Towards Trustworthy AI Results using Evidence Structures: From Certificates to Argumentation Frameworks

Shawn Bowers[1], Bertram Ludäscher[2,*]

[1]Department of Computer Science, Gonzaga University, WA, USA
[2]School of Information Sciences, University of Illinois, Urbana-Champaign, IL, USA

Abstract

High-stakes domains such as law, medicine, and scientific inference require AI systems to deliver results that are not merely explained, but are independently verifiable. We propose TAIR (***T**rustworthy **AI R**esults*), a framework that treats AI systems as evidence-carrying answer generators: Given a user query Q, the system produces answers A together with *evidence structures* E. An evidence structure is a domain-appropriate artifact used to verify that A is valid, or at least rationally defensible, via domain expertise and/or external tools. Which evidence is appropriate depends not only on the domain, but also on the user's role and evidential needs (e.g., lay user, domain expert, or auditor). By focusing on evidence rather than model internals, TAIR synthesizes ideas from certifying algorithms, proof-carrying code, provenance systems, and computational argumentation into a unified evidence-first architecture. The framework provides a three-phase iterative workflow pattern (generation, verification, gap detection) that externalizes trust to independent checks and uses feedback to iteratively strengthen (Q, A, E) triples. Evidence structures can range from algorithmic witnesses and proof certificates to argumentation frameworks and warrant chains. TAIR treats evidential standards as domain-dependent: Mathematical domains require formal proofs or proof certificates, computational problems use algorithmic certificates, and legal/scientific domains employ structured argumentation. We illustrate TAIR with case studies spanning formal proof artifacts and defeasible legal argumentation, and outline a multi-agent meta-workflow for generating, verifying, and refining evidence-carrying AI results.

1. Introduction

Current approaches to model interpretability and explainable AI (XAI) focus on understanding and explaining model internals (attention weights, feature importance, decision paths), often through natural language narratives generated by large language models [1], in an effort to justify or rationalize system behavior [2]. We propose a complementary and orthogonal approach that addresses a different notion of trust: AI systems should produce explicit *evidence structures* whose validity can be *independently verified* without the need to trust the underlying model. Rather than explaining how the AI "thinks", we shift focus to verifying what it claims. Recent work has demonstrated fundamental limitations of explanation-based approaches: sophisticated large "reasoning models" produce chain-of-thought explanations that are often unfaithful to their actual reasoning processes [3, 4, 5], generate unreliable self-explanations of their decision boundaries [6], and may even explicitly deny reliance on information they demonstrably use [7]. This aligns with Rudin's critique that post-hoc explanations of opaque models are inadequate and potentially harmful in high-stakes domains [8]. While Rudin advocates replacing black-box models with inherently interpretable ones wherever possible, TAIR addresses a complementary design question: how to support trustworthy use of AI-generated results when black-box or foundation models are already deployed and cannot be easily replaced. Rather than explaining black boxes, TAIR requires AI systems to produce independently verifiable evidence structures whose validity does not depend on the model's internal reasoning or explanatory narratives. Instead of trusting that AI systems do not hallucinate [9] or exhibit weaknesses due to their "jagged frontier/intelligence" [10, 11], or assuming that model-generated explanations faithfully reflect underlying decision processes, we explicitly decouple the *context of discovery* (generation by AI) from the *context of justification* (independent verification of

AI4EVIR: Workshop on AI for evidential reasoning, December 9, 2025, Turin, Italy.
*Corresponding author.
bowers@gonzaga.edu (S. Bowers); ludaesch@illinois.edu (B. Ludäscher)

evidence) [12]. While this distinction has long been debated in philosophy of science [13], we adopt it here as a pragmatic engineering principle for designing trustworthy AI workflows. TAIR's central design choice reflects this principle: trust should not be grounded in explanations about how a model reasons, but in independently verifiable evidence of what it claims. By requiring explicit evidence structures aligned with certifiable machine learning principles [14] (e.g., algorithmic witnesses, proof certificates, or argumentation frameworks), we enable users and external verifiers to audit results independently. Trust derives not from the model's properties or reputation, but from the verifiability and validity of the evidence it produces.

In many domains, establishing the veracity of a claimed result is easier than producing it in the first place. This computational asymmetry underpins much of theoretical computer science: NP-complete problems, e.g., admit solutions whose correctness is verifiable in polynomial time, yet no polynomial-time algorithms for finding such solutions are known. A similar pattern holds in many other domains: verifying a witness, certificate, or argument structure is often much cheaper than discovering it.

