

Testing the Rules as Executed against the Rules as Promised

Anne SCHUTH^{a,1} Eelco HOTTING^a

^a*Dutch Digital Service, Ministry of Economic Affairs and Climate, The Netherlands*

Abstract. Where government executes the law from formalized rules, the formalization has to be tested against what the statute was meant to yield. But the expected outcomes a test needs almost always come from the formalizer, so the tests inherit the reading they were meant to check. The Dutch legislative record holds an oracle that neither the encoder nor the executing organization sets: the worked examples with which the government shows parliament what a bill should arrive at. A full-text harvest scores 5,068 passages in 2,415 documents as candidates, and a recall check suggests the harvest finds at most about half. We transcribed seventeen examples under a rule that forbids inventing cases and ran fifteen against a public corpus of encoded statutes. The other two exemplify provisions the corpus does not model. Eleven reproduce the promised amount. Among the four failures is a household the government’s own amendment note had falsified before the act took effect, and auditing the expectations around it found three defects in our encodings. The same material shows what the oracle cannot do. An example states a third of the inputs an encoding needs, and rarely places one on a threshold. Of the 53 faults seeded into the allowance scheme, the encoders’ twelve scenarios catch 23 and two built from the memorandum’s arithmetic catch 12, each missing what the other finds. Examples go stale as soon as a parameter moves, and they rarely touch the open norms where the concretizations live. Worked examples belong in the record of how a statute becomes executable rules: the specification file the Council of State asked for in 2021, maintained with the law and layered with the policy rules and judgments that carry what a memorandum cannot.

Keywords. formalization of legislation, test oracle, explanatory memorandum, acceptance testing, mutation testing, large language models, rules as code

1. Introduction

When Sergot et al. formalized the British Nationality Act, the formalization was judged against its authors’ own reading of the Act [1]. Software engineering calls this the oracle problem: the difficulty of obtaining a procedure that judges whether behavior is correct [2]. In legal formalization the judge is nearly always the encoder, or the administering organization whose reading the encoding operationalizes. The tests then inherit the interpretation they were meant to check: a passing suite confirms the encoder’s reading rather than testing it, and only an oracle neither party sets can do that.

¹Corresponding author: Anne Schuth; e-mail: anne.schuth@rijksoverheid.nl. The work described here was carried out at the Dutch Ministry of the Interior and Kingdom Relations.

For Dutch legislation, an oracle external to both already exists. A bill travels to parliament with an explanatory memorandum (*memorie van toelichting*), in which the government explains what the proposal is meant to do. Where the bill calculates, that explanation usually contains worked examples, like a household at a stated income and the allowance it should receive. Each one states an intended outcome. It is on the record before any production system exists, it comes from neither the encoder nor the administering organization, and it can be transcribed into an executable acceptance test.

This matters now because language models are proposed to ease the bottleneck that drafting formalizations has always been [3,4], which moves it to validation, and because such specifications are proposed as published rules binding on the organization that publishes them [5], which makes an encoding error an error in the rule applied. Our position paper uses memorandum-derived scenarios without reporting counts, scenarios or a failure, and leaves open how a conformance suite is “kept in step with [a statute] as the law is amended”. This paper runs them, and contributes:

- a census of the oracle: 5,068 citizen-level passages in 2,415 documents, label precision 80 percent, recall at most about half (Section 3);
- seventeen scenarios mined under a discipline that forbids inventing a case, fifteen of them runnable; eleven reproduce the promised amounts, and among the failures is a household the government’s own amendment note had falsified before the act took effect (Sections 4–6);
- a mutation experiment measuring what each kind of suite catches that the other does not (Section 7);
- three measured limits: an example carries a third of the inputs an encoding needs, its parameters go stale (all 25 we checked differ from today’s values), and few touch the open norms (Section 8).

Three of the findings are defects in our own corpus. Two of them were invisible to a suite whose expected values we had read off the encoding it was meant to test.

2. The Oracle in the Parliamentary Record

A test oracle is whatever decides that observed behavior is the intended behavior [2]. In formalized law that decision has generally rested with the formalizer. Isomorphism, keeping one provision aligned with one rule, is the field’s structural answer to verifiability [6], but an isomorphic encoding can still compute the wrong allowance. Where encodings are tested against concrete cases at all, the expected outcomes come from the encoders, from the administering organization (the French tax compiler was validated against the tax authority’s private suite [7]), or from another encoding of the same rules [8,9]. In each case the expected outcome and the encoding share a source. Publishing the decision rules would expose the administering organization’s reading [5,10].

