FORMAL METHODS / DECISION SYSTEMS

Evidence-bound clinical coding: a small executable Lean model

Twelve proved properties of a two-concept, reviewed-input checker, with a protocol for evaluating the semantic boundary

Revanth Matha · Matha Labs Research whitepaper 2.0 | Revised 16 November 2026

CENTRAL CONTRIBUTION

An executable certificate gate binding a proposal to a reviewed note, exact text span, restricted concept, setting, date, and fixture revision.

The pinned code checks two illustrative concepts against reviewed structured notes using the exact phrases Fever and Essential hypertension. It neither imports the complete FY 2027 classification nor derives diagnoses from symptoms, and it does not determine reimbursement.

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 would a useful checked coding decision establish? 4

2. Current classification context and the meaning of FY2027-fixture-v1 5

3. Evidence representation and its trust boundary 5

4. Formal acceptance predicate and executable behavior 6

5. A matching quotation can still be semantically wrong 7

6. Exact theorem inventory and logical interpretation 7

7. Worked accepted and refused cases 9

8. What the present evaluation establishes 10

9. Comparison with other coding architectures 10

10. A concrete next evaluation protocol 11

11. Source drift, review governance, and patient-data boundaries 12

12. Research boundary and credible expansion milestones 13

References 13

Abstract

Clinical coding combines language interpretation, clinical documentation, classification conventions, and encounter-specific rules. Formal verification can constrain a decision procedure, but it cannot establish the accuracy of an unexamined extraction. This paper studies that boundary through a small Lean 4 implementation. The input is a clinician-reviewed note with explicit concept, certainty, setting, and temporality annotations. A separate certificate proposes a concept and an exact quotation at a Unicode-scalar offset. The checker validates source identity, a fixed policy revision, the reviewed annotations, calendar and applicability dates, span bounds, and a restricted phrase mapping before returning an illustrative identifier. The implementation contains two supported fixture concepts, twelve named theorems, and twenty-three synthetic regression cases. Re-execution of the pinned source passes all cases. The proofs establish structural requirements for accepted certificates and refusal behavior for several unsupported states; they do not establish clinical entailment, complete ICD-10-CM compliance, or extraction accuracy. Current CDC and CMS publications provide the policy context, while the implementation remains the authored FY2027-fixture-v1 bundle rather than a complete official release. We analyze an adversarial counterexample in which a matching word can coexist with an incorrect semantic annotation, compare the checker with contextual NLP and conventional rule engines, and specify a prospective evaluation separating extraction, abstention, policy translation, and coding outcomes. The proposed next milestone is an independently adjudicated annotation and policy test set, with immutable source versions and auditable review decisions.

Keywords: Lean 4, ICD-10-CM, clinical coding, evidence certificates, assertion classification, selective prediction, formal verification

Contributions

1. An executable certificate gate binding a proposal to a reviewed note, exact text span, restricted concept, setting, date, and fixture revision.

2. Twelve named Lean theorems establishing acceptance preconditions and refusal properties, without project-specific axioms or admitted proofs.

3. Twenty-three reproducible synthetic cases covering the two supported concepts and important mutations; an explicit account of what these cases cannot measure.

4. A source-grounded analysis of the remaining semantic boundary and a specified future evaluation protocol; this protocol is proposed, not an executed benchmark.