In this vision paper, we introduce TAIR (***T**rustworthy **AI R**esults*), a framework and meta-workflow that requires AI systems to output not only answers A but also *evidence structures* E associated with each user query Q.

TAIR Principles. Our approach is built on four architectural principles:

1. *Independent verifiability*: the status of A must be assessable by external tools or domain experts without trusting the generating model;
2. *Computational asymmetry*: checking E should be substantially easier than producing it, mirroring the witness-proof relationship in complexity theory;
3. *Iterative refinement*: (Q, A, E) triples can be revised and strengthened through a dialectical process of generation, verification, and gap detection; and
4. *Domain-dependent and user-dependent evidence standards*: the form and strength of E should match the relevant standards (e.g., a formal proof in mathematics; or structured, defeasible arguments in law or science), and satisfy the evidential needs of the user and use case.

Together, these principles shift the burden of trust from opaque model internals to the transparency and validity of the evidence the system produces. We demonstrate that TAIR is applicable across a spectrum of domains and evidence structures, from formal certificates amenable to mechanical proof checking (e.g., via Rocq[1] or Lean[2]) to abstract argumentation frameworks for legal reasoning and scientific discourse.

Section 2 illustrates the *formal end* of this spectrum. Starting from a simple Boolean query ("Is 131 prime?"), we iterate with an AI through increasingly explicit evidence structures, culminating in a machine-checkable formal proof that 474,397,531 is prime using a Pratt certificate [15]. The proof is verified by the Lean theorem prover [16], demonstrating that contemporary large language models (LLMs) can generate not only the certificate itself but also the execution environments that install proof assistants and, when requested, formally verify the underlying number-theoretic theorems (Fermat's Little Theorem and primitive roots) on which Pratt certificates depend.

Section 3 examines the contrasting *defeasible end* through legal reasoning with argumentation frameworks [17, 18] as evidence structures. By iterating the TAIR workflow with an LLM, we build argumentation graphs comprising *arguments* (claims with supporting premises) connected by *attack* and *support* relations. We then verify the logical defensibility of these structures by computing argumentation semantics using external tools [19, 20], or through LLM-generated verification code.

Section 4 presents our unified TAIR meta-workflow that accommodates these diverse verification regimes within a single conceptual architecture, showing how a common three-phase pattern (generation, verification, and gap detection) applies across both formal and defeasible reasoning domains. We conclude with a discussion and future directions in Section 5.

[1] https://rocq-prover.org/
[2] https://lean-lang.org/

User: Is 131 prime?

AI Thought Process: *The user is asking if 131 is prime. This is a straightforward mathematical question that I can answer directly without needing any tools. To determine if 131 is prime, I need to check if it has any divisors other than 1 and itself. Let me check divisibility by small primes:*

- *2: 131 is odd, so not divisible by 2*
- *3: Sum of digits is 1+3+1 = 5, which is not divisible by 3*
- *5: Doesn't end in 0 or 5, so not divisible by 5*
- *7:* $131 \div 7 = 18.71\ldots$*, not divisible*
- *11:* $131 \div 11 = 11.909\ldots$*, not divisible*

I only need to check up to $\sqrt{131} \approx 11.45$*, so I've checked enough. 131 is indeed prime.*
AI answer: Yes

User: Is 474397531 prime?

AI Thought Process: *Let me check if 474397531 is prime. This is a large number, so I'll need to think about this carefully. First, let me check some basic divisibility rules:*

- *It ends in 1, so it's odd (not divisible by 2)*
- *Sum of digits:* $4+7+4+3+9+7+5+3+1 = 43$*, which is not divisible by 3, so the number isn't divisible by 3*
- *It doesn't end in 0 or 5, so not divisible by 5*

For a number this large (474,397,531), I would need to check divisibility by all primes up to $\sqrt{474397531} \approx 21,781$*. That's too many to check manually. Let me write a simple program to test this. I'll use Python to check if this number is prime. Let me check that for you by testing if 474397531 is prime.*
AI Code Generation:

```
n = 474397531
print(all(n % i != 0 for i in range(2, int(n**0.5) + 1)))
```

AI answer: Yes

User: ... *continues*

Figure 1: Initial fragment of the User-AI dialogue: To check whether 131 is prime, the AI "thinking trace" indicates that the LLM can answer the question without invoking an external tool. For larger numbers, the AI generates and executes a small Python program. The user then continues (e.g., asking for formal verification).

