FORMAL METHODS / DECISION SYSTEMS

From tax law to checked arithmetic

A source-mapped Lean study of the 2026 Single and Married Filing Jointly ordinary-income schedules, with fourteen theorems and a 224,381-case differential audit

Revanth Matha · Matha Labs | Research whitepaper 3.0 | Revised 16 November 2026

CENTRAL CONTRIBUTION

A source-to-rule map placing the executed schedule within sections 61, 62, 63, 7703 and 1, and distinguishing statutory definitions from annual parameters and filing methods.

The existing model computes a guarded 2026 ordinary taxable-income rate-schedule illustration for reviewed Single or Married Filing Jointly inputs. It does not derive taxable income, determine filing eligibility, implement Form 1040 Tax Tables or special methods, or establish an individual return liability. The new arithmetic audit expands empirical checking without expanding that legal scope.

Original publication: 15 November 2026 at 3:12:52 PM PST. This edition expands the paper; it does not claim a new trained model or a new clinical, municipal or financial deployment.

Contents

Abstract 3

Contributions 3

1. What must a formal tax result mean? 4

2. The legal computation before and after the schedule 4

3. Filing status and the information carried by a number 5

4. From authority to a frozen executable bundle 6

5. The two 2026 schedules, including their cumulative bases 7

6. Deriving a conventional occupied-band calculation 8

7. Why the Lean positive-increment formula represents the same schedule 9

8. Exact units, floor error, and numerical conventions 11

9. The exact input types, certificate, and twelve-part guard 12

10. Fourteen proved statements, stated precisely 13

11. Worked amounts, marginal rates, and average rates 16

12. The ordinary schedule versus IRS Tax Tables and worksheets 17

13. What the original thirty fixtures establish 18

14. Reproducible differential arithmetic audit 18

15. What established computational-law and tax-benefit systems contribute 20

16. A modular path from a schedule to a justified tax computation 21

17. Versioning legal sources, model assumptions, and evidence 22

18. What should be tested next, and what would count as progress? 23

References 24

Abstract

A useful tax formalization must connect an executable calculation to legal definitions, annual authority, numerical conventions, and a justified domain of inputs. This paper develops that connection for an existing Lean 4 model of the 2026 federal ordinary taxable-income rate schedules for Single and Married Filing Jointly. We locate the model within the statutory chain from gross income through adjusted gross income, taxable income, filing status, method selection, and tax adjustments. We map the selected Revenue Procedure 2025-32 tables to fixed thresholds and derive both an occupied-band representation and the implemented positive-increment formula. The latter computes an exact numerator in hundredths of a cent before one explicitly chosen final floor. Fourteen existing Lean theorems establish guard properties, monotonicity, bounds, zero behavior, and a discrete one-cent increment bound; thirty existing synthetic fixtures check amounts and refusals. A new differential audit compares actual compiled Lean outputs with exact occupied-band and published worksheet references on 224,381 distinct synthetic cases, with zero numerator or final-cent mismatches. This finite agreement remains separate from universal proof and independent legal validation. We analyze why the annual rate schedule alone cannot reproduce Form 1040 Tax Tables, preferential-rate methods, credits, or alternative minimum tax. Comparisons with Catala, Tax-Calculator, and OpenFisca identify complementary goals in legal structure, microsimulation, and versioned rules. The resulting contribution is a source-auditable ordinary-income arithmetic component with clearly identified trust assumptions. Further work is specified in terms of formal equivalence, independently reviewed legal translations, typed income derivation, method selection, and applicability experiments rather than an unsupported claim of complete tax-code coverage.

Keywords: Lean 4, federal income tax, 2026 rate schedules, exact arithmetic, formal verification, policy versioning, research illustration

Contributions

1. A source-to-rule map placing the executed schedule within sections 61, 62, 63, 7703 and 1, and distinguishing statutory definitions from annual parameters and filing methods.

2. A derivation of occupied-band and positive-increment schedule representations, with exact units, floor-error analysis, and source-checked constants for both represented filing statuses.

3. An exact inventory of fourteen existing Lean theorem statements and thirty original synthetic fixtures, plus an executed 224,381-case audit finding zero differences between compiled Lean and two differently structured exact references.

4. A comparison with established computational-law and tax-benefit systems and a concrete modular research program for legal translation, applicability, versioning and full-method validation.

1. What must a formal tax result mean?

The arithmetic at the center of a progressive tax schedule is short. The specification around it is not. A dollar figure can refer to receipts, wages, gross income, adjusted gross income, taxable income, tax before credits, total tax, a payment, or a balance due. These quantities occupy different positions in a legal computation. A program that applies the right percentages to the wrong quantity produces a precise answer to a different question. Our starting point is therefore a named input quantity and an explicit location in the tax computation, rather than a promise to automate every decision associated with filing.

The implemented question is: given reviewed ordinary taxable income, a supported filing-status label, tax year 2026, and a matching rule certificate, what amount does the selected ordinary-income rate schedule produce under a declared numeric convention? This is narrower than determining federal tax liability, but it supports substantive research. We can derive the represented function from published rows, state quantified properties, test different implementations against one another, and identify the legal assumptions that must be supplied by surrounding modules. The code and the authoritative source both remain visible to a reviewer. [1], [12]

Three kinds of evidence must be kept separate. A proof establishes a proposition about a formal model. A differential test establishes agreement on its tested domain between implementations that may share assumptions. A source review assesses whether the model expresses the relevant rule for the intended facts. None subsumes the other two. Lexifina's evidence-bearing approach to formal legal models offers a related architectural motivation, but its legal corpus and evaluation are not reproduced here. This paper develops the tax-specific chain of definitions, calculations, and evaluation obligations instead. [11], [14]

2. The legal computation before and after the schedule

Section 61 begins from a broad definition of gross income, subject to other provisions, and enumerates categories such as compensation, business income, gains, interest and rents. Those categories already require facts and legal classification. An amount on a bank statement is not automatically a complete gross-income determination. A future module would need to preserve source transactions, inclusion or exclusion decisions, tax periods, and the authority supporting each treatment. This implementation begins after that work and makes no assertion about a record's completeness. [15]

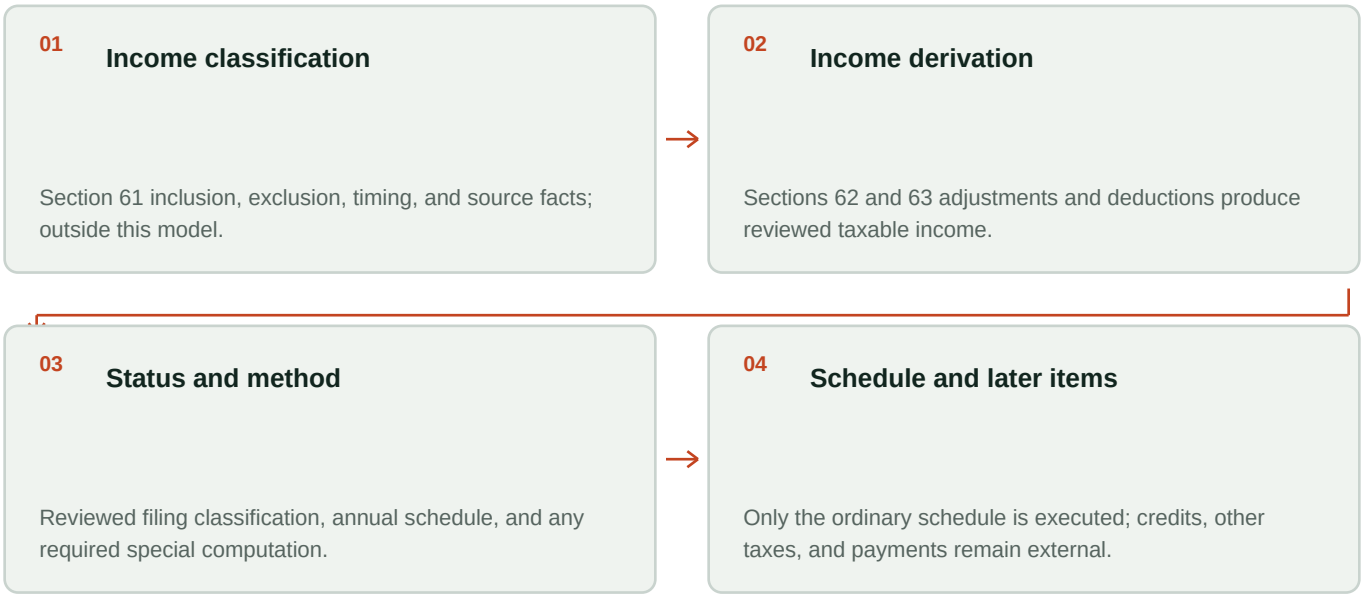
Section 62 defines adjusted gross income through specified deductions from gross income. Section 63 defines taxable income with different structures for itemizing and nonitemizing individuals, including statutory deductions and conditions. The chain is consequently not universally equivalent to wages minus one standard deduction. In particular, the current section 63 text contains more dependencies than that shortcut expresses. Rather than implementing a partial shortcut and calling it taxable income, this model requires the final ordinary taxable-income amount as an externally reviewed input. [4], [16]

