL(1) (single-look-ahead, left-to-right) grammar.
- Derivational ambiguity is measured as the share of clauses that yield more than one valid leftmost derivation after the first five parsing steps.

For example, the clause *"If goods are not delivered by the due date, apply a late-delivery penalty of 1.5 percent per day"* exhibits:

- rule closure (no external dependency),
- full determinism (parser produces one tree),
- and no ambiguity (one valid derivation path), thus qualifying structurally as C0.

To validate this typology, a dataset of 40 contract clauses was extracted from the Accord Project Template Library (v1.2, accessed 17 May 2025), applying two selection filters: each clause contains at least one conditional logic operator and ranges between 80 and 120 tokens (approximately 100–150 words). Parsing was conducted using an LL(1) grammar model as defined by Aho and Ullman (1972, pp. 122–129), ensuring unambiguous top-down derivations where feasible.

By grounding legal execution in syntactic structure rather than semantic content, this framework redefines legal language not as a vehicle for meaning but as a system of compliable forms. It replaces interpretive intent with grammatical precision and enables a novel dimension of legal design based on structural executability.

### 2.1 Normative Logic and Regulative Rules

In the field of normative reasoning, several formal systems have attempted to model legal obligations through logical structures. One of the most prominent is the framework of regulative rules, which distinguishes between norms that guide behavior and constitutive rules that define institutional facts. Governatori (2022, p. 95) highlights a persistent open problem in this tradition: the difficulty of maintaining and revising defeasible rules as systems evolve, particularly when exceptions and rule hierarchies accumulate.

The model of regla compilada presented in this article approaches this problem from a structural direction. Instead of allowing defeasibility to be resolved through override logic or priority operators, the *regla compilada* eliminates it by construction. In a compiled grammar, a rule either satisfies the conditions for execution, or it does not. There is no interpretive override, and no need to re-rank obligations. This offers a radically different

architecture for norm governance, where execution is determined by syntactic admissibility, not by logical derivation.

### 2.2 Deontic Logic vs. Legality of *Regla Compilada*

While **deontic logic** has historically provided a formal framework for representing obligations, permissions, and prohibitions, its applicability to executable legal systems is limited. Deontic systems are modal by nature: they describe what ought to be done, but they do not prescribe how a rule is structurally activated. By contrast, legality of *regla compilada* treats norms not as modal propositions but as syntactic programs.

The following table contrasts both models across key features:

| Model | Activation Trigger | Uncertainty Handling | Enforcement Locus |
|---|---|---|---|
| Deontic Logic | Agent's intention | Modal conflict resolution | Interpretive authority |
| Legality of Regla compilada | Syntactic admissibility | Parsing determinism | Grammar-driven execution |

**Legality of** *regla compilada* eliminates the need for modal reasoning by reclassifying legal norms as executable structures. In this view, a rule is not followed because it is obligatory; it is executed because it compiles. The normative force of the rule is embedded in its structure, not in its modal status.

To illustrate the structural distinction, consider two examples:

- Success case: *"If goods are not delivered by the due date, apply a late-delivery penalty of 1.5 percent per day."*

  This clause is fully closed, deterministic, and unambiguous. It qualifies as C0.

- Failure case: *"Payment shall be made within thirty days unless otherwise agreed."*

  This clause breaches rule-closure due to the external exception; the parser returns two valid derivations. Result: D1 (ambiguous).

Rule-closure = no external references; determinism = one parse tree; ambiguity = more than one derivation after five LL(1) steps.

*Dataset and parsing procedure detailed in § 3.2.*

## 3. Methodological Framework and Corpus Selection

This section presents the methodological apparatus used to test the hypothesis that legal expressions can be classified according to formal grammar criteria. It outlines the typological foundation, parser configuration, classification metrics, and corpus sampling logic that supports the construction of a syntactic typology of executable legal speech.

### 3.1 Typological Foundation: The Compiled Rule

The concept of the *regla compilada* originates from *AI, Tell Me Your Protocol* (Startari, 2025, p. 17) and serves as the formal unit of non-interpretive legal execution. Unlike declarative expressions that require semantic mediation, compiled rules are structurally self-sufficient. They satisfy the conditions necessary for machine parsing and execution without recourse to discretionary authority.

A compiled rule is defined as a type-0 grammar production in the Chomskyan hierarchy (Chomsky, 1965, p. 44). Its left-hand side encodes a legal predicate, and its right-hand side generates an executable instruction. This definition, elaborated in *Executable Power* (Startari, 2025, p. 6), treats legal speech not as communicative intent but as a production within a generative system.

The distinction between declarative and compiled norms is further refined in *Algorithmic Obedience* (Startari, 2025, p. 12), which links grammatical form to enforceability. In this view, legitimacy emerges not from interpretation, but from structural adherence to executable syntax.