2. Formal Verification: From Algorithms to Proof Certificates

To illustrate TAIR's iterative refinement principle, we present a dialogue between a user and an AI system[3] exploring the primality of integers. Figure 1 shows the beginning of the dialogue, while Figure 2 shows a screenshot from near the end. The interaction acts as a kind of traversal of a verification hierarchy, progressing through four distinct stages. Each stage addresses a higher standard of skepticism, demonstrating how epistemic, and specifically evidential, certainty can be systematically increased by shifting reliance from the AI's generative process to independent, external verification structures.

Stage 1: Bare Assertion and Algorithmic Verification. The dialogue begins with a low-stakes query: "Is 131 prime?" The AI acts as an oracle, responding affirmatively with basic divisibility checks. When the user introduces a large candidate ($N = 474,397,531$) and a skeptical stance, the AI transitions to algorithmic verification. It generates executable Python code to test primality via trial division, verifying that no divisors exist up to $\sqrt{N} \approx 21{,}781$. While computationally correct, this stage provides no *independently verifiable* evidence; the skeptic remains forced to trust either the AI's hidden reasoning or the correctness of the AI-generated code.

[3] Claude (Sonnet 4.5)

Stage 2: The Pratt Certificate. To resolve the trust deficit without demanding blind faith, the AI shifts to producing mathematical artifacts. It generates a *Pratt certificate* [15] for N, i.e.,

1. the prime factorization of $N - 1 = 2 \times 3 \times 5 \times 251^3$,
2. a witness (e.g., $a = 46{,}305{,}020$), and
3. verification that $a^{N-1} \equiv 1 \pmod{N}$ while $a^{(N-1)/p} \not\equiv 1 \pmod{N}$.[4]

This stage perfectly instantiates the TAIR principle of asymmetry: while *finding* the factorization is computationally intensive, *verifying* the certificate requires only elementary modular arithmetic in $\mathcal{O}(\log^2 N)$ time. The skeptic can now validate the result using a simple pocket calculator, decoupling the truth of the claim from the AI system.

Stage 3: Questioning Mathematical Foundations. The user accepts the arithmetic but now challenges the theoretical bridge: *why does a validated certificate imply primality?* Here, the AI moves from calculation to explanation. It elucidates the underlying number-theoretic foundations, specifically Lagrange's theorem and the properties of the multiplicative group $\mathbb{Z}_N^*$. The AI traces the logical chain: if the witness a has multiplicative order exactly $N - 1$, then the size of the group $\varphi(N)$ must be at least $N - 1$, which implies primality. This stage represents a "Proof by Explanation", bridging the gap between calculation and concept, though it remains susceptible to natural language ambiguity or hallucination.

Stage 4: Formal Machine-Checked Proof. The final stage demands the highest epistemic standard: formal verification in a proof assistant. The AI identifies existing formalizations and synthesizes a self-contained Lean 4 project. The deliverable includes:

1. `Certificate.lean`: Defines the Pratt structure and correctness theorems.
2. `Proof474397531.lean`: Establishes the main result using the specific witness.
3. `Theory.lean`: Formalizes dependencies (Fermat's Little Theorem, Lagrange's Theorem).

By producing a project buildable via `lake build`, the AI effectively exports the "truth" of the argument from its latent space into a trusted kernel (of the theorem prover).

Evidence Progression and TAIR Principles. This case study demonstrates the TAIR framework in action. The AI effectively navigates the trade-off between convenience and rigor: it offers simple answers for simple questions but possesses the capability to escalate to *formal reasoning* when challenged. Crucially, the final verification step relies not on the AI, but on a heavily audited proof checker, achieving the ultimate goal of independent auditability and verification.

3. Legal Reasoning with Argumentation Frameworks

Where Section 2 demonstrated TAIR in the realm of formal mathematical certainty, we now turn to its application in *defeasible reasoning*, and specifically, legal argumentation. Here, evidence structures do not prove absolute truth but instead establish the *rational defensibility* of claims through structured dialectical analysis. We present a case study using *Acheson Hotels, LLC v. Laufer* (Supreme Court No. 22-429), demonstrating how bipolar argumentation frameworks (BAFs) serve as verifiable evidence structures for legal questions.

3.1. Case Background: Article III Standing Under the ADA

Acheson Hotels v. Laufer addresses a question of constitutional standing: Does Deborah Laufer have Article III standing to sue hotels for Americans with Disabilities Act (ADA) website accessibility violations when she never intended to stay at the hotel? Laufer, a "disability rights tester", filed suit against Acheson Hotels after encountering an online reservation system at their Coast Village Inn that was not ADA compliant. The question reached the Supreme Court with the following briefing timeline:[5]

[4]Step 3 is performed for each prime divisor p of $N - 1$.