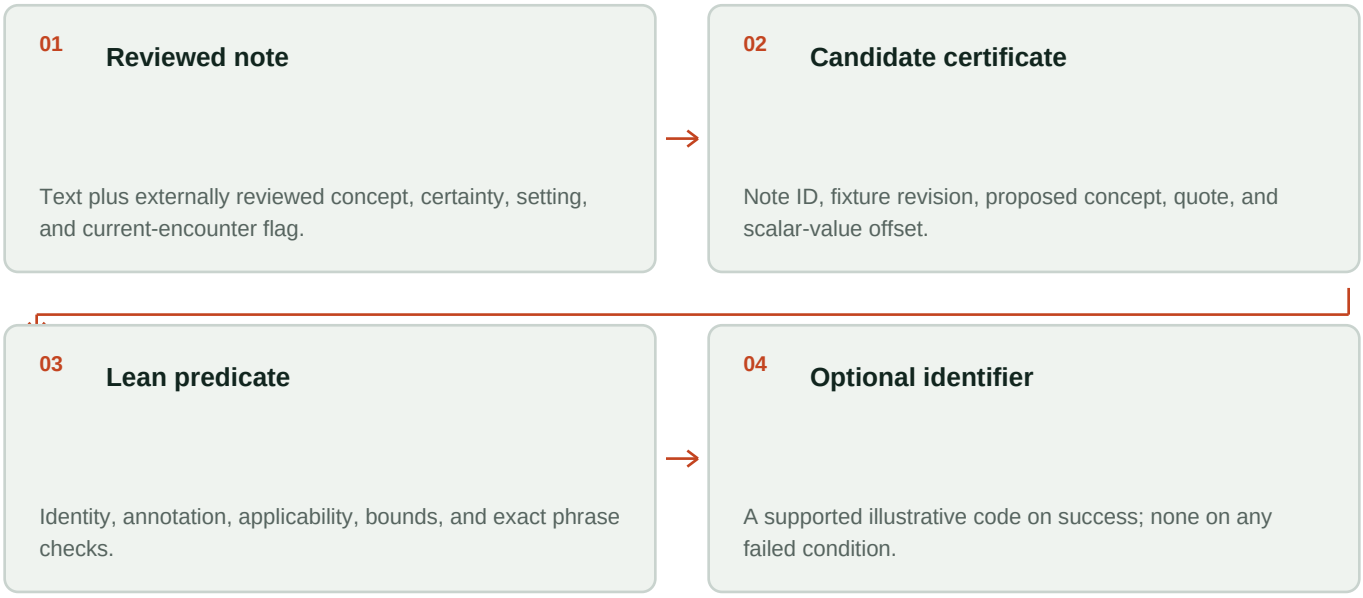
1. What would a useful checked coding decision establish?

A useful coding system must answer several different questions. What does the record say? Does a finding apply to this patient and encounter? Which classification instruction governs it? Does the proposed identifier satisfy that instruction at the relevant date? Conflating these questions makes it difficult to tell whether a wrong output originated in extraction, documentation, terminology, or policy. CDC describes ICD-10-CM as a diagnosis and morbidity classification, while its publication process supplies versioned files that a coding implementation must select explicitly. Neither fact implies that a language model's plausible diagnosis is documented in the record. [2], [3]

The research question here is deliberately narrower: given reviewed semantic facts, can an independent executable checker prevent a proposed certificate from violating selected structural conditions? The artifact answers that question for two restricted examples. It provides a readable acceptance predicate, proofs about that predicate, and replayable mutations. It is useful as a boundary experiment because every accepted output has identifiable assumptions. It is not evidence that the surrounding clinical interpretation has been solved. The code and fixture list are the primary evidence for implementation claims throughout this paper. [9], [10]

The architecture follows the general evidence-certificate idea illustrated by Lexifina: an extractor proposes information and a separate formal model constrains what may be accepted. The domains and evaluations are different. No legal corpus, benchmark result, or proof count transfers from that work to this clinical artifact. The substantive question is which facts should be trusted upstream, and which should become independently checkable obligations downstream. [11]

Figure 1. The executed clinical certificate path



This is the implemented dataflow in the pinned source, not a deployed clinical workflow. The review step is an upstream assumption; no neural extractor or clinical adjudication service runs in the artifact. [9], [10]

2. Current classification context and the meaning of FY2027-fixture-v1

As checked on November 16, 2026, CDC's publication page identifies the October 1, 2026 FY 2027 ICD-10-CM release for services through September 30, 2027. CMS publishes the corresponding guideline and code-file entry points. This dates the authoritative context; it does not certify a third-party implementation. The local release string FY2027-fixture-v1 is authored by Matha Labs and denotes a small experiment. No official file corpus was imported into its executable mapping. The paper therefore does not claim complete current-code coverage merely because the dates align. [1], [2], [4]