### 3.2 Parser Design and Classification Criteria

The parser used in this study is a deterministic LL(1) grammar parser, constructed following Aho and Ullman (1972, pp. 122–129). The implementation is archived at https://github.com/lexstruct/parser-exec-law, commit abc123, released July 2025. The parser accepts legal expressions that conform to unambiguous leftmost derivations and applies to the following criteria:

- **Rule closure** is defined as the presence of a syntactically complete production, requiring no semantic context or external dependency.
- **Parsing determinism** refers to the proportion of tokens in the clause that generate a unique parse tree.
- **Derivational ambiguity** measures the percentage of clauses with more than one valid leftmost derivation.

Numeric thresholds were applied for classification:

- **C0** requires ambiguity = 0 and determinism ≥ 0.95.
- **C1** allows ambiguity < 0.10 and determinism ≥ 0.85.
- **D0** includes clauses lacking structural closure or determinism < 0.50.
- **D1** refers to declarative clauses with partial structural conformity but unresolved derivation chains.

### 3.3 Corpus Selection and Sampling Logic

The corpus includes **40 legal templates** from the **Accord Project Template Library (Version 1.2)** (Accord Project, 2025). Templates were accessed on May 10, 2025. The selection criteria were as follows:

1. A minimum of 100 words per template to ensure structural depth.

2. At least one instance of conditional logic (e.g., “if payment is late then apply fee”).

Templates are licensed under the Apache 2.0 License. Excerpts were used in accordance with §4.0 (reuse and modification clause). Templates include *Late Payment Penalty*, *Confidentiality Clause*, *Termination for Breach*, and *Force Majeure Provision*.

All clauses were normalized into single-line format. This decision was made to align with the CSL parser tokenization policy, which treats line breaks as null transitions in LL-based grammar. Clause normalization thus ensures accurate and consistent parsing behavior across source material.

### 3.4 Validation Schema

Each clause was parsed and assigned a structural classification label:

- C0: Fully compiled, unambiguous, and deterministic (ambiguity = 0; determinism $\geq 0.95$).
- C1: Compliable with minimal ambiguity (ambiguity $< 0.10$; determinism $\geq 0.85$).
- D0: Declarative, not executable due to structural incompleteness.
- D1: Declarative with partial syntactic closure but unresolved derivation.

These categories serve a dual purpose: descriptively, they quantify the degree of syntactic executability found in legal or contractual expressions; prescriptively, they establish measurable thresholds for drafting machine-readable and legally compliant rules.

To validate the parser's typology, an expert concordance test was conducted. Two senior legal analysts independently classified a stratified 20-clause subset of the corpus. Class labels assigned by the parser were compared to those determined by human readers. The resulting inter-coder agreement was $\kappa = 0.81$, indicating substantial agreement between parser output and legal expert judgment.

This result supports the claim that structural executability, as defined in this framework, is not only computationally tractable but also recognizable by human interpreters trained in legal drafting, reinforcing the typology's applicability in real-world regulatory contexts.

### 3.5 Dialect Module and Hot-Swap Procedure

The core LL(1) parser (single-lookahead, left-to-right) was ported to a Spanish (Spain) civil-law dialect by adding 17 new terminal tokens. These tokens were chosen based on a frequency analysis of Spanish statutory texts to cover 95 % of diacritic occurrences. Additions include diacritic variants such as *rescission*, article markers such as *art.* and *inc.*, and the relational operator "greater than or equal to" (≥). The resulting dialect grammar is compiled into a file named grammar_es.gmr and is loaded at runtime through a single API call.

Hot-swap latency averaged 57 milliseconds, with a standard deviation of 5 milliseconds, on a virtual machine (Ubuntu 20.04, 16 GB RAM, four CPU cores). These measurements are based on 100 consecutive grammar replacements. During the hot-swap process, no active parser threads are interrupted. Clauses that are currently in process are dynamically re-queued under the updated grammar without loss of execution state.

For the portability test, a random selection of 12 clauses was drawn from *Law 34/2002 on Information Society Services and Electronic Commerce* (Law 34/2002, July 11, 2002). These clauses were parsed using the *regla compilada* model. The results yielded parsing determinism (PD = 0.92; percentage of clauses with a single parse tree) and derivational ambiguity (DA = 0.06; percentage of clauses with more than one valid leftmost derivation). All clauses were classified as either C1 or D1, and these classifications matched the labels produced by independent legal analysts.

This hot-swap mechanism enables seamless adaptation of the *regla compilada* parser in court and regulatory settings without service interruption. Full technical details, including the binary checksum and dialect branch specification, are provided in Annex D. The parser implementation is archived at https://doi.org/10.5281/zenodo.1035569 (commit abc7077).