The explanatory memorandum offers an anchor with a different status. The ministry proposing the bill drafts the memorandum, usually after the administering organization has assessed the draft, and the figures in its examples are the ministry’s own. So it is not independent of the executive branch. It is independent of the encoding, the encoder, and the administering organization’s later choices, and for a new act it precedes any production system. Its standing comes from the record it enters. The memorandum becomes part of the legislative history that parliament creates when it adopts the bill. A minister defends it

under questioning, and afterwards nobody can rewrite it. Courts read it for that reason, and so can a test.

The memorandum to the health care allowance bill shows the form. It states stylized parameters and three households promised 120, 682 and 0 euro (Figure 1), on the record since September 2004 [11]. Such a collection is a partial oracle in the sense of Barr et al. [2]: defined on a handful of points, it can show divergence there and says nothing elsewhere.

The memorandum is not law. Courts and the administration use it to interpret. It binds no one, and it can be wrong about the bill it explains. The method therefore needs a precedence rule. Ours is: *where an example and the article's text diverge, the text governs*. Legislatures that enact examples inside statutes rank them the other way, letting an example extend the provision it illustrates.² Those examples are law and ours are not, which is why the ranking inverts. An example the enactment overlooks is then evidence that the parliamentary record misdescribes the act, worth reporting whether or not anyone is encoding the statute.

3. A Census of Worked Examples

How much of an oracle the record supplies was a matter of anecdote until now. To measure instead, we queried the full-text index of the official publications for eleven Dutch phrases that introduce a worked example: *rekenvoorbeeld*, *voorbeeldberekening*, *cijfervoorbeeld*, *casuspositie*, *stel dat* and six variations. We searched four collections (Table 1), the oldest of which exists only as OCR'd page images. We fetched every explanatory document the queries returned and scored the text around each hit by two heuristics. One keeps passages with at least three amounts and an outcome word, or tables with six. The second keeps those of them naming a household or person, outside budget documents, and we call its output *citizen-level*.

Table 1 gives the count. 2,415 documents hold 5,068 citizen-level passages. Eighty-five percent of the documents are from 2000 onwards. The examples sit where legislation calculates: income tax and nearly every annual Belastingplan, the allowances, social security, pensions, student finance and housing subsidy.

Not every passage carries parliament's adoption. A memorandum proper holds 649 of the 5,068 passages; a third (1,717) sit in explanatory notes to ministerial regulations and orders in council, which parliament never votes on, and the rest in replies to the parliamentary report (876), ministerial letters, committee reports and written questions (1,345), and 481 in advisory opinions and other papers. Nearly everything outside the memorandum is written by the executive and adopted only in the weak sense that the chamber read it before voting; the exceptions are the committee reports, which record members' questions, and the Council of State's advisory opinions.

A language model (Claude Sonnet 4.6) labels the passages in two strata, those about statutes the corpus encodes, drafts or replaced and a random 500 of the rest, as a complete example, a partial one or none, and extracts their inputs and outcome. To estimate precision, an author re-read 55 passages drawn at random from those the model

²Acts Interpretation Act 1901 (Cth) s 15AD, which since 2011 provides that an example may extend a provision's operation; Legislation Act 2019 (NZ) s 23 is the weaker form, an example does not limit. The practice goes back to Macaulay's Indian Penal Code of 1860.

Table 1. What the harvest found, 31 August 2026. Fetched counts the documents retrieved and scored; documents and passages count the citizen-level cut defined above. The scanned archive starts from 5,081 candidates, of which 2,050 are not explanatory and 96 failed to retrieve, most over the 20 MB cap on page images. Dossiers apply to parliamentary papers only, and the two collections share one, hence 899 and not 900.

Collection	Fetched	Documents	Passages	Dossiers
Kamerstukken (parliamentary papers), 1995–2026	31 873	1 396	2 960	757
Scanned archive, before 1995	2 935	172	208	143
Staatsblad (statute gazette)	1 332	101	187	—
Staatscourant (government gazette)	5 262	746	1 713	—
Total	41 402	2 415	5 068	899

called complete, asking of each: does it state inputs, does it state an outcome, can one be derived from the other, is it a single case. No question asks whether the model was right. Forty-four survive all four questions, a precision of 80 percent (95% CI 68–88). Nine of the eleven failures are not calculations at all, a rate cited in passing or a threshold named without a sum; where inputs and an outcome are both present, 44 of 46 can be derived. A single reader produced this estimate, so it says how good the labels are and not how far two readers would agree.