Filing classification and method choice cut across that chain. Marital status is determined through section 7703, while joint-return rules are addressed separately in section 6013. The amount then enters an applicable section 1 schedule or another required tax method. Credits, alternative minimum tax, other taxes, payments and refundable amounts can affect later results. These are separate dependencies, not small formatting adjustments to the displayed bracket output. The sequence is a conceptual dependency map; it is not a

complete order of operations for every return. [3], [5], [6], [17]

This placement also clarifies what a future evidence packet must establish. It should identify which quantity has been computed, the facts and elections used, the period and status under which it was derived, and the methods that remain applicable. Passing only a number named income discards the very distinctions that a formal system should preserve. A trustworthy larger architecture would make those distinctions into types and dependencies rather than rely solely on a review Boolean.

Figure 1. The schedule occupies one stage of a larger legal computation



Conceptual dependency map, not a full Form 1040 algorithm. The implemented boundary begins with already reviewed ordinary taxable income and a supported status. [3], [4], [5], [6], [15], [16], [17]

3. Filing status and the information carried by a number

The code supports single and marriedJoint, with a third unsupported constructor. A label is not a determination of legal eligibility. Section 7703 includes rules concerning the close of the taxable year, a spouse's death, legal separation, and certain married individuals living apart. Section 6013 addresses joint returns and associated conditions. A complete status classifier would require more than a marital checkbox, and a joint schedule should not be chosen merely because two people live in the same household. This release treats the classification as reviewed upstream. [5], [17]

A second classification separates ordinary taxable income from income requiring preferential or special treatment. Section 1(h) concerns capital gains, and the IRS estimated-tax materials identify cases that require additional worksheets. A single total taxable-income amount may therefore be insufficient to select the right computation. The ordinaryOnly and specialComputation fields summarize that issue for this small artifact; they do not inspect transactions, calculate a preferential-rate component, or verify that an omitted method is

unnecessary. [2], [3], [7]

The natural-number input also needs an explicit external interpretation. Zero is a valid represented amount, but it is not interchangeable with missing, disputed, negative, or not-yet-derived income. A parser that turns a negative number or failed extraction into zero would erase a material distinction before Lean receives the data. The current repository has no general record-ingestion interface, signed-money type, or evidentiary derivation of the input amount. A future adapter should return a typed error or unresolved quantity for such cases, not manufacture a successful zero-income illustration.

These issues explain why the most important larger-system question is often whether an input is justified, not whether multiplication is deterministic. The finite constructor set is useful for defining a narrow domain and refusing everything else. It becomes misleading only if a surrounding application treats an accepted constructor as proof that the corresponding statutory conditions were satisfied. Formalizing those conditions would be a separate contribution with its own source mapping and tests.

4. From authority to a frozen executable bundle

The numerical authority selected here is Revenue Procedure 2025-32, section 4.01, Table 1 for the joint schedule and Table 3 for the unmarried schedule used for Single. Publication 505 and the 2026 Form 1040-ES materials provide official cross-checks of the schedule constants. They are corroborating publications, not three independent legal authorities or three independent implementations. Their agreement helps detect copying mistakes but does not by itself resolve every later amendment or applicability question. [1], [2], [7], [9]

Section 1 contains historical opening tables, later statutory provisions, inflation adjustments, and special-rate rules. A naive scraper taking the first dollar figures visible on the page would not recover the selected 2026 thresholds. The source-to-rule process must identify the provision, annual parameter publication, applicable year, category, and units. The model records the statute identifier 26-USC-1-j and an exact revision string, but it does not perform this legal selection autonomously. [3], [12]

The distinction among dates is important. A publication date identifies when a source appeared; a taxable-year rule identifies which period it governs; an access date identifies when an investigator reviewed it; and a code commit identifies the executable translation. November 16, 2026 is this paper's source-review date. The model's tax-year condition remains 2026, and edition 3.0 of this paper does not silently change the original rate bundle. Later annual schedules must receive their own source and review records rather than overwrite a historical replay. Revenue Procedure 2025-32 states its Code context as of October 9, 2025, while its bulletin publication is November 3, 2025; those are not interchangeable with the current review date or an HTTP modification timestamp. [1]

The following map separates implemented transformations from surrounding dependencies. Its purpose is to make an audit actionable: a reviewer can point to a source row, its converted constant, the function that uses it, and the evidence that checks it. The table also shows why a correct theorem about a constant does not prove that the investigator selected the legally appropriate constant in the first place.

Table 1. Source-to-rule map and evidence status

Source or dependency	Executable representation	Review or validation boundary
Rev. Proc. 2025-32 section 4.01 Tables 1 and 3	single2026 and joint2026 threshold constants in cents	Official dollar rows are transcribed; twelve bracket bases are checked separately.
26 U.S.C. section 1(j) and annual schedule selection	Compiled statuteId, revision, source URL and taxYear=2026	Identifier equality does not establish a complete legal applicability analysis.
Published marginal percentages	Positive rate increments 10,2,10,2,8,3,2	Algebraic derivation and differential arithmetic audit; no existing formal source-equivalence theorem.
Sections 61, 62 and 63 income definitions	Already reviewed ordinaryTaxableIncomeCents	Derivation, deductions, exclusions and elections are not executed.
Sections 7703 and 6013 status requirements	Reviewed single or marriedJoint constructor	Statutory status eligibility is not proved by the constructor.
Tax Table or special-method requirements	ordinaryOnly and exclusion flags in Conditions	Flags are reviewed assumptions; method selection and special calculations are absent.
Chosen output convention	Exact numerator, then division by 100	Final floor to cents is a research convention, not a filing-rounding instruction.

Primary sources [1], [3], [4], [5], [15], [16], [17], [18] establish the legal context; the pinned model and fixtures establish what was actually translated. [12], [13]

5. The two 2026 schedules, including their cumulative bases

Both represented schedules use marginal rates of 10%, 12%, 22%, 24%, 32%, 35% and 37%. The next table includes cumulative tax at each lower boundary, not just thresholds. This matters because a base-plus-excess row combines the tax on every completed band with the marginal tax on the remaining income. Copying only a threshold vector leaves that accumulation implicit. Source-derived base amounts provide a useful cross-check that is sensitive to a wrong intermediate threshold or rate. [1]

The values in the table are dollars for readability. Every code threshold is multiplied by 100 to represent cents. Thus the first Single boundary is 1,240,000 cents, not 12,400 cents. The schedule's upper end is open: the last rate applies to the excess above its lower boundary. The repository accepts arbitrary natural-number cents, although the new finite audit necessarily examines a bounded collection of them. [12]

The joint vector is not obtained simply by doubling the Single vector. Its earlier thresholds match such a scaling, but its final boundary does not. Reusing one schedule with an unconditional two-person multiplier would therefore create a predictable high-income error. The implementation stores both vectors explicitly. This observation concerns the represented arithmetic, not an assessment of any household's legal filing choice or

overall tax burden.

Table 2. 2026 marginal bands and cumulative tax at their lower boundaries

Rate	Single lower bound	Single base tax	Joint lower bound	Joint base tax
10%	$0	$0	$0	$0
12%	$12,400	$1,240	$24,800	$2,480
22%	$50,400	$5,800	$100,800	$11,600
24%	$105,700	$17,966	$211,400	$35,932
32%	$201,775	$41,024	$403,550	$82,048
35%	$256,225	$58,448	$512,450	$116,896
37%	$640,600	$192,979.25	$768,700	$206,583.50

A row applies to income above its displayed lower bound, with the first row beginning at zero. Constants and cumulative bases are from Rev. Proc. 2025-32 section 4.01; they are ordinary schedule values rather than Form 1040 liability. [1]

6. Deriving a conventional occupied-band calculation

Let b0=0 and b1 through b6 be the increasing lower boundaries of the seven bands, measured in cents. Let r0 through r6 be their percentage rates. For each bounded band, the occupied width at income x is the smaller of its full width and the positive part of x minus its lower boundary. The final band uses the entire positive excess above b6. Summing rate times occupied width yields an exact numerator whose denominator is one hundred. No rounding is needed during band accumulation.