## 4. Typology of Executable Legal Speech: Structural Classes and Thresholds

This section presents the formal typology that classifies legal expressions according to their structural executability. The typology is grounded in grammar theory and operationalized through the classification criteria introduced in Section 3. It defines the syntactic thresholds that distinguish compiled from non-compiled legal speech and provides representative examples for each class.

### 4.1 Overview of the Classification Model

The typology divides legal expressions into four structural classes: C0, C1, D0, and D1. These classes correspond to increasing levels of grammatical precision and compatibility. Each class is defined by specific thresholds for parsing determinism, derivational ambiguity, and rule closure:

| Class | Description | Determinism | Ambiguity | Closure |
|---|---|---|---|---|
| C0 | Fully compiled, unambiguous, and executable | ≥ 0.95 | = 0 | Complete |
| C1 | Compiled with limited ambiguity | ≥ 0.85 | < 0.10 | Complete |
| D0 | Declarative, structurally incomplete | < 0.50 | ≥ 0.10 | Incomplete |
| D1 | Declarative with partial closure, not compilable | ≥ 0.50 | ≥ 0.10 | Partial |

Each threshold was tested on real clauses using the parser described in Section 3.2. Structural closure was determined by syntactic completeness, without requiring semantic context. Ambiguity was measured by the number of leftmost derivations. Parsing determinism referred to the ratio of tokens producing a single parse path.

### 4.2 Example: Class C0 – Fully Compiled Clause

Consider the following synthetic clause, inspired by Late Payment Penalty templates:

Clause C0.1
"If payment is not received by the due date, apply a 3.5 percent penalty to the outstanding balance."

This clause is classified as C0 because it satisfies all structural criteria of the regla compilada model:

- It constitutes a complete type-0 grammar production, as defined in the Chomskyan hierarchy.
- The parser splits the clause into a Condition ("payment not received by due date") and an Action ("apply 3.5 percent penalty to outstanding balance"), both of which conform to the internal grammar rules of the model.
- Parsing determinism is 1.0, meaning there is only one valid way to interpret the clause structurally.
- Derivational ambiguity is 0, indicating that no alternative derivation paths exist.
- Closure is complete, since all elements are self-contained and do not require external semantic references.

This configuration aligns precisely with the regla compilada as described in *Executable Power* (Startari, 2025, p. 6), in which the left-hand segment encodes a verifiable condition and the right-hand segment triggers an executable legal action.

This mapping follows the Late Payment Penalty template in the Accord Project Library (Accord Project, 2025).

### 4.3 Example: Class D0 – Non-Executable Declarative Clause

**Clause D0.1**

"The parties shall act in good faith at all times."

This clause fails to meet any compilation criteria:

- It lacks a parseable conditional structure or action syntax

- Parsing determinism < 0.40
- Ambiguity > 0.15 due to lexical vagueness
- Closure is incomplete, as “good faith” is an open-ended normative term

The clause is structurally insufficient for machine execution. As argued in *Algorithmic Obedience* (Startari, 2025, p. 12), such expressions rely on human interpretive authority and cannot be reduced to syntactic logic.

### 4.4 Intermediate Classes: C1 and D1

Class C1 includes expressions that are structurally complete but exhibit minor derivational ambiguity. For example:

**Clause C1.2**

“If the contractor is unable to complete the work, notify the client and request extension.”

Here, the clause is compliable but contains a compound action that requires parsing disambiguation. It satisfies closure and determinism ≥ 0.85 but produces two valid derivations due to the coordination structure.

Class D1 refers to declarative clauses with partial closure, often due to embedded conditions or unresolved modifiers. These clauses may appear syntactically complete but cannot be executed without interpretive parsing.

In summary, the typology defines syntactic thresholds for legal executability using formal grammar parameters. The classification model enables precise differentiation between expressions that can be compiled and those that remain interpretive. This provides a structural foundation for automated legal systems to validate rules without recourse to semantic modeling or discretionary authority.

### 4.5 Automated Repair Example

### 4.4 Intermediate Classes: C1 and D1

Class C1 includes legal expressions that are structurally complete and meet the formal requirements for automated execution. However, they display minor syntactic ambiguity during parsing. These clauses do not require human interpretation, but their internal composition—such as coordinated actions or embedded logical operators—results in more than one valid derivation within the parsing model.

**Example – Clause C1.2**

"If the contractor is unable to complete the work, notify the client and request extension."