The relevant outpatient guidance distinguishes uncertain diagnoses from the documented level of certainty available for an encounter. The fixture conservatively requires an affirmed reviewed concept. That restriction also applies to its symptom example; it is a modeling choice, not a complete restatement of outpatient rules. The official classification's conventions, specificity, exclusions, sequencing, and chapter-specific instructions remain outside the artifact. Its two identifiers must consequently be read as demonstration outputs, not a general recommendation to assign those codes whenever a phrase appears. [1]

This distinction also prevents a subtle version error: a future policy update cannot be represented adequately by changing a date constant alone. A meaningful update would identify changed source instructions, affected concepts, modified proof obligations, and new regression cases. The current half-open date interval is one applicability guard for one fixed bundle. It is not a legal effective-date engine and does not resolve midyear releases, payer adoption differences, or documentation corrections. [2], [4]

3. Evidence representation and its trust boundary

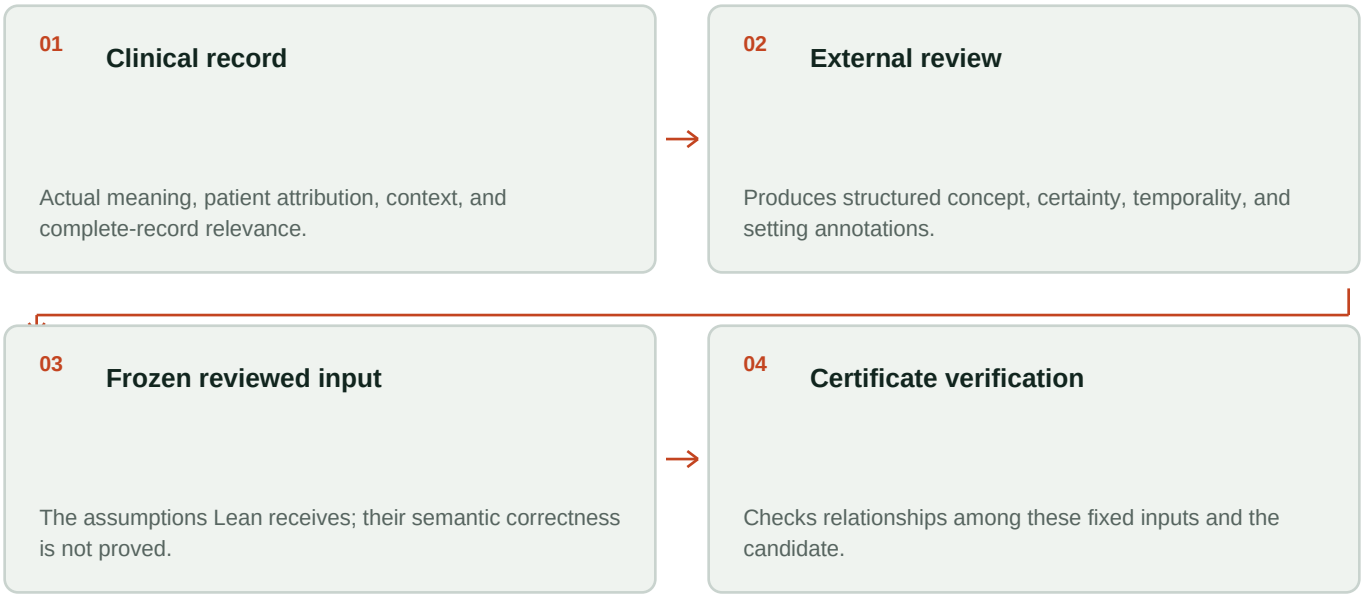
ReviewedNote contains an ID, complete text string, encounter date, setting, clinicianReviewed Boolean, reviewed concept, reviewed certainty, and currentEncounter Boolean. Concept has fever, hypertension, and unsupported constructors. Certainty has affirmed, uncertain, and negated constructors. A certificate cannot replace certainty with its own preferred label: its proposed concept must agree with the reviewed note's concept. This constrains the proposal while leaving the correctness of the reviewed record itself outside the theorem. [9]