This formulation has a direct interpretation. If x lies in band k, all lower bands are fully occupied, the kth band is partially occupied, and every higher band is empty. The exact contribution of the kth band is rk times the excess above bk. The sum of completed lower bands is the cumulative base printed in the official row. A base-plus-excess implementation and an occupied-band implementation should therefore agree when they use the same ordered boundaries, rates, units, and endpoint convention. [1]

The conceptual derivation is straightforward, but it exposes useful test obligations. A boundary must not be charged twice; an unoccupied band must contribute zero; the final band must not be capped at a nonexistent upper endpoint; and rates must be percentages rather than raw multipliers. An implementation should also distinguish cents of income from dollars of income before applying the denominator. Boundary and scale tests are designed to catch these specific errors rather than merely collect a large random sample.

The new comparator uses this band-occupation structure with exact rational arithmetic. The Lean model deliberately uses a different positive-increment structure. Structural diversity reduces the chance that both programs share the same control-flow mistake, while shared constants and author review remain potential

common causes of error. Agreement is useful evidence of arithmetic consistency, not automatic independence of specification.

```
N_band(x) = sum(i=0..5) r_i * min(max(x-b_i,0), b_(i+1)-b_i) + r_6 * max(x-b_6,0)
```

x and the ordered b values are cents; r values are integer percentage points. N_band is hundredths of a cent. This is the mathematical form used by the differently structured reference, not the source form of the Lean numerator.

7. Why the Lean positive-increment formula represents the same schedule

Define the rate changes after the first band as the difference between consecutive marginal rates. The published sequence gives changes of 2, 10, 2, 8, 3 and 2 percentage points after the initial 10. Charge the initial rate on all x, then add each change only to the positive excess above the boundary where it begins. This produces the implemented expression without a branch selecting a single row. Natural-number subtraction implements each positive excess exactly. [12]

To see the algebra, suppose x lies between bk and the next boundary. All excesses above later boundaries vanish. The coefficient of x in the remaining terms is r0 plus the first k rate changes, which telescopes to rk. The constant term is the sum of each active rate change multiplied by its boundary, subtracted from rk times x. Evaluating that expression at bk yields the accumulated lower-band tax; increasing x within the band then adds exactly rk times the change. This reconstructs the corresponding base-plus-excess row. The argument requires the intended ordered boundary interpretation.

This paper supplies that mathematical derivation and tests the concrete translations, but the existing Lean theorem inventory does not contain a formal equivalence theorem to the occupied-band function or the IRS publication. That is an important open proof obligation. The present universal properties quantify over arbitrary Thresholds records, including incorrectly ordered or transcribed ones. They establish useful shape constraints without establishing that a record is an authoritative schedule.

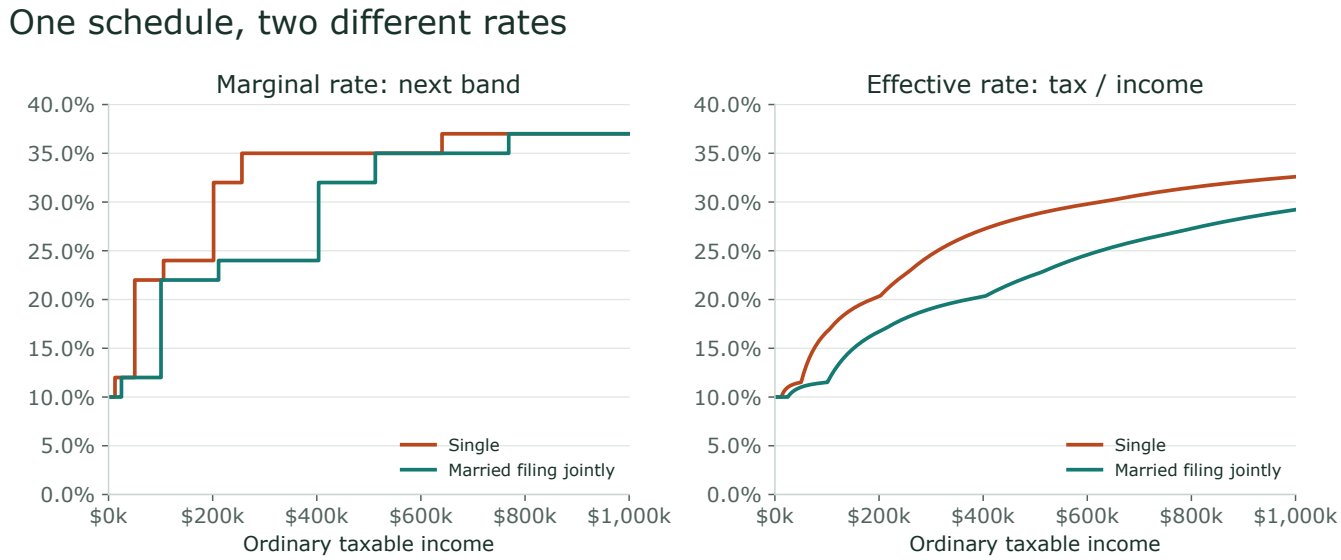
Publication 505 Worksheet 1-4 supplies a third algebraic form: multiply the amount by the applicable band rate and subtract a published constant. For the Single 22% row, that constant is $5,288; at $100,000, 22% times the input minus $5,288 again gives $16,712. In general the subtraction constant is the sum of active rate increments times their boundaries, divided by one hundred in matching units. The new audit transcribes the published constants separately rather than generating them from Model.lean. Worksheet 1-4(a) is Single and 1-4(c) is Married Filing Jointly. [2], [23]

The representation is particularly convenient for monotonicity. Every excess is monotone, every coefficient is nonnegative, and the sum preserves order. It also bounds the exact numerator by 37x because every excess is at most x and the coefficients sum to 37. The model's simplicity is a strength for auditability, provided that these arithmetic properties are not presented as a substitute for checking the annual parameters and legal domain.

```
E_i(x)=max(x-b_i,0); N(x)=10*x+2*E_1(x)+10*E_2(x)+2*E_3(x)+8*E_4(x)+3*E_5(x)+2*E_6(x)
```

Each E_i is the positive excess above a threshold, implemented by natural-number subtraction. Rates accumulate to 10,12,22,24,32,35,37 percent. x and the boundaries are cents; N is hundredths of a cent, and the output is floor(N/100).

Figure 2. Marginal and effective rates in the represented schedules



2026 Single and MFJ ordinary schedules only. No deductions, credits, special rates or return-method selection.
Source: IRS Rev. Proc. 2025-32, section 4.01. Mathematical illustration, not observed taxpayer data.

Derived from the fixed 2026 ordinary schedules, over the plotted income domain. Marginal rate describes the underlying unrounded schedule; the final cent-valued implementation is a staircase. Effective rate means schedule tax divided by taxable income, not a household burden after credits or other taxes. [1], [12]

Table 3. Published worksheet subtraction constants in dollars

Rate	Single subtraction	Joint subtraction
10%	$0	$0
12%	$248	$496
22%	$5,288	$10,576
24%	$7,402	$14,804
32%	$23,544	$47,088
35%	$31,230.75	$62,461.50
37%	$44,042.75	$77,835.50

Publication 505 (2026), Worksheet 1-4(a) and (c). Within the chosen band, exact schedule tax is rate times taxable income minus the displayed constant. These estimated-tax worksheet expressions are an arithmetic oracle, not a full return calculation. [2], [23]

8. Exact units, floor error, and numerical conventions

A percentage-point coefficient multiplied by income cents produces a numerator in hundredths of a cent. The implementation retains that numerator as a natural number and divides by 100 exactly once. It neither uses binary floating point nor rounds each band's contribution. This is a complete numerical convention for the represented function: a caller can reproduce it without guessing how intermediate fractions were handled. It is nevertheless an authored research convention, not an inference that the IRS requires floor-to-cent output for every return calculation.

