Checking clinical coding evidence in Lean

Matha Labs | Research prototype v0.1.0 | November 15, 2026

Abstract

This paper presents an executable certificate checker for a deliberately small clinical coding problem. A proposal names a reviewed note, a concept, an exact quotation, and a policy revision. A pure Lean function accepts the proposal only when its explicit conditions hold. Twelve named theorems establish structural properties of this function, and twenty-three synthetic regression fixtures exercise accepted examples and rejected mutations. The supported identifiers are R50.9 and I10, represented by two restricted synthetic phrases. The result demonstrates how evidence checking can constrain a future generative system. It does not demonstrate clinical extraction accuracy, complete ICD-10-CM compliance, or patient-level coding performance.

1. Research question

A language model can propose a code with fluent reasoning while omitting the source sentence, confusing a suspected condition with an established one, or applying a rule from the wrong period. A useful research question is whether a small, independently executable checker can make some of those failures impossible at the acceptance boundary. The answer depends on precisely naming what the checker receives and what it proves.

Our unit of work is a coding certificate, not a free-form explanation. The proposal must identify the source note and policy revision and provide a quotation at an explicit offset. The final public function produces either an illustrative code or no code. Human review remains necessary wherever interpretation or policy coverage is incomplete. A deterministic rejection is itself a reproducible result; a system need not manufacture a code to appear successful.

2. Relationship to evidence-based formalization

Lexifina describes a larger system in which extracted legal claims are accompanied by textual evidence and checked against formal authority rules [4]. We adopt the general separation between a fallible proposal mechanism and a deterministic checker. We have not reproduced that system, imported its legal corpus, independently verified its reported evaluation, or transferred its benchmark numbers to healthcare.

The clinical setting introduces a distinct semantic boundary. A quotation being present does not establish that the condition applies to the patient, belongs to the current encounter, or has the intended certainty. For example, the word "Fever" can occur inside a denial, a family history, or a hypothetical instruction. Therefore the checker consumes a `ReviewedNote` whose semantic annotations have already been reviewed. The proposal cannot override these annotations. This restriction makes the initial experiment smaller and its claims clearer.

3. Policy context and scope

The CMS and NCHS FY 2027 guidance applies from October 1, 2026 through September 30, 2027. Its outpatient section distinguishes uncertain diagnoses from documented signs, symptoms, and other established reasons for the encounter [1]. We use that distinction to motivate an affirmed-only outpatient fixture. The official classification includes many additional conventions and instructions; this project implements only the listed fixture conditions.

The current model has two supported concepts: fever and essential hypertension. Their illustrative identifiers are R50.9 and I10. The strings `Fever` and `Essential hypertension` are intentionally restrictive fixture phrases. A real note saying the same thing differently will generally be rejected. The model does not implement the Alphabetic Index, full Tabular List, exclusions, combination codes, sequencing, laterality, encounter characters, or the highest-specificity decision across the complete record. The official publication and lookup services remain the references for real classification work [2, 3].

4. Input and certificate model

`ReviewedNote` contains an identifier, text, encounter date, setting, clinician-review flag, reviewed concept, reviewed certainty, and current-encounter flag. Certainty is an explicit enumeration: affirmed, uncertain, or negated. Setting is outpatient or inpatient. Concept is fever, hypertension, or unsupported. These finite enumerations prevent arbitrary text from being silently treated as a supported diagnosis class.

`Certificate` contains a note identifier, policy revision, proposed concept, character offset, and quotation. Offsets count Unicode scalar values, as represented by Lean's string-to-list operation. They do not count bytes or UTF-16 code units. An adapter that uses a different indexing convention must convert it explicitly. The checker requires a nonempty quotation and validates both the span boundary and the extracted text.

Dates are represented as natural-number YYYYMMDD values. `validCalendarDate` validates month lengths, including Gregorian leap-year rules, before the effective-window comparison can succeed. The period is half-open: the first day is accepted, and the next period's first day is rejected. This treatment avoids an ambiguous inclusive upper bound.

5. Executable acceptance rule

`Conditions` is the conjunction of fifteen explicit checks. Identity must match; the revision must equal `FY2027-fixture-v1`; the input must be clinician reviewed; the setting must be outpatient; the date must lie inside the fixture interval; the condition must affect the current encounter; certainty must be affirmed; the proposed and reviewed concepts must agree; the concept must be supported; the quotation must be nonempty, in bounds, present at the stated offset, and equal to the restricted phrase; and the date must be a valid calendar date.