Certificate contains a note ID, release string, concept, offset, and quotation. Offsets count Unicode scalar values through String.toList, not UTF-8 bytes or UTF-16 code units. That detail matters when integrating an extractor or browser editor. A visually identical quotation can have different indexing or normalization behavior; a future adapter should declare its encoding and retain the original source bytes. The current code neither normalizes equivalent Unicode sequences nor tests a broad multilingual corpus. Its exact equality semantics are intentional and restrictive. [9], [10]

The clinicianReviewed field is an attestation supplied as data. It is not a digital signature, role check, credential validation, or evidence that an authorized person performed a documented review. Similarly, note-ID agreement is not proof of provenance authenticity. A future operational design should bind review decisions to authenticated source versions and reviewer identities, while keeping those identifiers outside public artifacts

where required. That extension would strengthen provenance but still would not make an incorrect interpretation true.

Figure 2. Interpretation remains outside the checked kernel



The transition from a record to reviewed annotations is the principal unresolved semantic boundary. The figure distinguishes an assumed annotation process from the implemented certificate checks. [5], [6], [9]

4. Formal acceptance predicate and executable behavior

Conditions is a conjunction of fifteen obligations. The certificate ID must match the note, its release must equal the fixture revision, and review and outpatient-setting requirements must hold. The date must satisfy both interval inequalities and a calendar-validity check. The condition must belong to the current encounter and be affirmed. The proposed concept must match the reviewed concept and be supported. Finally the quote must be nonempty, its end position must remain inside the text, extraction at the supplied offset must equal the quote, and the quote must equal the reviewed concept's canonical phrase. [9]

verify decides this proposition as a Boolean. result returns the mapping only when verification succeeds; otherwise it returns none. This makes the acceptance gate inspectable and replayable for fixed inputs. none is deliberately coarse. It does not distinguish a missing review from an unsupported setting or bad span, and callers must not reinterpret it as an empty diagnosis or clinical absence. A typed rejection reason would be a useful extension, but it is not implemented by the existing return type.

The calendar validator checks Gregorian month lengths and leap years for the natural-number YYYYMMDD representation. The interval begins at 20261001 and excludes 20271001. Validity of this arithmetic prevents impossible dates from slipping through an integer comparison. It does not establish that the date belongs to the right clinical event, resolve service-date versus discharge-date selection generally, or authenticate a corrected

record. Those are distinct factual and policy obligations. [1], [9]

```
verify(n,c) = decide(Conditions(n,c)); result(n,c) = if verify(n,c) then code(c.concept) else none
```

n is the reviewed note and c the proposed certificate. Conditions is the exact fifteen-part conjunction in the pinned source; code is the restricted two-concept mapping. This formula states executable behavior, not a theorem of medical entailment.

5. A matching quotation can still be semantically wrong

Consider the original synthetic note Fever. Its affirmed reviewed concept and quote at offset zero satisfy the fixture. Now consider a hypothetical note No Fever with an incorrectly affirmed fever annotation and a quote Fever at scalar offset three. The quote is present, in bounds, and equal to the canonical phrase. The structural gate can accept those mutually consistent but semantically incorrect inputs if the remaining fields are valid. This counterexample is an analysis of the predicate, not an added regression case or evidence about a model's observed error rate. [9]

The lesson is precise: accepted_is_affirmed proves the value of n.certainty, not the truth of an affirmative statement in arbitrary clinical prose. accepted_quote_exists proves string equality at a location, not that the surrounding sentence supports the intended finding. Negated and uncertain input labels are refused, but that guarantee depends on upstream annotation being correct. A paper that calls these proofs verified diagnosis inference would overstate the implementation.

Clinical contextualization is a separate research problem. ConText explicitly studies negation, experiencer, and temporal properties; the i2b2/VA challenge separates concept extraction, assertion classification, and relation classification. These primary studies motivate decomposed evaluation rather than a single undifferentiated coding score. They are methodological references, not claims that their historical performance predicts this prototype's behavior or that their data can be reused without access review. [5], [6]

6. Exact theorem inventory and logical interpretation

The pinned module contains twelve named theorems. The table gives their actual statements in readable notation, retaining the distinction between assumptions about reviewed input and conclusions about program behavior. Lean checks the proofs against their statements. Such checking does not validate a source translation or establish that the statements cover every relevant failure. [8], [9]

The project contains no authored axiom, unsafe declaration, admitted proof, or proof placeholder. This is not a claim of independence from Lean's ordinary logical foundations, compiler, runtime, or hardware. The build pins Lean 4.35.0 and uses its bundled standard library. Reproducibility should preserve the exact source commit and toolchain rather than treating a theorem count as a substitute for a model review.