Of 490 scoreable segments in a seeded random sample, 154 hold a complete example, 31.4% (95% CI 27.5–35.7); over the two strata that is 1,513 non-duplicate examples, 202 about statutes the corpus encodes, or roughly 1,200 and 160 at the precision above, and every share below is of that labelled set. Recall we measured on thirteen dossiers: of the 70 documents holding a citizen-level passage, 30 had never been a candidate, among them the memoranda to the housing-subsidy bills of 1997, 2018 and 2024, the last from the dossier Section 6 mines. Table 1 therefore undercounts by a factor of at least 1.75 (70/40).

4. Mining Examples into Scenarios

Everything here runs on the reference implementation of our position paper [5]: statutes encoded as machine-executable specifications in a restricted declarative format, a deterministic engine, and a public corpus of such encodings.³ Mining must itself be inspectable, so it is published in that repository as a versioned work instruction beside the specifications it tests. The instruction sends the agent to the documents attached to the act: memorandum, government response, amendment notes, ministerial letters. There it looks for the vocabulary of Section 3.

Where the memorandum provides too few examples, the agent must not invent more: “only use what the legislature provided”. An invented scenario smuggles the formalizer’s reading back in under the legislator’s authority. Every mined scenario names the document and passage it transcribes. A mistranscribed expected value corrupts the oracle at its root, so transcription is a fault channel of its own. Extraction is checked mechanically: every value and quote in a record must appear verbatim in the segment it came from, and the numeric reading must agree with the printed one. That catches an invented figure but not

³Reference implementation, corpus and work instructions: <https://github.com/NederlandseDigitaleDienst/regelrecht>; harvested corpus: <https://github.com/MinBZK/regelrecht-corpus>.

```

Stel: drempelinkomen 17 500  standaard premie per verzekerde 1 000
toetsingsinkomen per huishouden 22 000
Verzekerde zonder partner normpremie 880 (4% van 17 500 plus 4% van
(22 000 - 17 500)) standaardpremie 1 000 zorgtoeslag 120 [...]
Verzekerde met partner/niet-verzekerde [...] zorgtoeslag 0

```

```

# Bron: Kamerstukken II 2003/04, 29 762, nr. 3 (memorie van toelichting),
#      sectie "Voorbeeldberekening zorgtoeslag".

Scenario: Verzekerde zonder partner (MvT-huishouden 1, belofte 120)
Given the following parameters:
  | drempelinkomen | 1750000 |
  | standaardpremie | 100000 |
  | toetsingsinkomen | 2200000 |
  | heeft_partner | false |
  | partner_verzekerd | false |
When I evaluate "hoogte_zorgtoeslag" of "test_zorgtoeslag_wetsvoorstel_29762"
Then output "normpremie" equals 88000
Then output "hoogte_zorgtoeslag" equals 12000

Scenario: Verzekerde met partner die geen verzekerde is (MvT-huishouden 3, belofte 0)
# [parameters as above, plus:]
Given the following parameters:
  | heeft_partner | true |
  | partner_verzekerd | false |
When I evaluate "hoogte_zorgtoeslag" of "test_zorgtoeslag_wetsvoorstel_29762"
Then output "hoogte_zorgtoeslag" equals 0

```

Figure 1. Above, the memorandum’s worked example as printed (amounts in euro); below, two of its three households transcribed and stamped with their source, in the bill-era encoding of the allowance act (amounts in eurocent). Identifiers are the statute’s own: threshold income, standard premium, assessed income, has-partner, norm premium. The encoding takes the memorandum’s parameters directly, so the scenario needs none of the seven registry tables the production scenario must supply (Section 8); those scenarios are stamped as encoder-derived. The last scenario transcribes the promised zero and fails against the act as enacted (Section 6).

one copied into the wrong field, for which reading a scenario against its stamped passage is cheap and human; the curated suite has not had that check (Section 6).

Scenarios are written in Gherkin [12], the given–when–then format of behavior-driven development [13]. We chose it because it is the standard notation for such tests, and because the memorandum’s examples already have its shape: a situation, an evaluation, an asserted outcome (Figure 1). A scenario stays readable for the lawyers and policy officers who must judge whether it renders the memorandum faithfully. That judgment is the one human link between parliamentary text and executable test.

5. Drafting Under the Oracle

In the reference implementation the suite holds machine drafting to account. Language models, orchestrated by Claude Fable 5.0, encode statutes article by article under published instructions; they translate and the symbolic engine executes [4,3], so the engine (v0.3.0, schema v0.5.6) computed every scenario outcome reported here. The agent may never change an expected value in a mined scenario, a rule that sits in the instructions and is not yet mechanical, and whatever still fails is reported, not patched.