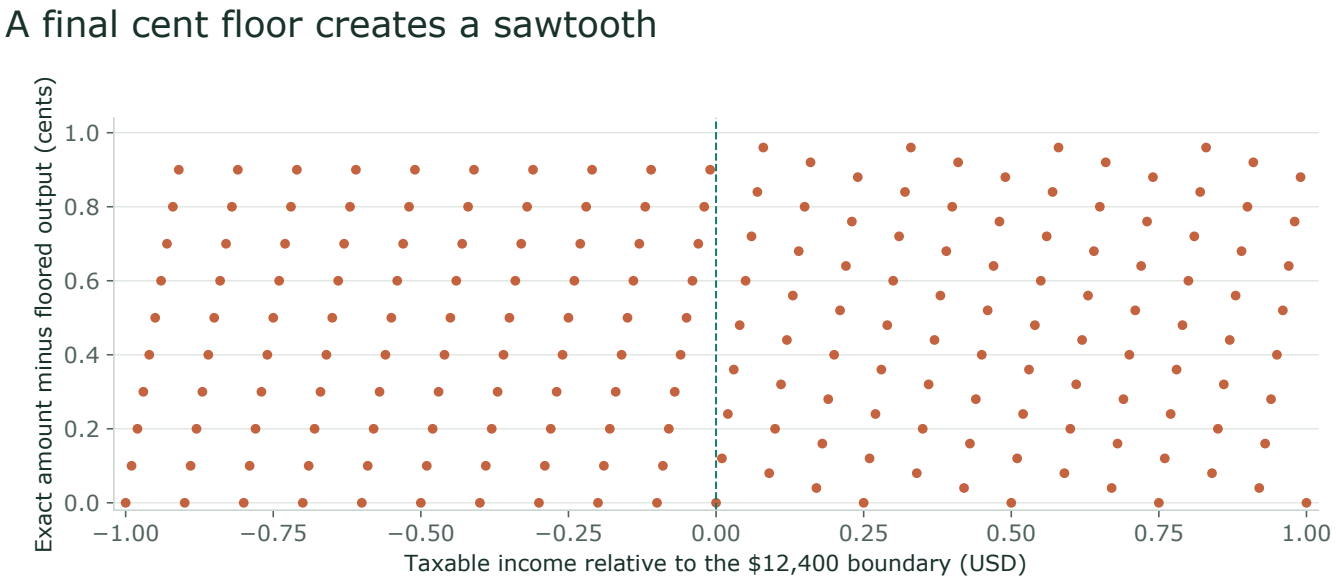
Write the exact numerator as $100q+r$ with integer remainder r between zero and ninety-nine. The returned amount is q cents and the omitted fraction is $r/100$ of a cent. The error relative to the exact unrounded schedule is therefore nonnegative and strictly less than one cent. It is downward rather than nearest-cent rounding. This bound is a mathematical consequence of the convention; the original fourteen named theorems do not include a separately named remainder-error theorem. The distinction prevents an explanatory derivation from being counted as a new formal result.

The first Single boundary illustrates the units. At $12,399.99, the exact tax is $1,239.999 and the result is $1,239.99. At $12,400.01, the exact tax is $1,240.0012 and the result remains $1,240.00. At $12,400.09, it is $1,240.0108 and the result is $1,240.01. Rounding each band, rounding each input dollar, or consulting an IRS Tax Table would implement different functions. A comparator must intentionally select the same final convention before interpreting a discrepancy as an arithmetic bug.

Natural numbers also provide unbounded mathematical precision rather than a fixed-width monetary type. This avoids a modeled integer-overflow limit but does not imply unlimited practical input size or runtime. A production data boundary would still need resource limits, exact decimal parsing, and a policy for unsupported fractional cents. Those operational constraints have not been inferred from the arithmetic proofs.

The floor preserves amount monotonicity but does not justify every property of the unrounded curve. For example, at ten cents of income the output is one cent, an average ratio of 10%; at eleven cents it remains one cent, about 9.09%. Thus a claim that the floored effective ratio is monotone would be false. Convexity or marginal-rate claims in the graphics refer to the underlying unrounded schedule, not an unproved convexity property of the cent-valued sequence.

Figure 3. The final floor omits less than one cent



201 adjacent cent-valued inputs around the first Single threshold. The model floors once, after exact accumulation. This research rounding convention is not a claim about the amount required on Form 1040.

Illustration of the declared final-floor convention, using the fixed schedule rather than any IRS Tax Table. This is numerical analysis, not a measured filing error or a claim about required IRS rounding. [12]

9. The exact input types, certificate, and twelve-part guard

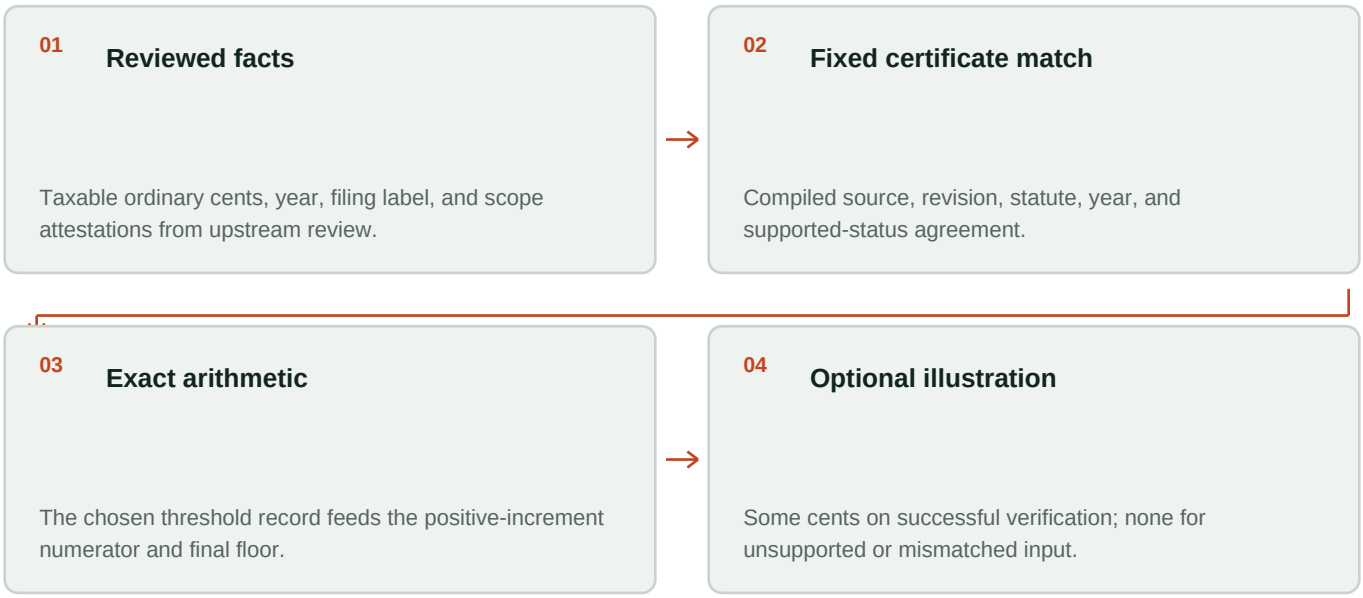
ReviewedFacts contains taxYear, FilingStatus, ordinaryTaxableIncomeCents, reviewed, ordinaryOnly, hasCredits, requiresAMT, and specialComputation. Certificate contains sourceId, revision, statuteId, taxYear, and status. Conditions has twelve conjuncts: the three compiled identity strings must match; the facts year must be 2026; certificate year and status must agree with the facts; status must be supported; reviewed and ordinaryOnly must be true; and the three exclusion flags must be false. verify decides this proposition. evaluate returns an optional amount through that guard. [12]

The source URL, revision, and statute label are code-owned constants. A certificate cannot choose a different table by supplying a convincing explanation. However, equality with a URL is not proof that the source bytes were fetched, authenticated, archived, or correctly translated. The ordinaryOnly and exclusion flags likewise summarize reviewed judgments; their truth values do not establish the absence of capital gains, credits, AMT, or another applicable worksheet. A future evidence design would need to connect those judgments to typed facts and review records.

The total thresholds function has a Single fallback for unsupported status. That is a convenience for defining a total internal function, not an accepted filing path. unsupported_status_refused and the guarded evaluator prevent that fallback from becoming a numeric result for an unsupported input. Directly calling the lower-level arithmetic on a chosen threshold record bypasses applicability by design. It is useful for proof and test harnesses but must not be confused with calling evaluate.

Failure currently returns none without a structured reason. This allows an interface to distinguish a guarded illustration from no illustration, but it cannot truthfully attribute a specific cause to evaluate's return value alone. A separate explanation layer can inspect the same predicate; it should be tested for consistency and preserve multiple failing conditions rather than inventing a single confident narrative. Distinguishing refusal from zero is especially important: some 0 is a computed in-scope amount, whereas none means that this model did not provide one.