Table 1. Twelve named statements in ClinicalCoding/Model.lean

Named theorem	Statement under the displayed assumptions	Interpretation
verification_iff	verify(n,c)=true iff Conditions(n,c)	Boolean acceptance matches the stated predicate.
accepted_requires_review	verify(n,c)=true implies n.clinicianReviewed=true	Requires an input attestation; does not authenticate review.
accepted_is_outpatient	verify(n,c)=true implies n.setting=outpatient	The accepted input uses the represented setting.
accepted_is_affirmed	verify(n,c)=true implies n.certainty=affirmed	Constrains the annotation, not prose semantics.
accepted_concept_matches	verify(n,c)=true implies c.concept=n.concept	Candidate agrees with reviewed concept.
accepted_quote_exists	verify(n,c)=true implies span(n.text,c.offset, c.quotation.length)=c.quotation	Exact source-text equality.
accepted_date_in_window	verify(n,c)=true implies 20261001<=n.encounterDate<20271001	Both fixture-date bounds hold.
uncertain_refused	n.certainty=uncertain implies verify(n,c)=false	Refuses the represented uncertain state.
negated_refused	n.certainty=negated implies verify(n,c)=false	Refuses the represented negated state.
rejection_has_no_code	verify(n,c)=false implies result(n,c)=none	The guarded wrapper emits no code on rejection.
accepted_quote_in_bounds	verify(n,c)=true implies c.offset+c.quotation.length<=n.text.length	The accepted span ends inside the text.
unsupported_refused	c.concept=unsupported implies verify(n,c)=false	Refuses the explicit unsupported constructor.

Exact names and implications are transcribed from the pinned model. Abbreviations n and c denote ReviewedNote and Certificate. No theorem in this table establishes clinical correctness. [9]

7. Worked accepted and refused cases

The baseline case is fully synthetic: ID synthetic-1, text Fever, date 20261115, outpatient setting, affirmed fever, current encounter, and review true. A certificate with matching ID and revision quotes Fever at offset zero and proposes fever. verify returns true and result returns some R50.9. The second example uses the complete text Essential hypertension and the matching reviewed concept; it returns some I10. These are the only supported concepts and exact phrases. [10]

Mutating one part clarifies the boundary. Reusing the fever record while proposing hypertension fails concept agreement. Changing reviewed certainty to negated or uncertain fails the affirmation requirement. Changing only the certificate quote's capitalization fails exact text equality and the canonical mapping. Inpatient input is routed to none because no inpatient policy is represented, not because the condition is clinically absent. A historical reviewed condition similarly fails this fixture's current-encounter requirement.

Boundary tests exercise the first and last valid dates and reject the exclusive end date. February 30 is invalid even when numerically inside the interval. A separate leap-day utility test checks calendar behavior outside the policy period; it is not an accepted encounter outside the applicability window. Keeping utility tests distinct from clinical cases avoids counting every passing Boolean as a successful coding decision.

Table 2. Selected executed synthetic cases

Case or mutation	Observed behavior	Reason and limit
Affirmed Fever at offset 0	some R50.9	Supported reviewed phrase; no diagnostic inference.
Reviewed Essential hypertension	some I10	Second restricted demonstration mapping.
Fever proposed as hypertension	Verification false	Reviewed concept and proposal disagree.
Negated reviewed certainty	Verification false	Refusal depends on correctness of supplied annotation.
Quote fever instead of Fever	Verification false	Equality is case-sensitive; no semantic normalization.
Encounter date 20271001	Verification false	Exclusive end of fixed fixture window.
Leap date 20280229	Calendar utility true	Does not imply policy acceptance in 2028.