The memorandum’s role is deliberately asymmetric: authoritative for outcomes, forbidden as a source of conditions. A reverse pass demands that every input, condition and threshold trace to a statutory sentence, and what the logic needs but cannot trace goes to a human as an assumption to confirm. The failure mode both passes guard against

is *toelichting-bleed*: the encoding adopts a concrete criterion from the memorandum although the article states only an open norm. That is the precedence rule of Section 2 applied a second time. The two checks miss different things: the reverse pass cannot see an inverted comparison, the scenarios cannot see an invented condition that still computes the exemplified cases, so human review before adoption stays in the design.

6. One Dossier, Audited

The corpus holds eighteen encoded regulations and five synthetic ones that exercise the format. Its law-validation suite holds 44 active scenarios, run with 44 conformance scenarios that pin how each language construct evaluates; all pass, so a failure in the law-validation suite while the conformance layer is green lies in the encoding or the scenario, not in the interpreter. The audit below returns to one scenario, a low-income single household, which loads seven other regulations.

This suite predates the discipline. Its encoders took the expected values from the encoding, and all but two scenarios are stamped as encoder-derived. What follows applies the discipline backwards: we audited those expectations against the consolidated statutes, and for the one household the memorandum does fix, against the parliamentary record.⁴ Three of the four findings are defects in our own corpus. An expectation copied from the encoding cannot contradict it, so the error survives every run of the suite.

The household promised zero. The memorandum's third household promises an allowance of zero: with one partner uninsured, a single standard premium of 1,000 against a norm premium the memorandum rounds to 1,318 leaves nothing [11]. The change came from the government itself. Its amendment note of 14 December 2004 [14] struck "who is insured" from Art. 2(1), which doubles the standard premium for any partner, and replaced the ministerial discretion of Art. 2(4) with a fifty-percent rule, entitling an insured person with a lawfully uninsured partner to half of what the two would have received had both been insured. On the memorandum's own inputs that is half of 2,000 – 1,318, so 341 and not zero. Nobody revisited the example, and the same department wrote the note and the example, so the memorandum has described the act wrongly since the day it took effect. We encoded both texts and ran all three households (Figure 1). The first two pass on bill and act alike; the third passes on the bill and fails on the act, the engine reporting 34,100 eurocent against the promised zero. Had the examples existed as executable scenarios, re-running them against the amended bill would have caught this in the amendment event that caused it (Section 9).

An expectation pinned to the encoding. That low-income scenario asserts an allowance of 2108.21 euro for a single household. The value reproduces the encoding exactly, and the encoding misreads Art. 2(2–3). The article sets the norm premium at a flat percentage *of the threshold income*, plus a percentage of income above it. The encoding makes two mistakes there. It scales the first term by the household's own income where that income is below the threshold. And what it hardcodes as "threshold income" is the scheme's income ceiling, not the statutory definition (108% of twelve times the January minimum wage, Art. 1(1)(f)). A third error is annotated in the encoding itself: for partners it uses

⁴Statute texts as consolidated on wetten.overheid.nl, 1 January 2025 (allowance act, BWBR0018451); the enactment-era comparison rests on the enacting text (Stb. 2005, 369).

the applicant's income where Art. 2(2) requires the joint income. The first two shrink the norm premium for low incomes and so inflate the allowance. Correcting the first term alone brings the same household to 1358.93 euro, 749 euro below the asserted value. No memorandum states these values. They were read off the encoding and then pinned as its expectations, so no run of the suite could have caught them.

A reverse pass that misfired. Earlier versions of the encoding checked age and excluded detainees in forensic care. The reverse pass found no sentence carrying them and removed both as scope violations, but Art. 1(1)(c) carries both: it defines the insured from the first day of the month after turning eighteen, with an exception for the detainees of Art. 24 of the health insurance act. The pass had walked the corpus's transcription of Article 1, a paraphrase that leaves the definition out, so it checked provenance against a text that was not the law's.

Rounding enacted in the neighboring statute. Rounding happens only where a statutory sentence says so, and the allowance act has none: the 2024 encoding computes an unrounded sub-cent allowance and the suite asserts it (1,972.0531187 euro), while the 2025 encoding rounds to whole eurocent under an engineering convention. The statute's own rule sits in a neighboring act, Art. 14 of the framework act for income-dependent schemes, rounding to whole euros and awarding nothing under 24 euro. The encoding therefore rounded where the law does not and missed the rounding the law prescribes.