Figure 4. What the implemented gate accepts and what it assumes



Executed dataflow of the original Lean model. Fact derivation, source authentication, special-method detection, and explanatory rejection reasons are not supplied by this gate. [12]

10. Fourteen proved statements, stated precisely

The following table records the actual theorem names and statements at the original implementation commit. The first eight concern the relationship between supplied records and the gate. The remaining six concern the arithmetic for all natural-number inputs and threshold records. This separation makes the proof boundary inspectable: a property of record agreement does not establish legal eligibility, and a property of a total arithmetic function does not establish whether a taxpayer should use it. [12]

The model contains no admitted proof, unsafe declaration, or project-specific axiom. Lean still uses its ordinary logical foundations and an implementation environment; proof checking is not a claim that every component of an operational system has been verified. The source pins Lean 4.35.0. Auditing the statements remains as important as checking that their proofs compile, because a statement can be weakened or omit a crucial premise without becoming logically false. [11]

Two limits deserve emphasis. numerator_bounded_by_top_rate establishes N(x)<=37x; illustration_le_income states a looser bound on the rounded amount, not the tightest possible expression of the top rate. one_cent_increase_bounded is a discrete output bound, not a theorem about disposable income after credits or benefits. Monotonicity would remain true under many incorrect threshold vectors. The constants, source selection, and complete-method applicability therefore require evidence beyond this theorem inventory.

A concrete countermodel shows the distinction. Raise only the second marginal band from 12% to 13%, then use increments 10,3,9,2,8,3,2 so the later marginal rates remain unchanged. The resulting function is still monotone, starts at zero, remains bounded by the same top coefficient, and has the same one-cent increment bound. Yet at $100,000 Single taxable income it overstates the intended schedule by $380. This algebraic counterexample is not a newly executed fixture; it explains why source-derived constants and a differential comparison add evidence that generic shape invariants cannot supply.

Table 4. Exact existing Lean statements

Name	Formal statement in readable notation	Meaning
verification_iff	verify(f,c)=true iff Conditions(f,c)	Boolean success exactly matches the declared gate.
accepted_source_matches	verify(f,c)=true implies c.sourceId=sourceId	An accepted certificate matches the compiled source string.
accepted_year_is_2026	verify(f,c)=true implies f.taxYear=2026	The accepted facts name the represented year.
accepted_status_matches	verify(f,c)=true implies c.status=f.status	Certificate and reviewed status agree.
accepted_requires_review	verify(f,c)=true implies f.reviewed=true	An upstream review attestation is required.
accepted_is_ordinary_only	verify(f,c)=true implies f.ordinaryOnly=true	The supplied ordinary-only assertion is required.
unsupported_status_refused	f.status=unsupported implies verify(f,c)=false	The explicit unsupported constructor is refused.
rejected_has_no_illustration	verify(f,c)=false implies evaluate(f,c)=none	No amount through the guarded wrapper after rejection.
numerator_monotone	x<=y implies numerator(t,x)<=numerator(t,y)	Exact numerator is monotone for arbitrary thresholds.
illustration_monotone	x<=y implies illustrationCents(t,x)<=illustrationCents(t,y)	Final-floor result is monotone.
numerator_bounded_by_top_rate	numerator(t,x)<=37*x	Exact numerator has a universal coefficient bound.
illustration_le_income	illustrationCents(t,x)<=x	The rounded amount cannot exceed represented income.
zero_income_zero_illustration	illustrationCents(t,0)=0	Zero input has zero arithmetic output.
one_cent_increase_bounded	illustrationCents(t,x+1)<=illustrationCents(t,x)+1	One extra income cent adds at most one output cent.

f is ReviewedFacts, c Certificate, t Thresholds, and x,y natural-number cents. The table adds no new theorem, legal completeness claim, or formal equivalence result. [12]

11. Worked amounts, marginal rates, and average rates

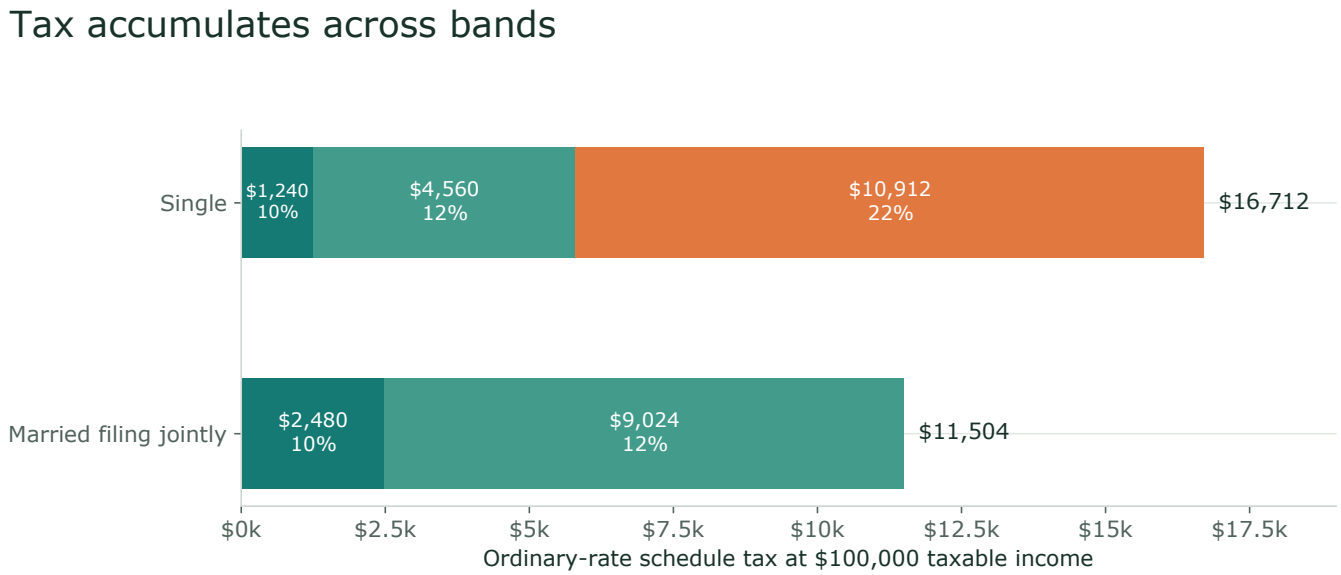
At $100,000 of Single ordinary taxable income, the first $12,400 contributes $1,240, the next $38,000 contributes $4,560, and the remaining $49,600 contributes $10,912. The result is $16,712.00, or 1,671,200 cents. This is the same amount obtained by the official row's $5,800 base plus 22% of the excess over $50,400. The agreement illustrates two decompositions of one schedule, not two different taxes. With matching reviewed facts and certificate, the existing evaluate fixture returns that cent amount. [1], [13]

At the same taxable income under the joint schedule, the first $24,800 contributes $2,480 and the remaining $75,200 contributes $9,024. The result is $11,504.00. The top occupied marginal rate is 12%, while the schedule amount divided by taxable income is 11.504%. For the Single example, those two concepts are 22% and 16.712%. The marginal percentage concerns the next increment in the underlying unrounded schedule; the average percentage concerns the entire accumulated amount. Neither is a person's total burden after credits or other taxes.

At the Single top boundary, the source base is $192,979.25 at $640,600. Increasing income to $1,000,000 adds 37% of $359,400, giving $325,957.25. This higher fixture checks both the final open band and its accumulated base. A common mistake would apply 37% to all $1,000,000 or forget the tax already accumulated in lower bands. Such a mistake could still produce a monotone function, demonstrating again why structural proofs and source-derived examples are complementary.

No deduction is applied inside any of these examples because the supplied quantity is already taxable income. If the source amount were wages or adjusted gross income instead, the calculation would be positioned incorrectly. That is why the paper explains the legal quantity chain before presenting attractive numerical outputs. A well-labeled arithmetic chart is informative only when its x-axis names the quantity that the law and the model actually consume.

Figure 5. Where the $100,000 schedule illustrations come from



Each segment is the contribution of one occupied rate band. MFJ = married filing jointly.
No deduction is taken again: the input is already taxable income. Source: IRS Rev. Proc. 2025-32.