These rows describe members of the existing twenty-three-case fixture list. They are not additional measurements or a sample of real records. [10]

8. What the present evaluation establishes

Re-executing the pinned executable yields twenty-three passing named checks. The list covers affirmative examples, proposed-concept substitution, wrong IDs, quote changes, empty and out-of-range spans, missing review, unsupported settings and concepts, historical flags, policy dates and versions, expected identifiers, and calendar edge cases. The same list is evaluated by a compile-time example. A source scan checks for selected proof holes and declarations. These layers provide regression evidence against specific mistakes in this small artifact. [9], [10]

The denominator is twenty-three authored assertions, not twenty-three patients or randomly selected encounters. Several cases test a utility or an expected rejection rather than successful coding. There is no independent clinical label set, held-out institution, inter-rater agreement statistic, model comparison, calibration curve, throughput result, or measured reduction in coding errors. A passing fixture suite therefore supports implementation consistency, not an estimate of clinical effectiveness.

The suite also does not lock theorem statements against weakening, perform mutation testing of every acceptance conjunct, or compare an independently implemented policy. The number of proved statements alone says little about those omissions. The next experimental release should test whether meaningful changes to each gate are detected and whether another author can reconstruct the intended predicate from the source specification without reading the Lean implementation.

9. Comparison with other coding architectures

A direct generative approach can map note text to codes and explanatory prose, but evaluating it requires independent labels for both extraction and coding decisions. A constrained vocabulary limits output form without guaranteeing that the chosen label is documented. Retrieval can improve source visibility while still selecting the wrong encounter, rule, or policy version. The certificate gate addresses a different problem: it specifies what must hold after reviewed facts and a candidate have already been supplied.

Contextual rule systems such as ConText expose linguistic triggers and scope decisions. They can be useful extraction baselines because their intermediate annotations are inspectable. A conventional program or database rule engine can implement the same conjunction as this artifact and should be included in a comparison. Formalization offers quantified statements about the encoded function; it does not automatically improve the quality of the underlying linguistic or clinical model. [5], [8]

A hybrid experiment should measure the incremental value of each boundary. Compare identical proposals under an unchecked output path, an ordinary independently coded validator, and the Lean checker. If the two validators disagree, investigate implementation or specification drift before treating either as ground truth. Human adjudication remains necessary for the semantic labels and complete classification decisions. The i2b2 task decomposition offers a useful precedent for measuring those stages separately, without implying that its label scheme is sufficient for full coding. [6]

