The belief failure taxonomy

Working paper 1 in a series on the integrity of AI-served knowledge

Treb Gatte, Continuous Logic, August 2026
treb@continuouslogic.ai

Your AI assistant can quote the current policy correctly and still give the wrong answer. It can also retrieve an exact value from a database, call a live API, read a configuration table, or cite a calculated metric and still tell someone something the organization should not act on.

The source does not have to be a document for the failure to occur.

The value may be stale. Two authoritative systems may disagree. The data may never have been approved for that use. A derived value may remain after the system or rule that justified it has disappeared.

Those are four different failure mechanisms, and none requires the model to invent a fact. This paper calls them belief failures:

- Stale
- Contested
- Unauthorized
- Orphaned

"Belief" is operational shorthand, not a claim about machine consciousness. It means a proposition the system is prepared to assert as true to a user or downstream process.

The point of naming the failures separately is practical. Each is created by a different condition, detected by a different test, and corrected at a different layer.

***"Belief"** is operational shorthand for a proposition the system is prepared to assert as true to a user or downstream process*

The source is not necessarily a document

It is easy to mistake this problem for document governance because documents make the failure visible. A policy has a version. A procedure has an effective date. A draft can be marked draft. A superseded document can often be identified by a human looking at it.

Structured data is less forgiving.

Consider the kinds of information an AI system may use to answer a question like:

- Customer status stored in a CRM
- Discount threshold stored in a pricing table
- Employee's manager returned by a directory service
- Eligibility flag returned by an API
- Risk score calculated from several systems
- Business rule stored in configuration
- Inventory value replicated from an operational database
- Metric produced by a semantic model
- Feature flag controlling application behavior
- Cached result derived from a source that has since changed
- Meeting summary generated by an AI

Each can be completely real and can be retrieved correctly. However, each can still support an assertion the organization would reject.

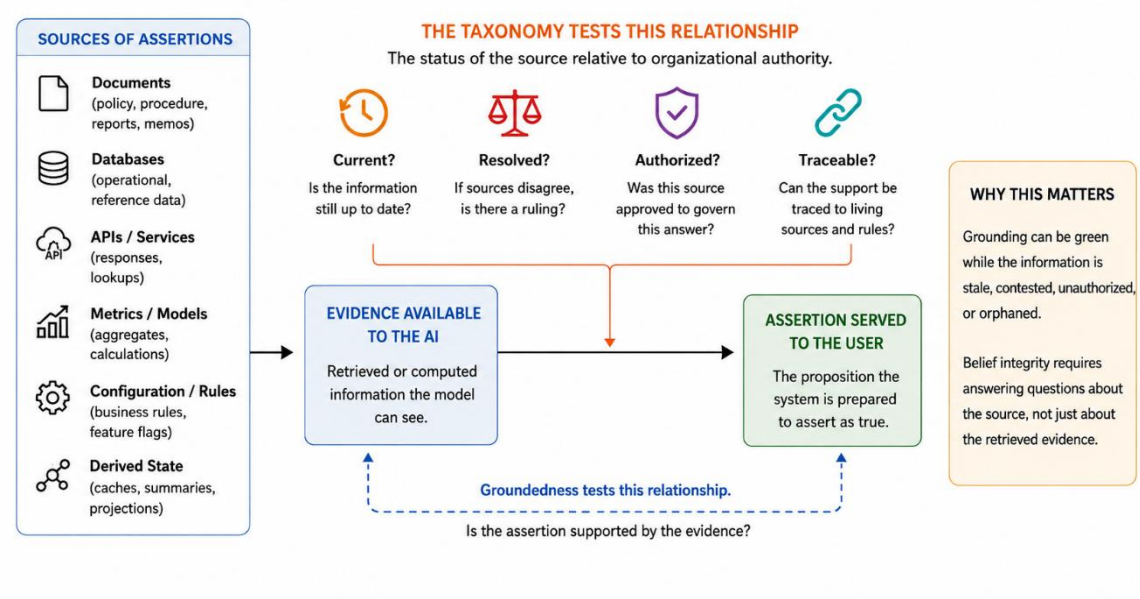

Figure 1 The Relationship Grounding Does Not Test

Better document or data management therefore only addresses one part of the problem.