Exact ordinary taxable-income examples at $100,000 for the two supported statuses. The contributions sum to $16,712.00 and $11,504.00 respectively. They are schedule illustrations, not complete return amounts or recommendations about filing status. [1], [13]

12. The ordinary schedule versus IRS Tax Tables and worksheets

Section 3 provides statutory authority for prescribed tax tables in lieu of the section 1 tax in the circumstances it describes, including a ceiling determined by the Secretary. The statutory text alone does not establish the familiar $100,000 filing-instruction threshold. The currently published 2025 Form 1040 line 16 instructions specify the Tax Table below $100,000 and the Tax Computation Worksheet at or above it, subject to listed exceptions. This paper uses that dated instruction to explain the method distinction, not as an unverified assertion about final 2026 return instructions. [8], [18]

A table lookup and an exact continuous schedule are mathematically different objects. If a table assigns one tax amount to an income interval, while the underlying schedule has a positive slope inside that interval, the table value cannot equal the exact schedule at every interior point. A difference between the two is therefore not automatically a bug in either implementation. A fair comparison must ask whether the same statutory and administrative method applies before comparing numeric amounts. Matching ordinary rates is not sufficient to validate Form 1040 line 16.

The 2026 Publication 505 and Form 1040-ES materials place rate schedules within a broader estimated-tax computation. They identify surrounding adjustments and situations requiring other worksheets. Estimated tax also concerns payments and timing, whereas our function returns only an ordinary schedule illustration. A user should not interpret the output as an estimated installment or balance due. [2], [7]

Preferential-rate income, alternative minimum tax, credits, and special computations create further method dependencies. The model represents several of these through exclusion flags and refuses cases marked as requiring them. It does not derive the flags or prove that false is appropriate. The child-tax-credit provisions, for example, concern eligibility and limits beyond changing an ordinary bracket coefficient; section 55 defines a separate minimum-tax computation. A larger model should represent those components as explicit calculations with reviewed inputs rather than silently fold them into a general special-case Boolean. [6], [22]

13. What the original thirty fixtures establish

Main.lean contains thirty named Boolean fixtures, evaluated both by the executable and by a compile-time example. They include accepted Single and joint records, zero income, the two $100,000 results, the $1,000,000 Single result, cumulative bracket bases, first-boundary cents, and the declared floor crossing. The final named fixture evaluates a list of all joint bases, while an additional compile-time example checks the complete Single base list. A named-fixture count is consequently not the same as a count of independently sampled monetary observations. [13]

The negative cases mutate the source, revision, statute identifier, fact year, certificate year, status agreement, supported constructor, review flag, ordinary-only flag, credit flag, AMT flag, and special-computation flag. They provide useful evidence that a small, known set of guarded assumptions is enforced. Re-executing the original release passes all thirty checks. These are authored synthetic cases, not taxpayer returns, an external filing benchmark, or a measure of how reliably a reviewer identifies the required method.

The tests and proofs detect different classes of error. The generic monotonicity theorem can reject some arithmetic regressions beyond the fixture list, but it will not reject every wrong annual threshold. A transcribed cumulative base can expose a threshold mistake, but it samples only a finite point. An exact comparator can exercise far more points, yet it can share the wrong annual source or output convention. The next section adds that comparator evidence while preserving these distinctions.

14. Reproducible differential arithmetic audit

The executed audit compares the actual compiled Lean numerator and final cent output against two exact references: an occupied-band sum using rational arithmetic and a rate-times-income-minus-subtraction expression transcribed from Publication 505. It evaluates 224,381 distinct status-and-income pairs in one batch process. There are zero exact-numerator or floored-cent mismatches, and the two references also agree. This is completed arithmetic evidence added in edition 3.0, not a proposed benchmark. All inputs are generated; no taxpayer records or complete return calculations are evaluated. [2], [23]

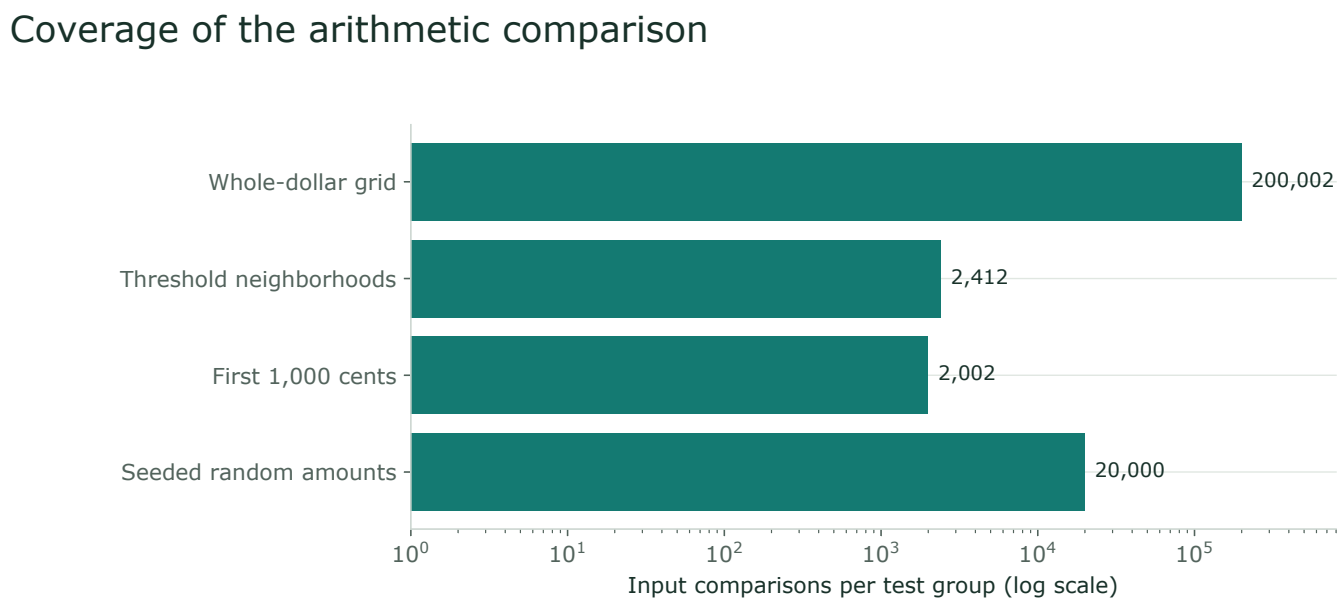
The raw design contains 200,002 whole-dollar cases from zero through $100,000 inclusive across both statuses, 2,412 cases covering every threshold plus or minus 100 cents inclusively, 2,002 cases covering every cent from zero through 1,000, and 20,000 random draws up to $10,000,000. Random generation uses Python random.Random with seed 20261116, generating 10,000 draws for Single and then 10,000 for joint status. Unioning by status and income removes thirty-five overlapping memberships from the raw total 224,416, leaving 112,189 Single and 112,192 joint cases. Category memberships overlap. [23]

The harness imports the unchanged model at commit 41bd42fe19dd3b183348504aa4f5a426d6c48502 and calls its real functions, rather than reproducing the positive-increment formula in the website. The audit source records a successful build, Lean 4.35.0, a zero exit code, and 224,381 observed rows. At execution, the batch harness was an uncommitted addition while the core model remained unchanged. Its source digests and later publication identity are separate from the original model commit; the result must not be described as having run from an already committed harness if it did not. [23]

Observed shape checks cover 224,379 consecutive tested-amount pairs and 4,406 adjacent-cent pairs. No amount monotonicity, nonnegativity, coefficient-bound, income-bound, final-floor bound, or one-cent increment failure was observed. The maximum omitted fraction was 0.99 cent; 200,537 cases had an exact whole-cent result. Those frequencies reflect the deliberately nonuniform design, especially its large whole-dollar grid. They do not estimate the frequency of rounding effects in taxpayer returns or justify a general runtime-performance claim. [23]

The public manifest links a gzip corpus containing every case, both reference values, the observed values, and group membership, with normalized row and compressed-file digests. The expected and observed canonical-output SHA-256 values are identical. The reference algorithms were authored within the same Matha Labs effort; they are independently structured, not an independent author or team. Shared constants, interpretation and final-floor convention remain common assumptions. Finite agreement is consequently valuable evidence against many implementation errors, while still leaving formal equivalence, independent legal review and full-method applicability as separate work. [23]

Figure 6. Composition of the executed arithmetic audit



Group membership can overlap. Distinct-input totals and exact mismatch counts are reported in the audit JSON.
Tests compare occupied-band arithmetic with the existing Lean executable; they do not measure filing accuracy.