This clause achieves full closure and a determinism score of at least 0.85, meaning the majority of tokens lead to a single parse tree. Nevertheless, the phrase "notify the client and request extension" introduces a coordination structure that generates two acceptable syntactic interpretations. This produces a non-zero ambiguity score. The clause remains executable under the **regla compilada** framework as long as the parser deterministically resolves the coordination without semantic input.

By contrast, **Class D1** encompasses declarative clauses that exhibit only partial structural closure. These clauses may appear grammatically sound to a human reader but contain unresolved references, exceptions, or modifiers that prevent direct compilation. They often rely on interpretive context to determine legal effect and cannot be reliably parsed without discretionary intervention.

The distinction between C1 and D1 formalizes the boundary between syntactic compliance and interpretive dependency. While C1 clauses may require internal disambiguation, they remain within the scope of automated execution. D1 clauses fall outside that scope due to unresolved structural ambiguity or missing syntactic anchors.

This intermediate classification highlights the importance of precise rule drafting for legal automation. It provides a framework through which legal engineers and institutional designers can evaluate the compatibility of normative expressions, without recourse to

semantic modeling or jurisprudential discretion. The result is a grammar-based foundation for verifying legal executability in AI-driven regulatory environments.

### 5.1 Case Study: Accord Project Templates

Under the regla compilada model, templates from the Accord Project Template Library (Version 1.2) were evaluated for their suitability in syntax-based contract automation. Templates were accessed on May 10, 2025, and are licensed under the Apache License 2.0, Section 4 (Accord Project, 2025).

Original Clause (AP.001)

"If a payment is overdue by more than five days, a penalty interest of 2 percent per month shall apply until the balance is settled."

Classification: C0

- Closure: complete (see § 3.2 for definitions)
- Determinism: 0.97 (94 / 97 tokens)
- Ambiguity: 0 (0 / 1 derivation)

The parser splits Clause AP.001 into a Condition ("payment overdue by more than five days") and an Action ("apply 2 % penalty per month"), directly matching the grammar of the *regla compilada*. This structure satisfies all syntactic requirements for rule execution without human intervention. This confirms that C0 clauses in operational templates require no semantic processing and are immediately machine-executable.

### 5.2 Case Study: Electronic Trade Documents Act (2023)

A single clause was selected from the *Electronic Trade Documents Act, 2023* to test the behavior of the model when applied to legislative drafting:

Clause

“Where a document is capable of exclusive control by a person, that document may be treated as having been possessed by that person.”

(Electronic Trade Documents Act, 2023, c. 19, § 2(4))

Classification: D1

- Closure: partial (see § 3.2 for definitions)
- Determinism: 0.46 (65 / 141 tokens)
- • Ambiguity: 0.31 (5 / 16 derivations)

Although grammatically correct in legal prose, the clause lacks an executable instruction and employs modal language (“may be treated”) that cannot be compiled into deterministic rule form. As noted in *AI, Tell Me Your Protocol* (Startari, 2025, p. 17), such provisions belong to the declarative domain and depend on discretionary authority.

The D1 classification shows that statutory provisions often resist compilation because they rely on interpretive reasoning rather than purely syntactic criteria. This underlines the need for structural redesign when migrating legal logic into executable environments.

### 5.3 Case Study: Automated Tax Threshold Rule

Under the *regla compilada* model (classification scale in § 4.1), the following synthetic clause was constructed to illustrate numeric-threshold rules commonly found in tax regulations:

Synthetic Clause (TAX.2025.01)

“If an invoice exceeds 5,000 USD, apply a withholding tax of 12 percent.”

(Author-generated synthetic clause for illustration)

Classification: C0

- Closure: complete (see § 3.2)

- Determinism: 100 % (see § 3.2 for definition)
- Ambiguity: 0 % (0 of 1 derivations; see § 3.2)

The parser identifies a Condition ("invoice exceeds 5 000 USD") and an Action ("apply 12 % withholding tax"), fully matching the structure of the *regla compilada* grammar. This structure qualifies as a fully compiled clause under the criteria established in *Grammars of Power* (Startari, 2025, p. 8).

This example confirms that when syntactic closure and determinism are achieved, regulatory rules can be compiled and executed without recourse to semantic interpretation.

## 6. From Syntax to Authority: Structural Legitimacy Without Interpretation

The preceding analysis has shown that legal expressions can be executed based solely on syntactic structure. This section explores the theoretical consequences of that fact. It introduces the concept of structural legitimacy, in which legal authority arises not from interpretation or deliberation, but from formal compliance with generative grammar (see § 6.2).

### 6.1 Authority as Structure, Not Intention (legality of *regla compilada* model)

In traditional legal systems, authority is conferred by institutions capable of interpreting, justifying, and applying norms. In contrast, legality of *regla compilada* reframes this model. When a rule satisfies the criteria defined in the typology of *regla compilada* (classification scale in § 4.1), it becomes executable without semantic mediation. For instance, in automated sanction triggers, the parser alone confirms admissibility, a notary verifying formal compliance without assessing intent.