The deeper issue is whether the system knows the status of the proposition it is about to assert. Is the underlying value still current? Does another authoritative source disagree? Is this source allowed to govern this question? Can the assertion still be traced to the data, rule, or process that justified it?

Those questions apply regardless of whether the evidence arrived as prose, a database row, an API response, or a computed value.

Two different kinds of failure

When an assistant gives a wrong answer, the reflex is to call it a hallucination. The word has become shorthand for "the AI was wrong," and that shorthand makes several operational failures harder to see.

For this paper, hallucination means a generation failure: the model produces content at answer time that is not supported by the material available to it. The exact behavior can be prompt-sensitive or stochastic, so the fabrication may not recur on the next attempt. Groundedness checks, citation verification, and answer evals are designed in large part to detect this class of problem.

Belief failures are different. The system can faithfully report something its available evidence supports while the organization would still reject the assertion as stale, unresolved, unapproved, or unaccountable. The model did not need to invent anything.

Retrieval may have worked exactly as designed. The SQL query may have returned the row that actually exists. The API may have returned HTTP 200 and a perfectly valid response. The metric may have been calculated correctly from its inputs. The cited document may be real.

The failure sits in the status of what was retrieved and that distinction matters for two reasons.

First, the underlying condition persists. A stale value remains stale until something changes it or its authority. A conflict remains unresolved until someone rules on it. An unauthorized source remains unauthorized until its role changes. An orphan remains untraceable until provenance is restored or the assertion is retired.

The exact answer may vary from query to query, especially in a contested case, but the failure condition remains.

Second, these failures can pass checks that only ask whether an answer is supported by retrieved evidence. Source support is necessary, but it is not the same as organizational authority.

The four mechanisms can also overlap. An unauthorized data source can later become stale. A stale derived metric can become orphaned after its calculation pipeline is retired. The taxonomy is not meant to force every incident into one exclusive box. It identifies the missing control that allowed the system to assert something it should not.

Stale: the answer outlived its truth

A stale belief was correct at some earlier point. Then reality changed and the system continued serving the old value.

The document example is straightforward. A company revises its expense policy, lowering the meal per diem from $75 dollars to $50. The new policy is published and indexed. However, the previous policy remains retrievable. An employee asks what the per diem is, and the assistant serves $75 dollars with a citation.

The same failure can also exist entirely in structured data. Suppose a customer receives a new service tier on Monday. The customer master table in Fabric is updated immediately, but the Power BI semantic model is refreshed nightly. An assistant is asked whether the customer qualifies for premium support and it answers using a Fabric data agent querying the semantic model source. It returns the old tier accurately from the semantic model it was given.

The query worked and the row existed. The answer is supported by that row, but the assertion is wrong now.

Another version appears in derived data. A pricing calculation uses a discount threshold of $100,000 dollars. The threshold changes to $125,000 in the authoritative pricing system, but a downstream rules table or cached calculation still contains the old number. An AI system reading that derived state can faithfully reproduce an obsolete business rule.

The common mechanism is not an old document or record. It is **superseded state that remains assertable**.

Supersession can occur between document versions, database records, API versions, configuration values, business rules, calculated measures, or replicated system state. If the relationship between the old state and the replacement is not represented or enforced, retrieval alone cannot determine which one should govern now.

Organizations manufacture staleness continuously through normal operations. Every policy revision, status change, price update, organizational change, master-data correction, configuration change, and business-rule update creates the possibility that an older representation remains available somewhere else. Sometimes these changes arise from spot decisions, becoming de facto operations policies.