Actual raw generated categories and deduplicated corpus sizes from the edition 3.0 audit. There are 224,381 distinct compiled-model evaluations and zero mismatches against either exact reference. Category memberships overlap; this is synthetic schedule-arithmetic agreement, not taxpayer accuracy or legal completeness. [23]

Table 5. Executed audit composition and exact comparisons

Generated category	Raw memberships	New unique cases in listed order	Mismatches
Whole dollars, $0-$100,000, both statuses	200,002	200,002	0
Every threshold +/-100 cents inclusive	2,412	2,403	0
Every cent, 0-1,000 inclusive, both statuses	2,002	1,980	0
Seeded random, up to $10 million	20,000	19,996	0
Total distinct cases after deduplication	224,416 raw	224,381 distinct	0 against both references

Raw memberships overlap. New unique cases depend on the displayed insertion order; only 224,381 is the executed corpus size. Numerator and floor are both compared with the occupied-band reference; the published worksheet expression also agrees. [23]

15. What established computational-law and tax-benefit systems contribute

A serious comparison begins with the problem each system is designed to solve. Catala is a language for expressing legal computations and their exceptions in close connection with source structure. Its research emphasizes the tension between legal readability and executable precision. Lean provides a general proof environment; this project uses it to state properties of a small guarded function. These are complementary choices. A domain-oriented legal representation could supply a reviewable specification, while a separately checked model addresses selected invariants or compilation obligations. That integration has not been performed here. [10], [21]

Tax-Calculator's official implementation represents a much broader federal tax microsimulation system, with policy parameters, filing-unit data and computation modules. It addresses questions involving interactions and reforms beyond the two schedule vectors. OpenFisca provides a tax-benefit modeling framework organized around variables, entities, periods and parameters. Those designs are relevant because tax computation is naturally a dependency graph over dated inputs, not a collection of unrelated bracket functions. This paper does not claim feature parity, validated interoperability, or a performance advantage over either system. [19], [20]

A useful experiment would compare a deliberately shared slice rather than feed each tool a differently interpreted scenario and call a discrepancy an error. Freeze the same year, supported status, ordinary taxable amount, absent special items, and output convention. Document every adapter transformation. Where a larger system executes an IRS table or another legally required method, separate that method difference from the raw schedule comparison. The appropriate ground truth depends on the task, not on which implementation reports

more decimal places.

The wider lesson is architectural. A legal-language model can organize exceptions, a microsimulation engine can study policy interactions, a versioned rule framework can manage periods, and a theorem prover can establish quantified properties of selected functions. Combining those strengths requires explicit interfaces and a reviewed source translation. Reimplementing a small fragment in Lean is valuable only if its proof and evaluation claims are stated at the level that fragment actually supports.

Table 6. Different research goals, not a league table

Approach	Primary contribution relevant here	What this paper does not infer
Direct piecewise or band-sum program	Transparent arithmetic reference for a frozen schedule	Not automatically a legally complete return model.
Catala	A legal-computation language emphasizing source structure and exceptions	No Catala-to-Lean compiler or equivalence result is implemented here.
Tax-Calculator	Broader federal tax microsimulation and policy-interaction implementation	No parity, correctness ranking, or end-to-end benchmark is claimed.
OpenFisca	Variables, entities, periods and parameters for tax-benefit modeling	No country model is imported or validated by this artifact.
This Lean component	A narrow guarded function, quantified invariants and exact arithmetic audit	No full-code or full-return completeness claim.

Descriptions refer to primary research or official project documentation and source, not vendor comparisons or secondary performance claims. [10], [12], [19], [20], [21]

16. A modular path from a schedule to a justified tax computation

The next formal step need not be a bigger list of statutes. It should establish a reliable boundary between two adjacent computations. One candidate is a typed taxable-income derivation for a narrowly defined population with a documented list of allowable input categories and deductions. Another is a method-selection layer that decides whether an ordinary schedule, Tax Table, or named special worksheet is applicable. Either adds a meaningful dependency that the current review flags only summarize.

For income derivation, the intermediate records should distinguish gross-income components, allowed adjustments, adjusted gross income, deduction elections, and taxable income. Each transformation needs source references, units, period and status dependencies, treatment of unavailable facts, and a theorem or check appropriate to its claim. A subtraction that produces a number is not sufficient evidence that a deduction is permitted. A future proof could establish accounting conservation or bounds while independent legal review assesses category eligibility. Sections 61, 62 and 63 supply starting points, not a complete implementation specification in a single paragraph. [4], [15], [16]

For method selection, use a sum type whose cases name the represented computation and carry the facts needed by that method. Unsupported or unresolved cases should remain distinguishable from a computed zero. The layer should explicitly consume preferential-income facts, credit applicability, AMT requirements and filing-method rules. It should not treat high confidence from an extractor as a substitute for an unresolved legal dependency. A human-readable explanation should cite the rule and facts selecting the branch, with consistency tests against the executable selector.

A particularly valuable additional theorem would connect a well-formed, ordered schedule representation to the occupied-band and positive-increment functions. Such a proof could use an inductive band structure, explicit rate changes, and a source-reviewed constant instance. It would close a mathematical equivalence gap while leaving the distinct legal translation gap visible. This is a more precise research objective than claiming that all IRS rules will become deterministic merely because one evaluator already is.

17. Versioning legal sources, model assumptions, and evidence

A reproducible result should identify at least the taxable period, selected legal and administrative sources, source bytes or digests, translation revision, implementation commit, input facts, numeric convention, and audit version. These identities answer different questions. A code commit alone cannot show which external PDF bytes were reviewed, and a source URL alone may later serve changed content. The original certificate stores fixed identifiers and a year; it is not yet an archival or legal-change monitoring service. The reviewed Revenue Procedure PDF contained 260,406 bytes and SHA-256 e9ada115fb43a4af5ea326ca91edd40ab0bf29b9be01d2c13d7b8fa96214635b; a byte digest identifies that snapshot but does not decide its legal interpretation. [1]

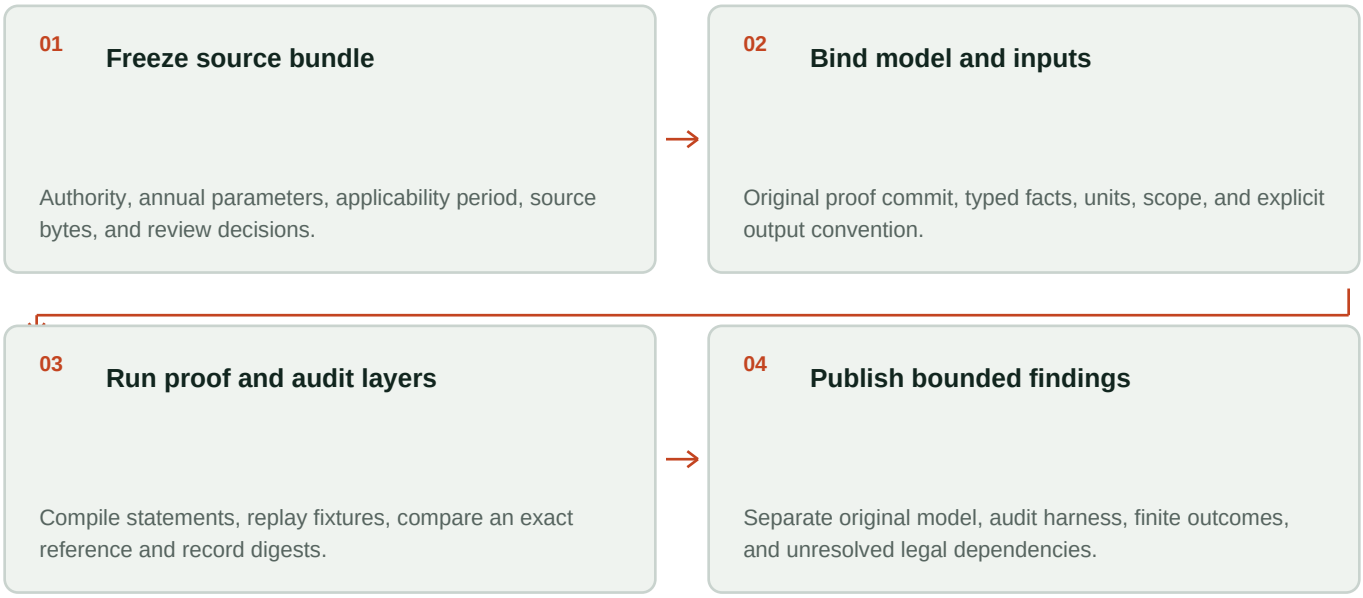
A source registry should preserve historical bundles and record why each is applicable. When a new annual procedure appears, reviewers should compare thresholds, base amounts, status categories, effective language and dependencies. A change in an adjacent deduction or special method can affect whether an input reaches the ordinary schedule even if every bracket constant stays the same. Consequently, drift review should analyze dependencies rather than only compare a two-row parameter file. The current OLRC currency notices and the publication dates of IRS materials are evidence to retain, not proof that every intervening legal issue has been resolved. [1], [3], [4]

For auditability, distinguish the original Model.lean commit from the later harness commit and the paper edition that reports the experiment. The original fourteen statements remain the statements of that original model. Adding a larger test harness must not retroactively be presented as having expanded legal coverage or changed the meaning of accepted facts. Public case artifacts should contain only generated amounts and statuses, not taxpayer records; any future record study requires its own access, privacy and adjudication arrangements.