`verify` decides this proposition and returns a Boolean. `result` returns a code only after `verify` succeeds. All inputs and policy constants are ordinary data. There is no network call, hidden prompt, mutable database, temperature setting, or language-model invocation in this computation. Holding the reviewed input and certificate fixed therefore holds the result fixed. This property does not imply that different extractors will produce the same reviewed input.

6. Proof inventory

The twelve named theorems are declared in `ClinicalCoding/Model.lean` and checked during `lake build`:

- `verification_iff` establishes equivalence between successful Boolean verification and the stated acceptance proposition.
- `accepted_requires_review` and `accepted_is_outpatient` establish the review and setting requirements for every accepted certificate.
- `accepted_is_affirmed` and `accepted_concept_matches` bind the accepted proposal to the reviewed semantic annotations.
- `accepted_quote_exists` and `accepted_quote_in_bounds` establish the exact span and boundary conditions.
- `accepted_date_in_window` establishes both effective-date inequalities.
- `uncertain_refused`, `negated_refused`, and `unsupported_refused` establish rejection for those input classes.
- `rejection_has_no_code` establishes that failed verification cannot emit an identifier through `result`.

These are properties of the implemented model. None proves that the reviewed annotations are medically correct or that the policy is complete. The source contains no proof placeholders, custom axioms, admitted results, or unsafe declarations. The project uses Lean's bundled standard library and pins Lean to version 4.35.0.

7. Synthetic evaluation

The executable contains twenty-three named fixtures. Two affirmative cases cover the supported concepts, and expected-output checks confirm their identifiers. Rejected mutations cover uncertainty, negation, unsupported concept substitution, wrong source identity, modified quotation, an out-of-bounds offset, an empty quotation, missing review, inpatient setting, a historical condition, the wrong revision, and dates outside the fixture period. Boundary fixtures test the first and last effective dates, invalid February dates, and leap-day validation.

The complete list is also evaluated in a compile-time `example` using `by decide`. The executable prints one result per fixture and exits with an error if any fixture fails. At publication preparation, the pinned toolchain built successfully and the executable reported 23/23 passing synthetic fixtures. These examples are authored regression cases, not a random sample, external benchmark, sensitivity estimate, or clinical validation cohort. Their purpose is to document specific acceptance and rejection behavior.

8. Trust boundary and unresolved work

The trusted inputs are the reviewed note text and its semantic annotations, the curated phrase mapping, and the selected policy revision. The Boolean review flag records an upstream assertion; it does not authenticate a clinician or verify a signature. Source identity is a data equality check, not a cryptographic provenance guarantee. The code makes these boundaries visible rather than solving them implicitly.

An end-to-end system would need authenticated record ingestion, independently reviewed terminology and policy updates, appropriately governed patient-data handling, and a measured extraction stage. It would also need a richer abstention result that explains which checks failed. This release returns `none` on every rejection,

so callers cannot yet distinguish all failure reasons from the return value alone. They can inspect each explicit condition during development.

9. Next research steps

The next useful experiment is a reviewed annotation set with adjudicated labels for certainty, temporality, patient attribution, and exact spans. Evaluation should measure extraction errors separately from checker rejection, report abstention coverage, and preserve a held-out set. Additional concepts should be introduced only with their classification dependencies, rather than by enlarging a name-to-code dictionary. A versioned policy registry and independently implemented reference checker would support disagreement analysis. Those are proposed extensions; they have not been implemented or evaluated here.

References

1. CMS and NCHS. FY 2027 ICD-10-CM Official Guidelines for Coding and Reporting. https://www.cms.gov/files/document/fy-2027-icd-10-cm-coding-guidelines.pdf

2. CMS. ICD-10 code publications. https://www.cms.gov/medicare/coding-billing/icd-10-codes

3. CDC. ICD-10-CM lookup tool. https://icd10cmtool.cdc.gov/

4. Lexifina. Formalizing all US case law in Lean. November 13, 2026. https://lexifina.com/blog/formalizing-all-us-case-law-in-lean

5. Lean. Installation and language resources. https://lean-lang.org/install/