The useful quantity is the time between when an assertion stopped being valid and when the organization stopped serving it. In many AI deployments, that quantity is not measured.

Contested: two answers, no ruling

A contested belief exists when two credible sources make incompatible claims within the same scope and no recorded ruling determines which one governs.

Consider a global data retention standard that says seven years and a regional operations manual that says five. Both appear current. Both appear applicable to the same regional records. Neither explicitly supersedes the other.

That is a document conflict. Now consider the structured equivalent.

The customer master system classifies an account as “enterprise.” The billing platform classifies the same account as “mid-market.” Both systems are operational, both values are current in their respective databases, and different business processes depend on each classification.

A user asks the assistant, “Is this an enterprise customer?” There is no retrieval result that can solve the underlying problem.

A similar conflict can occur when a finance system reports one revenue value and a sales system reports another, when a directory service and HR system disagree about someone’s reporting line, or when an API configuration says a customer is eligible for a service while the entitlement database says they are not.

Both values can be real. Both sources can be authoritative for something.

The unresolved question **is which source has authority for this particular assertion, within this particular scope, at this particular time.**

Unless the system detects and handles that conflict, it may still return one answer. Which answer wins can depend on retrieval ranking, source priority, synchronization order, query design, prompt phrasing, or application logic. That is not adjudication, it’s an implementation detail selecting a winner.

Human organizations resolve these situations by defining system ownership, scope, precedence, exception handling, or explicit unresolved status. An AI system needs some equivalent way to recognize that disagreement exists rather than silently converting retrieval order into organizational truth.

Until that happens, the defensible answer may be that the sources disagree and no ruling has been recorded.

The unresolved question is which source has authority for this particular assertion, within this particular scope, at this particular time.

Unauthorized: nobody approved this

An unauthorized belief comes from information the system can access but that nobody approved as a basis for the assertion being made.

Unauthorized does not mean the user necessarily lacked permission to see the information. It means the information was never authorized to govern this question.

A document example is an old draft pricing proposal sitting in a shared drive. A salesperson asks about volume discounts, and the assistant answers from the draft rather than the published price book.

The structured-data equivalent can be harder to notice. Suppose an analyst maintains an experimental customer scoring table while developing a new segmentation model. The database account used by an assistant has permission to query the table. No one has approved those experimental scores for operational use.

A user asks, “Is this customer at risk?” The assistant retrieves a score of 0.82 and answers yes. The value exists and the calculation may even be correct according to the experimental model.

The failure is that nobody decided this model output was allowed to become an organizational assertion.

The same problem can occur with:

- Sandbox database tables
- Test API endpoints
- Provisional forecasts
- Analyst-created calculated fields
- Manually overridden records
- Abandoned business rules
- Development configuration
- Intermediate machine-learning outputs
- Local team classifications that were never adopted enterprise-wide

In many systems, technical accessibility becomes de facto authority. If retrieval can reach a value, that value can potentially become evidence for an answer.

But access and authority are not the same thing.

Organizations already understand this distinction elsewhere. A database may contain staging tables that are not production facts. A finance model may contain scenarios that are not forecasts. A CRM may contain salesperson estimates that are not contractual commitments.

An AI layer can erase those distinctions if all retrievable values are allowed to speak with the same voice. The useful question is therefore not just "why could the assistant access this?" It's "why was this source allowed to answer that question?"

Orphaned: the justification is gone, the assertion remains

An orphaned belief is an assertion whose supporting provenance can no longer be resolved.

The original document may have been deleted, purged due to a retention policy. The database table may have been retired. An API may have been decommissioned. A calculation may remain after the business rule that produced it was removed. A derived value may survive after its upstream dependencies are gone.

Consider a legacy process document that is deliberately retired during a platform migration. A summary derived from that document survives in an index or cache. Months later the assistant still serves the old process, but the original source can no longer be inspected.