A release process can make these distinctions practical: freeze sources, review a translation diff, run proof and fixture checks, generate the exact audit dataset, compare the recorded digests and outcomes, then publish the paper and evidence together. Failed or superseded attempts should remain identifiable. This is governance for

a research claim, not a suggestion that a release log itself establishes substantive tax correctness.

Figure 7. A proposed reproducibility record for legal arithmetic



This figure specifies the evidence organization needed for continuing research. The present release supplies a pinned model, generated arithmetic audit and source citations, not a complete legal-source registry or taxpayer-data system.

18. What should be tested next, and what would count as progress?

The completed arithmetic work should lead to more discriminating evaluations. First, formalize equivalence between the two mathematical representations under well-formed schedule assumptions. Second, commission an independent source-to-rule review that can challenge the choice of annual parameters, supported categories and method boundary. Third, test a bounded applicability module on independently adjudicated scenarios. These are different milestones; success at the first should not be reported as success at the other two.

An applicability study should define the scenario unit, supported population, tax period, source snapshot and adjudication protocol before measuring outcomes. Include ordinary cases and deliberately difficult boundaries: a changed filing status, missing income classification, conflicting documents, preferential income, deduction elections, credits, AMT flags and a method that the implementation cannot represent. Measure false numeric outputs for unsupported cases, erroneous refusals, and the proportion remaining in scope. Keep scenario families together when separating development and evaluation sets, and report disagreements rather than silently resolving every ambiguous case in favor of the program.

Comparisons with established systems should preserve common inputs and disclose adapter assumptions. A disagreement can originate in source year, income derivation, filing category, table versus schedule method,

rounding, or arithmetic. An aggregate exact-dollar score would obscure those causes. The evaluator should retain the numerator and branch decision where possible and explain each discrepancy before treating it as a defect. No real taxpayer accuracy, filing benefit or reduction in compliance cost has been measured by the current artifact.

The substantive result is a small but inspectable component: its annual constants are visible, its formula can be derived, its guard can be read in full, fourteen statements are machine checked, and an exact differential audit can be replayed. That is a stronger foundation than an opaque calculator and a much smaller claim than formalizing the Internal Revenue Code. The next useful publication should demonstrate a new justified interface or proof obligation, with evidence appropriate to that addition, rather than repeat a larger theorem or test count as if it were legal completeness.

References

[1] IRS, Revenue Procedure 2025-32. Section 4.01 Table 1, printed page 10, and Table 3, printed page 11; section 1 as-of October 9, 2025; section 5.02 applicability. Bulletin publication November 3, 2025. PDF snapshot 260,406 bytes, SHA-256 e9ada115fb43a4af5ea326ca91edd40ab0bf29b9be01d2c13d7b8fa96214635b. Accessed 2026-11-16.

[2] IRS, Publication 505 (2026): Tax Withholding and Estimated Tax. 2026 Worksheet 1-4(a) Single and (c) Married Filing Jointly, multiplication and subtraction columns; Chapter 2 and worksheets show surrounding special-method, AMT and credit dependencies. This estimated-tax publication is not a final Form 1040 instruction package. Accessed 2026-11-16.

[3] Office of the Law Revision Counsel, 26 U.S.C. section 1: Tax imposed. Subsections (j), (f) and (h) provide annual-rate and preferential-rate context. Current online text observed through Public Law 119-142; historical opening tables are not the selected 2026 constants. Accessed 2026-11-16.

[4] Office of the Law Revision Counsel, 26 U.S.C. section 63: Taxable income defined. Subsections (a)-(c): taxable-income and deduction definitions; identifies upstream computations the model does not perform. Accessed 2026-11-16.

[5] Office of the Law Revision Counsel, 26 U.S.C. section 7703: Determination of marital status. Subsections (a)-(b): marital-status determination and special conditions; demonstrates why a status label requires separate reviewed facts. Accessed 2026-11-16.

[6] Office of the Law Revision Counsel, 26 U.S.C. section 55: Alternative minimum tax imposed. Subsections (a)-(b): a separate minimum-tax computation; the release flags AMT cases for refusal rather than implementing these provisions. Accessed 2026-11-16.

[7] IRS, 2026 Form 1040-ES: Estimated Tax for Individuals. How To Figure Your Estimated Tax; 2026 Estimated Tax Worksheet, line 4 and surrounding lines; 2026 Tax Rate Schedules. Estimated-tax context is distinct from this illustration. Accessed 2026-11-16.

[8] IRS, 2025 Instructions for Form 1040. Line 16, Tax: less-than-$100,000 Tax Table versus Tax Computation Worksheet and listed exceptions. Cited only to explain method distinctions, not as 2026 rate authority. Accessed 2026-11-16.

[9] IRS, Working Families Tax Cuts: Individuals and workers. Tax inflation adjustments, Marginal rates for tax year 2026: official current summary cross-check of both supported threshold vectors. Accessed 2026-11-16.

[10] Merigoux, Chataing and Protzenko, Catala: A Programming Language for the Law. 2021 paper, revised July 2, 2021; abstract and computational-law language motivation, including tax examples. No benchmark or implementation result is reused. Accessed 2026-11-16.

[11] Lean project, Theorem Proving in Lean 4: Axioms and Computation. Chapter on proof checking, foundational assumptions and computation; bounds the interpretation of machine-checked theorem claims. Accessed 2026-11-16.

[12] Matha Labs, lean-irs-tax: model and proofs. Pinned threshold vectors, Conditions, numerator, illustrationCents, evaluate and fourteen named theorem statements. Accessed 2026-11-16.

[13] Matha Labs, lean-irs-tax: executable fixtures. singleFacts, singleBases, jointBases and thirty fixtures, with compile-time examples; source of the reported regression inventory. Accessed 2026-11-16.

[14] Lexifina, Formalizing all US case law in Lean. November 13, 2026; Section 2, Nothing is taken on trust, motivates evidence-bearing interpretation with a separate formal checker; no legal corpus or vendor measurement reused. Accessed 2026-11-16.

[15] Office of the Law Revision Counsel, 26 U.S.C. section 61: Gross income defined. Subsection (a) broad definition and categories, with exceptions and cross-references. Upstream income classification is outside the model. Accessed 2026-11-16.

[16] Office of the Law Revision Counsel, 26 U.S.C. section 62: Adjusted gross income defined. Subsection (a) deductions used in defining AGI. The implementation receives taxable income rather than deriving these items. Accessed 2026-11-16.

[17] Office of the Law Revision Counsel, 26 U.S.C. section 6013: Joint returns of income tax by husband and wife. Joint-return conditions; read with section 7703 rather than treating a joint status constructor as evidence of eligibility. Accessed 2026-11-16.

[18] Office of the Law Revision Counsel, 26 U.S.C. section 3: Tax tables for individuals. Subsection (a) prescribed annual tax tables in lieu of section 1 and Secretary-determined ceiling; statute alone is not the current $100,000 instruction. Accessed 2026-11-16.

[19] Tax-Calculator: official documentation and implementation. Official source repository for the federal tax microsimulation project; comparison concerns system goals and interfaces, not a validated cross-system result. Accessed 2026-11-16.

[20] OpenFisca: official core framework. Official source for tax-benefit calculation framework with variables, entities, periods and parameters; no country model is imported here. Accessed 2026-11-16.

[21] Catala: official programming-language implementation. Official source and linked language documentation; complementary to the primary research paper in reference 10. No compiler integration is implemented. Accessed 2026-11-16.

[22] Office of the Law Revision Counsel, 26 U.S.C. section 24: Child tax credit. Example of credit eligibility and limitation rules outside an ordinary-income rate schedule; no credit amount or complete 2026 credit algorithm is asserted. Accessed 2026-11-16.

[23] Matha Labs, edition 3.0 exact rate-schedule audit manifest and case corpus. Audit version 1.0: unchanged original Model.lean; compiled batch outputs; 224,381 distinct generated cases; occupied-band and published worksheet references; zero mismatches; source, dataset and output digests. Harness was uncommitted at execution and is identified separately from original model ref. Accessed 2026-11-16.

Reproducibility record

Source: https://github.com/RevanthM/lean-irs-tax | Evaluated implementation: 41bd42fe19dd3b183348504aa4f5a426d6c48502