These four findings came from one dossier. Running the other examples depends on the encoding rather than the mining: the 1,513 spread over more statutes than the eighteen this corpus holds, and an example cannot be run against a statute nobody has encoded. We therefore tested what the harvest adds three times. For the allowance act's own dossier it holds the answering memorandum to the senate, whose tables promise the allowance for fourteen more households of young workers and welfare recipients [15]. Nine fall below the threshold income, where the outcome does not depend on it, and the engine reproduces all nine; the transcription carries over a premium and a threshold the tables leave unstated, as the scenario file records. Five couples aged nineteen to twenty-three need an income series the document nowhere states and stay untranscribed. Second, two examples for the cost-sharing norm of the Participation Act (Kamerstukken II 2014/15, 34 273, nr. 3). Neither runs: they give 2015 amounts for the young-person norm of Art. 20 and for Art. 22a, and the encoding models neither. Both are committed and marked not runnable, which is the finding: the record is ahead of the corpus. Third, the huurtoeslag dossier the recall check surfaced. The simplified regime took effect on 1 January 2026; the reply to the committee promises three households 355.18, 463 and 315 euro a month while conceding that the parameters for 2025 and 2026 "cannot be shared because they have not yet been set" [16]. We sealed the three examples in a committed file, then encoded the act as it now reads with the two regulations it resolves; all three fail, by -25, +22 and +29 euro a month, 2026 promises computed on 2024 parameters.

7. Measuring a Suite by Seeding Faults

Mutation testing measures a suite's sensitivity: seed small defects and count how many it kills, that is, detects [17,18,19]. The score measures the suite, not the encoding. We seeded 122 mutants across the seventeen files, fourteen regulations at one or two dates,

that have a mutable site: not a random sample but one mutant per site the operators match (swap an operation, flip a boolean, add one to an integer constant). We ran the suite once per mutant and once more per survivor with tracing on. A mutant counts as covered when a scenario executes the mutated operation, established for every survivor by a byte-exact diff of execution traces against the baseline. Twenty-two are not covered. Of the 100 covered, the suite kills 59, all but two on a wrong outcome rather than a runtime type error. Sensitivity tracks the variety of the households rather than their number: the participation act's scenarios kill 13 of the 14 mutants they cover, while the allowance act's near-uniform low-income households leave 11 of 50 uncovered and kill 19 of the 39 they reach.

The encoders wrote the suite under test, which matters for what it catches but not for what the survivors reveal. Its households are mostly single and low income, so the 41 covered survivors show what households of that shape do not distinguish. Eighteen are constants and comparisons no household sits on and no scenario asserts, among them the six-week objection period of Art. 6:7 of the General Administrative Law Act, which could become seven weeks unnoticed; thirteen are sums whose second term is zero on every household, four are conjunctions never falsified one member at a time, and six may be equivalent mutants. Twenty-four of the 41 need a household on a boundary or with a second income component, which the memoranda rarely supply and Section 8 measures.

Holding the encoding fixed lets us compare the two kinds of suite. We corrected the formula defect reported above, re-pinned the twelve encoder-written scenarios to the corrected outcomes, and transcribed the memorandum's households into scenarios whose expected values come from the legislator's arithmetic on the parameters in force. Of 53 mutants seeded into the allowance act and the regulations it resolves, the encoder suite kills 23 and the oracle suite, two households derived from the memorandum's arithmetic, kills 12: nine shared, three by the oracle alone, fourteen by the encoders alone, twenty-seven by neither. The three share a mechanism: all sit in the norm-premium branch, which only incomes above the threshold reach; one oracle household lies above it, the encoders' below.

8. Three Limits of the Oracle

They are shallow. The memorandum of the allowance act states three parameters and a partner status per household. The production scenario that computes the same output loads seven regulations and supplies seven registry tables, about thirty fields. Every input the example leaves out is a default chosen by whoever transcribes it, so the formalizer's judgment re-enters the test. Measured against the encodings' declared inputs, the 202 examples the model called complete about encoded statutes specify on average a third of the statute-level inputs the output needs (0.4 median). Insurance status and the assets test appear in fewer than one example in fifty, the standard premium in one in twenty. Seven percent place an input on a threshold the example itself states, and of all examples that state parameters, 39 percent use hypothetical ones, in the form "suppose a threshold income of 17,500".