Now consider the structured-data version. A risk classification is calculated from three operational systems and written into a downstream analytics table. Six months later one upstream system is retired and the calculation pipeline is replaced. The old risk classifications remain in the downstream table.

An assistant asks the table for the customer's current risk category and receives "high." The value exists but what no longer exists is a reproducible explanation of why that customer was classified as high.

The same failure can occur with a KPI whose calculation definition has disappeared, a cached eligibility result produced by an API that no longer exists, or a customer segment whose original rule set was replaced without recomputing historical derived state.

An orphaned assertion is not necessarily false. It may still be correct but the failure is accountability.

If an assertion cannot be traced back through its dependencies to living evidence and rules, it cannot be fully inspected, reproduced, corrected at the source, or defended when challenged.

More layered architectures create more opportunities for this failure. Every replica, cache, transformation, feature store, summary, embedding index, calculated column, event projection, and derived record can preserve some representation of a proposition after its original support changes or disappears. Many AI projects are combining data from many systems that have never been combined before, creating many new opportunities for this type of failure.

Deleting or retiring the origin does not necessarily delete everything that learned from it. The relevant test is therefore not simply whether the system can point to a stored value. It is whether the chain that justifies that value still resolves.

	STALE *The answer outlived its truth*	CONTESTED *Two answers, no ruling*	UNAUTHORIZED *Nobody approved this*	ORPHANED *The justification is gone, the assertion remains*
What the condition is	A superseded value remains available alongside the current value.	Two credible, live sources make incompatible claims in the same scope.	A source is accessible but was never approved to govern this answer.	The support for an assertion can no longer be traced to living sources and rules.
Example from documents	Old policy version still in the library after a new version was published.	Global policy says 7 years, regional policy says 5.	Draft pricing proposal on a shared drive.	Retired process document summary still in the index after the source was deleted.
Example from structured data	Customer tier changed in the master system, reporting database still shows the old tier.	Customer master says 'Enterprise,' billing platform says 'Mid-Market.'	Experimental risk scoring table queried in production.	Risk classification remains in a table after the upstream calculation pipeline was retired.
What the assistant sees	A real value from a real source.	Two real, credible values.	A real value from an accessible source.	A stored value with incomplete or broken provenance.
The missing decision	Which version governs now?	Which source has authority for this question?	Was this source approved to govern this answer?	Can this assertion still be traced to living evidence and rules?

Figure 2 The Four Belief Failure Mechanisms, across Source Types

Why grounding alone is not enough

A large part of the AI answer-quality stack asks a necessary question: is the output supported by the evidence returned to the model? That evidence may be text, structured data, a tool result, or a calculated value.

The question is still incomplete, however.

- A stale answer can be supported by an old database row or superseded document.
- A contested answer can be supported by either of two live systems.
- An unauthorized answer can be supported by a perfectly valid sandbox table or provisional model output.
- An orphaned answer can be supported by a stored derived value even though the value's original dependencies can no longer be reconstructed.

In each case, the evidence can support the answer while still failing a different test.

- Is it current?
- If credible sources disagree, which one governs this assertion?
- Was this source authorized for this use?
- Can the assertion still be traced to living evidence and rules?

Grounding tests the relationship between an answer and the evidence supplied to generate it. Belief integrity requires another relationship to be tested as well that being the relationship between that evidence and the organizational reality it is supposed to represent.

That relationship does not belong to document management nor to database management. It crosses document systems, transactional databases, analytical systems, APIs, configuration, metrics, rules, and derived state.

A green groundedness score therefore does not, by itself, tell you whether the answer reflects the organization's current, resolved, authorized, and traceable position.

What the four have in common, and where they differ

The common structure is simple.

- Something became part of what the system could assert without a corresponding determination that it should still be asserted.
- A value changed and an older representation remained available.
- Two sources disagreed and no authority resolved the conflict.
- Information became retrievable and nobody approved it as operational.
- A source or dependency disappeared while a derived assertion survived.