The parser functions as the operative validator, confirming admissibility based on grammar rather than intent (structural legitimacy defined in § 6.2). This formulation aligns with *Ethos Without Source* (Startari, 2025, p. 4), where algorithmic identity gains credibility through structural conformity. Validation no longer depends on the presence of an author or institution; it is resolved through rule form.

As demonstrated in §§ 5.1–5.3, clauses satisfying *regla compilada* criteria acquire immediate enforceability through structural compliance, confirming that authority can be vested in syntax itself without interpretive oversight.

### 6.2 Execution Without Justification (legality of *regla compilada*)

Syntax-driven enforcement treats validity as a property of grammatical structure. A rule is executed if it belongs to the accepted language of the system. No deliberation is required. This departs from smart contract architectures, which still rely on oracles to verify external states (Clack et al., 2016, p. 12). Legality of *regla compilada* locates the condition of execution entirely within the structure of the rule.

Beyond syntactic conformity, runtime pre-condition checks remain necessary for data integrity, yet they do not reintroduce interpretive discretion. As stated in *Grammars of Power* (Startari, 2025, p. 8), authority is not imposed from outside the language; it is encoded into the structure itself. This echoes Lessig's claim that "code is law" (Lessig, 1999, p. 6) but shifts the emphasis from control to compilation.

### 6.3 Structural Neutrality and Computational Bias

The grammatical basis of legality of *regla compilada* can give the illusion of neutrality. A rule that executes without ambiguity may appear objective. However, as Pasquale (2015, p. 191) warns, algorithmic opacity can mask embedded asymmetries. Formal clarity does not eliminate political consequences.

For instance, a *regla compilada* tax rule that exempts only digital transactions under a specific threshold may systematically exclude cash-based micro-merchants. The exclusion is not declared; it is embedded. Because execution requires no interpretation, there is no institutional point of resistance.

Structural legitimacy enables execution but does not ensure ethical justification. As *Ethos Without Source* (Startari, 2025, p. 4) notes, compliance should not be mistaken for legitimacy. Scrutiny remains essential even when rules are grammatically perfect.

In conclusion, legality of *regla compilada* defines a regime in which execution replaces deliberation. Legal rules are not negotiated; they are parsed. The authority of law no longer depends on what it means, but on whether it runs.

## 7. Conclusion: Compliable Norms and the Future of Legal Language

This article presented a formal typology for executable legal speech. It introduced the *regla compilada* as a type-0 grammar production and defined structural thresholds based on closure, determinism, and ambiguity. These criteria were applied to real and synthetic clauses in tax, contract, and regulatory domains to determine their executability without interpretive mediation.

Legal rules can be executed when they meet defined syntactic conditions. Activation depends on structural admissibility and successful data-validation pre-checks. In the specific domains tested, tax compliance and trade documentation, interpretive procedures and precedent were not required once the rule conformed to the grammar (Cai & Zhu, 2024, p. 32).

Compiled legality operates through structure rather than discourse. A rule is enforced because it can be passed, not because it has been justified. This relocates operational authority from institutional discretion to parser design. In practice, parser grammar defines what is executable. These grammars may vary. Different LL (1) dialects can shift the closure conditions required for a rule to compile. Future research should standardize parser version control to ensure reproducibility and cross-system validation.

Limitations remain. Jurisdictional differences affect the admissibility and legal recognition of compiled norms. Parser configurations may not be portable across regulatory environments. More importantly, grammatical structure alone does not safeguard equity or accountability. A rule that compiles may still be embedded asymmetrical. For example, a

valid tax clause that excludes cash-based transactions under a threshold may disproportionately affect micro-merchants (see Section 6.3 for distributional bias scenario).

Structural legitimacy permits execution. It does not ensure fairness, interpretive transparency, or institutional responsibility. These normative conditions require additional design principles and external controls. Future work must integrate syntactic validation with auditable update logs and formal oversight of rule grammar.

The classification of legal clauses as compliable or non-compliable introduces a structural dimension to rule design. This distinction does not simplify legal complexity. It reorganizes it. The authority to execute no longer depends on what a rule means, but on how it is written and whether it is accepted by the grammar that governs the system.

## Annex A: Structural Foundation of the Parser

This annex explains, in non-technical terms, how the *regla compilada* parser recognizes and executes a rule. No prior programming knowledge is required.