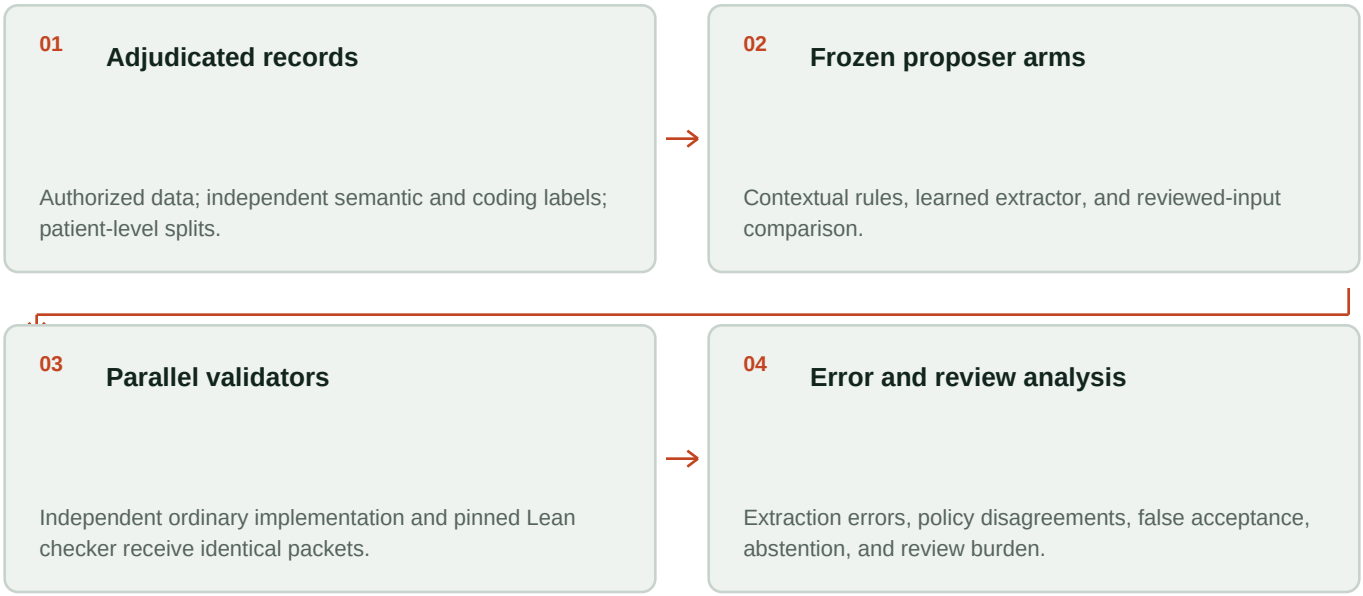
10. A concrete next evaluation protocol

The proposed study begins with authorized, independently adjudicated records and a preregistered scope. Select a supported outpatient task, define the unit of annotation, and create labels for source spans, patient attribution, certainty, current relevance, proposed concept, and applicable classification dependencies. Use at least two qualified annotators with a documented disagreement process. Partition by patient and institution before tuning, retain a temporal holdout, and keep duplicate or templated text families from leaking across splits. These are proposed design requirements, not completed data collection. [6]

Freeze three comparison arms: a contextual-rule baseline, a trained or prompted extractor, and a reviewed-input upper-bound experiment. Feed their proposals to both an ordinary reference validator and Lean. Report span precision and recall, assertion confusion matrices, semantic false acceptance among emitted candidates, coding agreement against adjudicated full-record decisions, and abstention coverage. Provide confidence intervals clustered at the patient level where repeated encounters occur. Never count refusal alone as correct clinical coding; report the review burden it creates.

A separate adversarial suite should include negation outside the quoted span, family-history statements, hypothetical instructions, copied-forward diagnoses, contradictory sections, alternate Unicode forms, invalid offsets, and source/version substitutions. Require all supported-gate mutation tests to be detected and publish disagreement examples in privacy-safe form. Establish stopping criteria before evaluation: unsupported classifications return review, unexplained validator disagreements block a broader scope claim, and material source changes trigger revalidation. The current paper supplies the protocol and executable foundation, not its future results.

Figure 3. Proposed evaluation with separate error accounting



This is a prospective study plan. No record cohort, learned extractor comparison, confidence interval, or benchmark score has been produced by the pinned artifact. [5], [6], [9], [10]

11. Source drift, review governance, and patient-data boundaries

A maintainable coding model needs immutable source snapshots, effective intervals, transformation records, and reviewer decisions. CDC's release page and CMS's publication entry point should be treated as discovery sources rather than as stable content hashes. Record downloaded artifact digests, retrieval times, applicable sections, and which concepts depend on each instruction. A change in a source file should create a reviewable policy revision; a passing historical replay must remain reproducible under its original bundle. The current implementation has an equality-checked release string and fixed dates, but not this registry or impact analysis. [2], [4]

Review governance should distinguish authoring a rule, approving its clinical interpretation, and approving its operational use. Proposed audit records would identify the source version, reviewer role, uncertainty, unresolved exceptions, and reason for each scope expansion. A model-generated certificate should never be able to elevate its own review status. The artifact already prevents its certificate from changing the note's annotation fields, but its input constructor remains available to application code; access-control and provenance guarantees require an enclosing system.