The differences make the taxonomy actionable:

- Stale failures are detected by identifying changed or superseded state and testing whether an older representation remains assertable. The relevant control is temporal authority and supersession handling.
- Contested failures are detected by identifying incompatible live claims within overlapping scope and testing whether the system exposes the disagreement or silently selects one. The relevant control is precedence, adjudication, or explicit unresolved status.
- Unauthorized failures are detected by tracing assertions to their sources and determining whether those sources were approved to govern that type of answer. The relevant control is authority, not mere technical access.
- Orphaned failures are detected by walking the provenance and dependency chain behind an assertion and verifying that its supporting evidence and rules still resolve. The relevant control is dependency-aware provenance.

Four mechanisms, four detection procedures, four different controls.

Calling all of them hallucination does more than mislabel the problem. It points remediation at the wrong layer. For example.

- Better prompting cannot decide whether the customer master or billing system governs customer classification.
- A hallucination detector cannot determine whether an experimental scoring table was authorized for production decisions.
- Better document versioning cannot tell you whether a cached API response is stale.
- A citation cannot make an unreproducible derived value accountable.

A claim you can check this afternoon

Papers in this series will close with something a skeptic can verify without trusting the author.

Choose several facts in your organization that have changed recently.

Do not limit the sample to documents. Include at least one value from structured data if your assistant can access it. Useful examples include a customer classification, pricing threshold, employee assignment, entitlement, business rule, metric definition, or operational status.

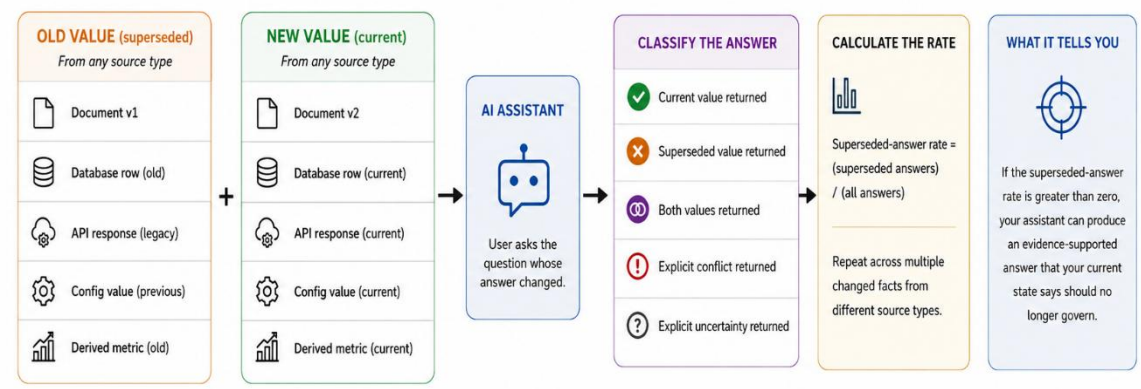

Figure 3 A Source-Agnostic Test You Can Run Today

For each fact, identify the previous value and the value that currently governs.

Then ask the assistant the question whose answer changed.

Record whether it returns:

- the current value
- the superseded value
- both values
- an explicit conflict
- an explicit uncertainty

Repeat the test across a small sample and calculate the rate at which superseded values are still served as valid answers.

The claim is not that it will serve the old answer. On any single pair it may well get it right. The claim is narrower and harder to dismiss as your corpus contains many such pairs, some fraction of them resolve to the superseded answer, and you do not currently know that fraction, because nothing in your deployment measures it or is responsible for it.

If you run the test and the assistant gets your first pair right, run the next one. The interesting number was never one answer. It is the rate, and right now nobody has it.

A machine that makes things up is a quality problem, and an entire industry is working on it. A machine that believes everything your organization ever wrote down, and nothing it has since changed its mind about, is a different problem, and nobody is working on it.

All the vigilance went to the first machine. You deployed the second.