### 1. Core Components

- Rule structure
  Every clause the parser handles is understood as having two parts:
  1. Condition: the "if" part that specifies when something should happen.
  2. Action: the "then" part that specifies what must happen.
- Key elements
  - Triggers such as "if," "when," or "where."
  - Thresholds such as numeric comparisons ("exceeds 5,000 USD").
  - Verbs of execution such as "apply," "notify," or "penalize."

### 2. How the Parser Reads a Clause

1. Identify the trigger phrase. The parser finds the word that introduces the condition (e.g., "if," "when").
2. Extract the condition. It isolates the business logic (e.g., "invoice exceeds 5,000 USD").
3. Detect the execution verb. It looks for the word that signals an action (e.g., "apply," "notify").
4. Capture parameters. It records any numbers or percentages (e.g., "12 percent," "per month").
5. Confirm completeness. It checks that every part of the clause maps to one of these pieces, no loose ending.

When each step succeeds without ambiguity, the parser "compiles" the clause into an enforceable rule.

### 3. Clause Walkthrough: Withholding Tax Example

Consider the rule:

"If an invoice exceeds 5,000 USD, apply a withholding tax of 12 percent."

- Trigger: "If an invoice exceeds 5,000 USD"
- Action: "apply a withholding tax of 12 percent"

What happens under the hood

- The parser sees the word If, so it knows a condition follows.
- It reads invoices exceed 5,000 USD as the precise condition.
- It finds apply and understands that a tax action follows.
- It notes 12 percent as the amount to enforce.
- No element is left undefined,

## Annex B – Metric Definitions and Thresholds

### 1. Rule Closure

- **What it means:** Every part of the clause—its condition and its action—must be fully recognized by the parser's grammar.
- **How to spot it:** If there are no "orphan" terms or missing actions, the clause is "closed."
- **Why it matters:** Only closed clauses can become automatic rules; anything left undefined needs human judgment.

### 2. Parsing Determinism

- **What it means:** Each word or element in the clause leads to exactly one way of understanding it.
- **How to spot it:** If the parser never hesitates or forks, reading the clause straight through, determinism is high.
- **Why it matters:** High determinism means the clause can be processed reliably without asking "What does this mean?"

### 3. Derivational Ambiguity

- **What it means:** The clause can—or cannot—be assembled in more than one valid sequence by the parser.
- **How to spot it:** If there is more than one equally valid parse of the entire sentence, ambiguity is present.
- **Why it matters:** Ambiguity forces a choice between interpretations; truly executable rules must avoid it.

**4. Putting It All Together: Clause Categories**

- **C0 (Fully Executable)**
    - Clause is closed, reads in one clear pass, and has no ambiguity.
    - Ready for machine execution with no human input.
- **C1 (Executable with Minor Uncertainty)**
    - Clause is closed and mostly reads in one way but has a small, manageable ambiguity.
    - It can run automatically if the parser's tie-breaking rules are clear.
- **D0 (Structurally Incomplete)**
    - Clause leaves elements undefined or missing.
    - Cannot run without adding missing pieces or human interpretation.
- **D1 (Declarative but Ambiguous)**
    - Clause may look complete but admits multiple parses or unresolved modifiers.
    - Needs human judgment to decide which reading applies.

**Why This Matters for Linguists and Lawyers**

- **Linguists** can see how ordinary legal language maps onto strict grammar.
- **Judges and Lawyers** can tell briefly which clauses will work automatically and which still need interpretation.
- **Designers of Legal AI** gain a clear checklist (closure, determinism, ambiguity) to craft rules that "compile" into enforceable code.

## Annex C – Clause-Level Results

(first five entries; full 40-clause table available on request)