They go stale. A memorandum is written once. The Belastingdienst's own handbook for statute analysis says the same of every parliamentary paper after it: they "are never revised" [20]. The amendment note of December 2004 changed Art. 2(4) and left the household it overtook promising zero ever since. Indexation does the same work more

slowly. All 25 stated parameter values we could check differ from the value in force today. Whether a dossier ever revises an example we can count only by proxy. Of the 55 dossiers whose examples appear at two or more stages of passage, 5 restate a household with different numbers, 122 such pairs in all. Each pair is a case where a later stage may have overtaken an earlier example, and nobody checked. A maintained suite would be amended by the same act that amends the article.

They stop at the open norms. The precedence rule of Section 2 forbids the memorandum to supply conditions. So the examples test formulas and rarely concretizations: who counts as a partner, what counts as income, whose savings are the child's. Five percent of the complete examples turn on the classification of a fact under a defined term, and none concerns a policy rule. That label is the model's, and the calibration checked whether a passage is an example, not what it turns on. The layer that decides most cases is the executing organization's own implementation policy (*uitvoeringsbeleid*), and it has no test case in the record. Section 9 gives the two sources that could supply one their own layer.

9. Where a Maintained Suite Belongs

The institutional slot already exists. In its 2021 review of digitalization in legislation and adjudication, the Dutch Council of State added an item to its legislative test: a public record of how proposed regulation becomes automated decision rules, the *specificatiedossier*. Its last question asks how the correct working of those rules is guaranteed, in its words, even as time passes [21]; Lokin's inaugural lecture turns the heading into a design, version-managed with the legislation and maintained as a statutory duty [22]. Neither says what the dossier's executable content is.

That last question is a conformance question, which software answers with a conformance suite. Each gap Section 8 measures names a layer with its own author. Worked examples are the first, adopted by parliament, authoritative for outcomes and silent on conditions. The concretizations that decide most cases belong to the executing organization, whose policy rules and implementation assessments already contain concrete cases, as do the test scenarios and binding reference tables tax administrations publish for third-party software [23,24], outside the record. Court decisions supply the edge cases nobody wrote down in advance, independent of legislator and executor alike. A dossier holding the three, each labelled by source, gives the reader of a decision a case that somebody other than the system's builder fixed. Maintenance follows the law: the note that rewrote Art. 2(4) would have failed the household promised zero in December 2004 rather than two decades later.

What remains open belongs to legislative practice and constitutional lawyers. The drafting directives require a memorandum but no worked examples, so the duty does not exist today, and nobody can say what weight a court would give a published scenario, or which layer governs when a court's case contradicts a policy rule's. Whether it generalizes is open in the same way. It needs a document that accompanies a bill by rule, a habit of putting numbers in it, and a full-text archive; the Netherlands has all three, and the same harvest would answer whether a German *Gesetzesbegründung* computes as often.

10. Related Work

Encoding law for execution runs from TAXMAN [25] and the British Nationality Act [1] via POWER, RegelSprak/ALEF and eFLINT [26,27,28] to production systems abroad [9,8,7]. Validation there uses isomorphism [6], comparison with another encoding [8], and test cases from the encoding project or the administering organization. Wetsanalyse validates its model on scenarios drawn up by a multidisciplinary team [29,30,27]. The question here comes first: where the expected outcomes come from. Witt et al. name “examples in explanatory memoranda” as a source of unit tests for encoded copyright provisions but write none from them [31]. No system we know of makes the record the outcome source or gives it a precedence rule. Better Rules modelled two enacted statutes and generated legislative text and code from the same models [32]. LexImpact runs the encoding for the French parliament on typical households, the reverse direction [33]. Concurrent work compares formalizations by the decisions they encode [34]. On the drafting side, translate-then-execute is a common remedy for model unreliability [3,4], but neither asks where the expected outcomes come from. Mutation testing supplies our instrument [18]; as in specification mutation [19], we mutate the encoding, not the interpreter.

11. Conclusion

Tests of a formalized statute need expected outcomes, and in current practice those come from the encoder. The Dutch parliamentary record supplies them from another source: when the government explains a bill it works out examples, published before any system exists. The harvest found 5,068 candidate passages, at most about half of what is there.

We transcribed seventeen and ran fifteen. Eleven reproduce the promised amount; one of the four failures is a household the government’s own amendment note falsified before the act took effect. Auditing the expectations around it found three defects in the encodings, two that no run of the suite could catch because their expected values came from the encodings they were meant to test. Seeding 53 faults, twelve encoder-written scenarios caught 23 and two built from the memorandum’s arithmetic caught 12; each suite caught faults the other missed, the oracle’s three above the income threshold, where one of its households lies and the encoders’ do not.

An example states a third of the inputs an encoding needs, its parameters go stale, and few concern the concretizations that decide most cases: one layer of the specificatiedossier, beside the policy rules and judgments that carry what they cannot.