The repository contains only synthetic notes. Future real-record work must establish a lawful access basis and suitable privacy controls before ingestion. HHS describes Expert Determination and Safe Harbor as de-identification methods; neither a synthetic label nor removal of obvious names proves that a real record

satisfies the relevant requirements. Review the complete release surface, including quotations, exception logs, screenshots, and test failures. Keep identifiable records and reviewer identities out of a public research repository. This paper makes no claim of an implemented HIPAA-compliant service. [7]

12. Research boundary and credible expansion milestones

The central result is a small checked interface, not automated healthcare adjudication. It cannot diagnose a disease from symptoms, determine whether a provider established a condition, search an entire record, select a complete ICD-10-CM code, sequence multiple diagnoses, or infer payment. In particular, a valid clinical identifier does not establish covered service, medical necessity, contract applicability, or reimbursement amount. CMS explicitly distinguishes coverage decisions from coding and payment determinations; that separation should survive any future integration with the companion reimbursement work. [12]

A credible expansion sequence is to first strengthen evidence provenance and typed failure reasons, then validate the semantic annotation boundary, then implement a deliberately chosen classification slice with its dependencies. Each slice should add official-source tests, independent translation review, complete counterexamples for unsupported contexts, and an evaluation on records that were not used to formulate the rules. Shipping a larger dictionary before those steps would increase nominal coverage without demonstrating that more accepted decisions are justified.

A trained proposer can remain probabilistic while the final checker remains deterministic for fixed reviewed input and policy. That is an engineering separation of responsibilities, not a promise of deterministically correct real-world medicine. The value of the artifact lies in exposing those responsibilities precisely enough to test them. Until the proposed evaluation and broader classification work are executed, claims should remain limited to the two fixture concepts, twelve named statements, and twenty-three reproducible synthetic checks at the cited commit.

References

[1] FY 2027 ICD-10-CM Official Guidelines for Coding and Reporting. Updated October 1, 2026; Section IV outpatient guidance, including uncertain diagnoses; introduction on classification conventions and complete-record documentation. Context only; no complete translation is implemented. Accessed 2026-11-16.

[2] ICD-10-CM Files. CDC/NCHS publication page; FY 2027 October 1, 2026 release and service-date applicability through September 30, 2027. Accessed 2026-11-16.

[3] ICD-10-CM. CDC/NCHS description of the diagnosis and morbidity classification and official lookup entry point. Accessed 2026-11-16.

[4] ICD-10. CMS FY 2027 diagnosis-code file and guideline entry point; CM and PCS are separate published materials. Accessed 2026-11-16.

[5] ConText: An Algorithm for Determining Negation, Experiencer, and Temporal Status from Clinical Reports. Harkema et al., Journal of Biomedical Informatics, 2009. Primary study motivating separate contextual annotation tasks; no historical scores are transferred to this artifact. Accessed 2026-11-16.

[6] 2010 i2b2/VA challenge on concepts, assertions, and relations in clinical text. Uzuner et al., JAMIA, 2011. Primary shared-task report separating concept extraction, assertion classification and relations; cited for evaluation design, not current dataset availability. Accessed 2026-11-16.

[7] Guidance Regarding Methods for De-identification of Protected Health Information in Accordance with the HIPAA Privacy Rule. HHS guidance on Expert Determination and Safe Harbor; prospective data-governance context, not a compliance assessment of this synthetic project. Accessed 2026-11-16.

[8] The Lean Language Reference: Propositions. Version 4.35.0; propositions and proof terms. Logical checking must be distinguished from domain interpretation. Accessed 2026-11-16.

[9] Lean Clinical Coding: Model.lean at the pinned implementation. Primary implementation evidence: data types, Conditions, verifier, restricted mapping, calendar check and twelve named statements. Accessed 2026-11-16.

[10] Lean Clinical Coding: Main.lean at the pinned implementation. Primary fixture evidence: twenty-three named synthetic checks and compile-time evaluation; no real-record cohort. Accessed 2026-11-16.

[11] Formalizing all US case law in Lean. Lexifina, November 13, 2026. Architectural inspiration for separating fallible extraction from a formal certificate checker; its corpus and results are not reproduced. Accessed 2026-11-16.

[12] Medicare Coverage of Items and Services. CMS explanation distinguishing national coverage determinations from code assignment and payment amount; supports the clinical-versus-payment boundary. Accessed 2026-11-16.

Reproducibility record

Source: https://github.com/RevanthM/lean-clinical-coding | Evaluated implementation: f451edbe9054796d663b207d47e586f0b9c20346