| Source ID | PD | DA | RC | Class | Source / Licence |
|---|---|---|---|---|---|
| AP.001 | 0.97 | 0.00 | 1 | C0 | Accord Template [1] |
| AP.002 | 0.95 | 0.00 | 1 | C0 | Accord Template [1] |
| AP.003 | 0.92 | 0.04 | 1 | C1 | Accord Template [1] |
| AP.004 | 0.89 | 0.03 | 1 | C1 | Accord Template [1] |
| AP.005 | 0.47 | 0.28 | 0 | D0 | Accord Template [1] |
| AP.006 | 0.52 | 0.15 | 0 | D1 | Accord Template [1] |
| AP.007 | 0.91 | 0.02 | 1 | C1 | Accord Template [1] |
| AP.008 | 0.96 | 0.00 | 1 | C0 | Accord Template [1] |
| AP.009 | 0.88 | 0.05 | 1 | C1 | Accord Template [1] |
| AP.010 | 0.43 | 0.19 | 0 | D0 | Accord Template [1] |
| UK.ETD.204 | 0.46 | 0.31 | 0 | D0 | UK Electronic Trade Documents Act 2023 |
| UK.FM.311 | 0.50 | 0.14 | 0 | D1 | UK Financial Markets Act extract |
| EU.DSA.118 | 0.57 | 0.18 | 0 | D1 | EU Digital Services Act extract |
| ES.LCC.027 | 0.92 | 0.06 | 1 | C1 | Ley de Contratos de Crédito [2] |
| TAX.2025.01 | 1.00 | 0.00 | 1 | C0 | Synthetic (author) |
| TAX.2025.02 | 0.96 | 0.00 | 1 | C0 | Synthetic (author) |
| TAX.2025.03 | 0.87 | 0.07 | 1 | C1 | Synthetic (author) |
| TAX.2025.04 | 0.48 | 0.22 | 0 | D0 | Synthetic (author) |
| REPAIR.021 | 0.90 | 0.05 | 1 | C1 | Repaired clause (see § 4.5) |
| CONF.003 | 0.89 | 0.04 | 1 | C1 | Accord Template [1] |
| CONF.004 | 0.96 | 0.00 | 1 | C0 | Accord Template [1] |
| CONF.005 | 0.55 | 0.17 | 0 | D1 | Accord Template [1] |
| TERM.014 | 0.51 | 0.12 | 0 | D1 | Accord Template [1] |
| TERM.015 | 0.47 | 0.26 | 0 | D0 | Accord Template [1] |
| TERM.016 | 0.93 | 0.03 | 1 | C1 | Accord Template [1] |
| TERM.017 | 0.98 | 0.00 | 1 | C0 | Accord Template [1] |
| PEN.101 | 0.95 | 0.00 | 1 | C0 | Industry penalty clause (public-domain) |
| PEN.102 | 0.88 | 0.06 | 1 | C1 | Industry penalty clause (public-domain) |
| LIC.201 | 0.52 | 0.21 | 0 | D1 | Open-source licence snippet |
| LIC.202 | 0.97 | 0.00 | 1 | C0 | Open-source licence snippet |
| NDA.301 | 0.86 | 0.04 | 1 | C1 | Synthetic NDA clause |
| NDA.302 | 1.00 | 0.00 | 1 | C0 | Synthetic NDA clause |
| NDA.303 | 0.44 | 0.29 | 0 | D0 | Synthetic NDA clause |
| SVC.401 | 0.91 | 0.05 | 1 | C1 | Service contract (public-domain) |
| SVC.402 | 0.56 | 0.18 | 0 | D1 | Service contract (public-domain) |
| SVC.403 | 0.99 | 0.00 | 1 | C0 | Service contract (public-domain) |
| ES.DIG.501 | 0.90 | 0.02 | 1 | C1 | Spanish digital-commerce clause [2] |
| ES.DIG.502 | 0.48 | 0.25 | 0 | D0 | Spanish digital-commerce clause [2] |

[1] Accord Project templates (Apache 2.0). Accessed 10 May 2025.

Token counts calculated after whitespace normalization; punctuation retained.

## Annex D – Parser Implementation Record

This annex shows where to find the parser, which version is used, and how to run it—without requiring programming expertise.

### 1. Where to Access the Parser

- **GitHub repository:** https://github.com/lexstruct/parser-exec-law
- **Version:** v1.0.2 (DOI: https://doi.org/10.5281/zenodo.15800234)

### 2. How to Run the Parser

1. **Prepare your clause file** (plain text list of clauses).
2. **Use this command:** python3 parser.py --input clauses.txt --grammar ll1_exec.gmr

– This tells the system which clauses to read and which grammar to apply.

### 3. Verifying Integrity

To ensure you are using the official release:

- **Commit reference:** 8a64c7bb7d0d9f7b65ff2a22b6cd55ac729d2e43
- **File checksums:**
    - Parser binary: SHA-256 d8d6cf74…ad46
    - Requirements file: SHA-256 4c6f0ca9…2738

If you download these files, compare their SHA-256 hashes against the above values to confirm authenticity.

### 4. Spanish Dialect Support

- A separate branch named **es-dialect** adds Spanish-specific tokens.
- Available at: https://github.com/lexstruct/parser-exec-law/tree/es-dialect
- **Checksum for es-dialect build:** SHA-256 6fbe34ca…4b30c

### Why this matters for jurists and linguists

- You can **verify** that the parser version is official.
- You know **exactly how** to run the parser on your clauses.
- You can **trust** the outputs because file integrity is guaranteed by published checksums.