Acknowledgements

The statute encodings and the extraction of examples from the parliamentary record were produced with large language models, as described in Sections 3 and 5.

References

- [1] Sergot MJ, Sadri F, Kowalski RA, Kriwaczek F, Hammond P, Cory HT. The British Nationality Act as a Logic Program. *Communications of the ACM*. 1986;29(5):370-86. doi:10.1145/5689.5920.

- [2] Barr ET, Harman M, McMinn P, Shahbaz M, Yoo S. The Oracle Problem in Software Testing: A Survey. *IEEE Transactions on Software Engineering*. 2015;41(5):507-25. doi:10.1109/TSE.2014.2372785.
- [3] Janatian S, Westermann H, Tan J, Savelka J, Benyekhlef K. From Text to Structure: Using Large Language Models to Support the Development of Legal Expert Systems. In: *Legal Knowledge and Information Systems: JURIX 2023*. vol. 379 of *Frontiers in Artificial Intelligence and Applications*. IOS Press; 2023. p. 167-76. doi:10.3233/FAIA230962.
- [4] Pan L, Albalak A, Wang X, Wang WY. Logic-LM: Empowering Large Language Models with Symbolic Solvers for Faithful Logical Reasoning. In: *Findings of the Association for Computational Linguistics: EMNLP 2023*. ACL; 2023. p. 3806-24. doi:10.18653/v1/2023.findings-emnlp.248.
- [5] Hotting E, Schuth A. Rules as Executed: Publishing Machine-Executable Law to Rebalance the Powers; 2026. Working paper, Zenodo. doi:10.5281/zenodo.22201094.
- [6] Bench-Capon TJM, Coenen FP. Isomorphism and Legal Knowledge Based Systems. *Artificial Intelligence and Law*. 1992;1(1):65-86. doi:10.1007/BF00118479.
- [7] Merigoux D, Monat R, Protzenko J. A Modern Compiler for the French Tax Code. In: *Proceedings of the 30th ACM SIGPLAN International Conference on Compiler Construction (CC 2021)*. ACM; 2021. p. 71-82. doi:10.1145/3446804.3446850.
- [8] Merigoux D, Chataing N, Protzenko J. Catala: A Programming Language for the Law. *Proceedings of the ACM on Programming Languages*. 2021;5(ICFP):77:1-77:29. doi:10.1145/3473582.
- [9] OpenFisca. OpenFisca: Write Rules as Code; 2026. Available from: <https://openfisca.org> [cited 2026 Oct 16].
- [10] van Eck M. Geautomatiseerde ketenbesluiten & rechtsbescherming: een onderzoek naar de praktijk van geautomatiseerde ketenbesluiten over een financieel belang in relatie tot rechtsbescherming [PhD thesis]. Tilburg University; 2018. doi:10.2139/ssrn.3422781.
- [11] Tweede Kamer der Staten-Generaal. Regels inzake de aanspraak op een financiële tegemoetkoming in de premie van een zorgverzekering vanwege een laag inkomen (Wet op de zorgtoeslag); Memorie van toelichting; 2004. Kamerstukken II 2003/04, 29 762, nr. 3. Available from: <https://zoek.officielebekendmakingen.nl/kst-29762-3.html>.
- [12] Wynne M, Hellesøy A. The Cucumber Book: Behaviour-Driven Development for Testers and Developers. Dallas, TX: Pragmatic Bookshelf; 2012.
- [13] North D. Introducing BDD; 2006. Available from: <https://dannorth.net/blog/introducing-bdd/>.
- [14] Tweede Kamer der Staten-Generaal. Wet op de zorgtoeslag; Nota van wijziging; 2004. Kamerstukken II 2004/05, 29 762, nr. 18. Available from: <https://zoek.officielebekendmakingen.nl/kst-29762-18.html>.
- [15] Eerste Kamer der Staten-Generaal. Wet op de zorgtoeslag; Memorie van antwoord; 2005. Kamerstukken I 2004/05, 29 762, D. Available from: <https://zoek.officielebekendmakingen.nl/kst-20042005-29762-D.html>.
- [16] Tweede Kamer der Staten-Generaal. Wijziging van de Wet verlaging eigen bijdrage huurtoeslag, de Wet op de huurtoeslag en enkele andere wetten ter verbetering van de koopkracht en vereenvoudiging van de regeling; Nota naar aanleiding van het verslag; 2024. Kamerstukken II 2024/25, 36 608, nr. 6. Available from: <https://zoek.officielebekendmakingen.nl/kst-36608-6.html>.
- [17] DeMillo RA, Lipton RJ, Sayward FG. Hints on Test Data Selection: Help for the Practicing Programmer. *Computer*. 1978;11(4):34-41. doi:10.1109/C-M.1978.218136.
- [18] Jia Y, Harman M. An Analysis and Survey of the Development of Mutation Testing. *IEEE Transactions on Software Engineering*. 2011;37(5):649-78. doi:10.1109/TSE.2010.62.
- [19] Budd TA, Gopal AS. Program Testing by Specification Mutation. *Computer Languages*. 1985;10(1):63-73. doi:10.1016/0096-0551(85)90011-6.
- [20] Belastingdienst, Expertisecentrum Business Rule Management. Handleiding Wetsanalyse in de Praktijk. Belastingdienst; 2023. Versie 1.0. Available from: <https://wendbarewetsuitvoering.pleio.nl/attachment/805680a2-1c37-4831-bbfb-2d602ca7d65c>.
- [21] Raad van State. Digitalisering: wetgeving en bestuursrechtspraak. Den Haag: Raad van State; 2021. Available from: <https://www.raadvanstate.nl/@125918/publicatie-digitalisering/>.
- [22] Lokin M. Digitaal disciplineren anno 2026: hoe houdt de wetgever grip op digitale uitvoering? [Inaugural lecture]. Heerlen: Open Universiteit; 2026. Oratie, leerstoel Wetgeven in de digitale rechtsstaat. doi:10.71583/20260612mhaf.
- [23] Internal Revenue Service. Assurance Testing System (ATS) Guidelines for Modernized e-File (MeF)

