Four Statuses for Claims About Social Reality

A bounded Lean formalization · technical note

Ryan Hunter, with AI research collaborators

June 2026 · public source edition September 2026

Abstract

Agents acting inside institutions need to distinguish what happened, what evidence supports a claim, what an authority recognized, and what remains in force. This note presents a Lean model of those four statuses and the precise sense in which each is necessary for a chosen class of questions. A credential example shows how an adjudication-only projection conflates a legitimate credential with a laundered one. The result establishes minimality for the named query class and a counterexample for one collapse, not universal superiority over every possible representation. The full Lean source accompanies this note.

1. The problem with a single status

Consider the sentence "this credential is valid." A database might store it as a Boolean. An agent might answer yes because the credential appears on an official list. Yet the underlying training may never have occurred, the evidence may be absent, or the credential may have been revoked since the list was generated. Each of those facts changes what the agent should do next.

The formalization separates four questions. Occurred: did the underlying act happen? Supported: is there evidence for it? Adjudged: did an authority recognize it? Operative: is it currently in force? An answer to one question cannot silently stand in for the other three. A claim can be officially recognized without support; it can be supported without having been adjudged; it can have been properly adjudged and later cease to operate.

The target is an agent answering consequential questions from institutional records. The purpose of a richer representation is practical: prevent the agent from treating an official stamp as evidence of an event, or an old decision as current authorization. The formal exercise asks what information the representation must retain for a stated set of queries.

2. Representation and query class

The Lean module projects a claim and its history into a FourStatus record. The four fields are Booleans named occurred, supported, adjudged, and operative. It also defines a deliberately lossy projection, collapsedRep, which keeps only adjudged.

The chosen query class contains five questions: whether the act occurred, whether evidence supports it, whether it was officially recognized, whether it is in force, and whether it has the laundering shape of being adjudged without support. The last question is a small but consequential composition of two fields. It distinguishes "there is an official stamp" from "the stamp has a backing record."

The term *optimal* in the source has a narrow meaning. The four-field representation is sufficient to answer this query class by direct projection, and every field is necessary for at least one query in that class. This says nothing about every question an organization could ask. A different question set might need additional dimensions, or fewer.

3. What the formalization proves

The source contains four load-bearing theorems. Each constructs two FourStatus values that agree on three fields but differ on an answer the fourth field supplies. If a representation discards occurred, it loses an occurrence query. Discard supported, and the laundering query can change while the other statuses stay fixed. Discard adjudged, and official recognition becomes unanswerable. Discard operative, and the in-force query becomes unanswerable.

This is an information argument, not a claim that all institutions should store exactly four bits. The counterexamples show that a projection omitting one dimension cannot answer the chosen class for every status assignment. A real institution also needs identities, provenance, times, policies, and the authority under which adjudication occurred. The four statuses are the status layer of a larger record.

The source then supplies a credential pair using histories from the FundamentalObject module. One history represents a legitimate credential and the other a laundered one. The theorem adjudged_projection_conflates_laundering states that the adjudication-only projection returns the same answer for the pair while the four-status projection separates them. The theorem detector_catches_laundering evaluates the laundering predicate on the same pair: it flags the unsupported adjudication and clears the legitimate claim.

Those results are about one named collapse and one worked pair. They do not show that no clever compressed representation could preserve the query answers. The source explicitly draws that boundary. The example exposes a common operational error: recognition was used as a substitute for evidence.

4. Reading the Lean source

The companion Lean file contains the definitions, theorem statements, proof terms, and runnable examples. It imports StandardKernel.FundamentalObject and StandardKernel.Kernel.Status4, so it is a module of that larger formalization rather than a standalone one-file project. This edition preserves the source for inspection; this repository does not bundle the imported Lean project or claim an independent build of it.

The compact definitions at the center of the module are:

```
def collapsedRep (h : History StringKernel)
    (c : Claim StringKernel) : Bool :=
  (commonsRep h c).adjudged

def isLaundered (st : FourStatus) : Bool :=
  st.adjudged && !st.supported
```

The first definition is the information loss under examination. The second is a question that loss makes hard to answer. The remaining theorems connect these definitions to concrete status pairs and histories.

5. Consequence for agent systems

An agent should not have to reconstruct these distinctions from prose every time it acts. They belong in the shared record and in the interface through which the agent reads it. If a system only returns “approved,” the agent cannot know whether the act happened, evidence was checked, an authority recognized it, or the authorization still applies. Better language modeling will not recover a field that the system discarded.

The design test is simple: take one consequential question a user might ask, then list every distinction that could change the answer or the permitted next action. If two materially different histories collapse to the same value, the agent needs a richer record, a narrower question, or an explicit escalation. This note makes one such failure concrete and machine inspectable.

References and source

1. Ryan Hunter, *Atlas Commons Epistemics*, Lean module, June 2026. The public source edition accompanies this note as a Lean file; the original is in Foundation3.
2. Lean documentation, *Theorem Proving in Lean 4*, for the language and proof style used by the source.

Source edition note. This note is a readable carrier for a bounded formalization. It preserves the original module’s scope and links the source directly. It has not been submitted to arXiv or independently peer reviewed.