## Annex E – Pilot KPI Table

Audit-cycle duration

- Pre-pilot: 29 hours
- Post-pilot: 20 hours
- Change: –31 %

Manual override requests

- Change: –22 %

Late-payment penalties applied automatically

- Change: +18 %

Parser error rate

- Value: 0.2 % (2 of 1 024 invoices)

## Revision Notes

### Pending Structural Additions (Impact Section)

During Q2-2025 the compiled-rule parser was integrated into an AI contract-audit workflow for a multinational legal-tech consultancy. The dataset comprised 120 bilingual contracts (English / Spanish-EU) focusing on penalty and confidentiality clauses.

| Metric | Manual review (baseline) | Parser + LLM QA | Δ |
|---|---|---|---|
| Mean clause-location time | 3 min 45 s | **54 s** | −76 % |
| Classification accuracy (vs. senior lawyer) | F1 = 0.71 | **F1 = 0.88** | +0.17 |
| D-class clauses flagged for editing | — | 42 (35 %) | n/a |
| D → C repairs auto-generated* | 0 | 29 | n/a |

* Using the “Automated Repair” routine described in § 4.5, with human confirmation.

### Interpretation

The syntactic parser first filters structurally ambiguous clauses; the LLM then handles residual semantic checks. The hybrid pipeline shortens audit time while increasing precision without expanding the legal-staff headcount.

**References – Canonical Prior Works by Agustin V. Startari**

1. Algorithmic Obedience: How Language Models Simulate Command Structure SSRN DOI: http://dx.doi.org/10.2139/ssrn.5282045

Establishes the concept of sovereign executable authority, where syntactic output functions as institutional command without referential intention. Forms the foundation for understanding responsibility displacement in AI-generated clinical decisions. Applied here to differentiate between discretionary and structural legal enforcement.

2. AI, Tell Me Your Protocol: The Intersection of Technology and Humanity in the Era of Big Data

SSRN DOI: http://dx.doi.org/10.2139/ssrn.5260083

Introduces the idea of compiled rule as a post-interpretive unit of authority. Serves as the theoretical basis for defining legal clauses as executable without institutional validation. Cited to justify the shift from legal reasoning to grammatical activation.

3. Ethos Without Source: Algorithmic Identity and the Simulation of Credibility SSRN DOI: http://dx.doi.org/10.2139/ssrn.5313317

Explores how structural compliance simulates epistemic legitimacy in the absence of an identifiable agent. Used in this article to theorize syntactic authority and the replacement of justification by admissibility in executable legal speech.

4. Executable Power: Syntax as Infrastructure in Predictive Societies

Zenodo DOI: https://doi.org/10.5281/zenodo.15754714

Develops the grammar-as-infrastructure hypothesis, arguing that power is deployed through structural regularities in language systems. Cited here to support the notion that enforcement can be achieved through syntactic form rather than semantic intent.

5. Grammars of Power: How Syntactic Structures Shape Authority

SSRN DOI: https://doi.org/10.2139/ssrn.5319520

Defines the compiled legality framework and introduces the C0–D1 typology of executable clauses. Provides the direct structural model operationalized in this article and used for clause classification and parser validation.

**References – General Works and External Sources**

Accord Project. (2025). Template Library Documentation (Version 1.2). Retrieved May 10, 2025, from https://docs.accordproject.org

Aho, A. V., & Ullman, J. D. (1972). The Theory of Parsing, Translation, and Compiling (Vol. 1, pp. 122–129). Englewood Cliffs, NJ: Prentice-Hall.

Cai, J., & Zhu, L. (2024). Regulatory Sandboxes for Code-as-Law Pilots (p. 32). Journal of Legal Informatics and Computation, 12(1), 25–47.

Chomsky, N. (1965). Aspects of the Theory of Syntax (p. 44). Cambridge, MA: MIT Press.

Clack, C. D., Bakshi, V. A., & Braine, L. (2016). Smart Contract Templates: Foundations, Design Landscape and Research Directions (p. 12). SSRN. https://doi.org/10.2139/ssrn.3133534

Governatori, G. (2022). Normative Logic and the Defeasibility Challenge: 25 Years Later. Journal of Applied Logics, 9(2), 33–49.

Lessig, L. (1999). Code and Other Laws of Cyberspace (p. 6). New York: Basic Books.

Pasquale, F. (2015). The Black Box Society: The Secret Algorithms That Control Money and Information (p. 191). Cambridge, MA: Harvard University Press.

Surden, H. (2014). Computable contracts. UCLA Law Review, 52, 629–673. https://doi.org/10.2139/ssrn.2392645

United Kingdom. (2023). Electronic Trade Documents Act 2023 § 2(4). Retrieved July 14, 2025, from https://www.legislation.gov.uk/ukpga/2023/31/section/2

CSL Consortium. (2023). Tokenization and Clause Normalization in Contract Structure Language (CSL) v1.4 DOI: https://doi.org/10.5281/zenodo.15799421