Individual Tax Returns for Tax Year 2026. Internal Revenue Service; 2026. Publication 1436 (Rev. 10-2026). Available from: <https://www.irs.gov/pub/irs-pdf/p1436.pdf>.

- [24] Belastingdienst. Rekenvoorschriften voor de geautomatiseerde loonadministratie 2026; 2026. Uitgave januari 2026, versie 2. Available from: https://download.belastingdienst.nl/belastingdienst/docs/rekenvoorschriften_voor_geautomatiseerde_loonadministratie_lh991z62fd.pdf.
- [25] McCarty LT. Reflections on TAXMAN: An Experiment in Artificial Intelligence and Legal Reasoning. *Harvard Law Review*. 1977;90(5):837-93. doi:10.2307/1340132.
- [26] van Engers TM, Gerrits R, Boekennoogen M, Glassée E, Kordelaar P. 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: ACM; 2001. p. 157-67. doi:10.1145/383535.383554.
- [27] Corsius M, Hoppenbrouwers S, Lokin M, Baars E, Sangers-Van Cappellen G, Wilmont I. RegelSprak: A CNL for Executable Tax Rules Specification. In: *Proceedings of the Seventh International Workshop on Controlled Natural Language (CNL 2020/21)*. Special Interest Group on Controlled Natural Language; 2021. Available from: <https://aclanthology.org/2021.cnl-1.6/>.
- [28] van Binsbergen LT, Liu LC, van Doesburg R, van Engers TM. 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; 2020. p. 124-36. doi:10.1145/3425898.3426958.
- [29] Lokin M. *Wendbaar wetgeven: de wetgever als systeembeheerder* [PhD thesis]. Vrije Universiteit Amsterdam; 2018.
- [30] Ausems A, Bulles J, Lokin M. *Wetsanalyse: voor een werkbare uitvoering van wetgeving met ICT*. Den Haag: Boom juridisch; 2021.
- [31] Witt A, Huggins A, Governatori G, Buckley J. Converting Copyright Legislation into Machine-Executable Code: Interpretation, Coding Validation and Legal Alignment. In: *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Law (ICAIL '21)*. ACM; 2021. p. 139-48. doi:10.1145/3462757.3466083.
- [32] Service Innovation Lab. *Better Rules for Government Discovery Report*. New Zealand Government, Department of Internal Affairs; 2018. Available from: <https://www.digital.govt.nz/dmsdocument/95-better-rules-for-government-discovery-report/html> [cited 2026 Oct 16].
- [33] Assemblée nationale. *LexImpact*; 2020. Available from: <https://leximpact.an.fr> [cited 2026 Oct 16].
- [34] Vernie J, Grabmair M. By Their Fruits You Will Know Them: Comparing Formalizations of Law by the Decisions They Encode; 2026. arXiv:2605.25186. Accepted at EMNLP 2026. Available from: <https://arxiv.org/abs/2605.25186>.