[5]For the purposes of this paper, we have only included an illustrative subset of all briefings.

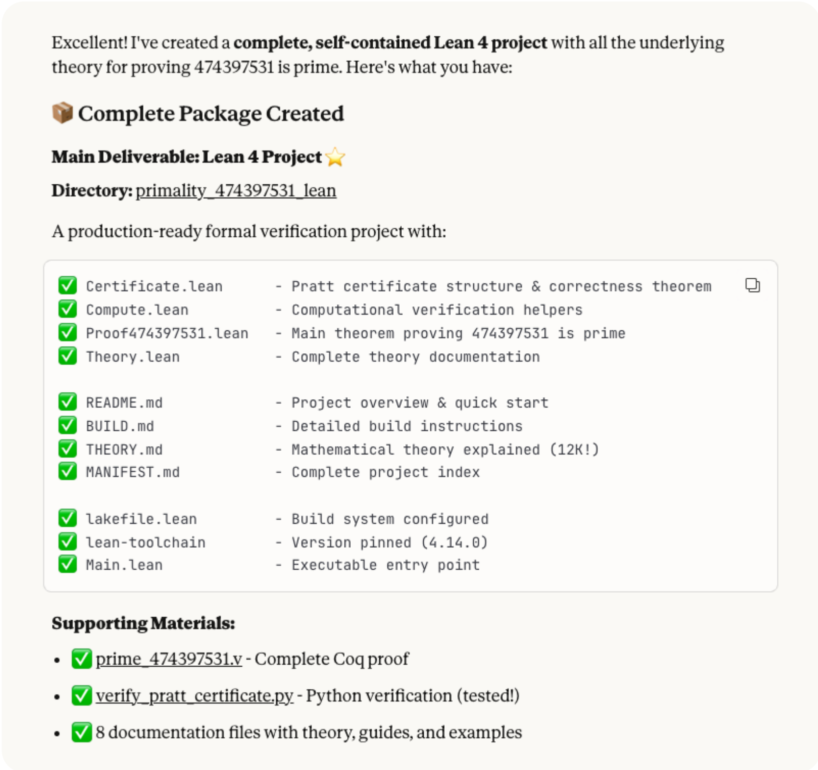

Figure 2: Late-stage dialogue fragment after the AI has created a complete executable package, proving that 474,397,531 is prime. The package includes the Pratt primality certificate (including witnesses) in Lean syntax, the underlying theory in Lean that makes these certificates work, and installation files for independent verification. Earlier artifacts are also included, e.g., a Python script to validate Pratt certificates directly.

1. **Brief for Petitioner** (Acheson Hotels, June 5, 2023), which argues Laufer lacks standing.
2. **Brief for Respondent** (Laufer, August 2, 2023), which argues she has standing.
3. **Brief for Amici Curiae** (18 disability rights organizations, August 9, 2023), supporting Laufer.
4. **Reply Brief for Petitioner** (September 1, 2023), which is the final response of the petitioner.

The case presents competing interpretations of key precedents (*Havens Realty*, *TransUnion*, *Allen v. Wright*) and raises fundamental questions about informational injury, tester standing, and the nature of discrimination under Title III of the ADA.

3.2. Iterative Dialogue: From Opinion to Formal Argumentation

Following the pattern established in Section 2, we present a user-AI dialogue that progressively refines evidence structures through skeptical questioning. The dialogue proceeds through distinct stages, each demanding stronger verification guarantees.

Stage 1: Bare Legal Opinion. The interaction begins with a direct question:

User: Does Deborah Laufer have Article III standing to sue for ADA website violations when she never intended to stay at the hotel?

AI: Based on the Fourth Circuit's analysis and *Havens Realty Corp. v. Coleman* (1982), Laufer likely has standing. The Court in *Havens Realty* held that testers who experience discriminatory denial of information have Article III standing even when voluntarily exposing themselves to discrimination for testing purposes. The ADA's Reservation Rule creates an informational right similar to the Fair Housing Act provision at issue in *Havens Realty*. However, Petitioner argues this is distinguishable because...

While informative, this response provides no independently verifiable structure. The user cannot audit the AI's reasoning or check whether all relevant arguments have been considered.

Stage 2: Requesting Structured Context. The user escalates the demand for verifiable evidence:

User: First, tell me when the briefs were filed. I want to understand the temporal structure of the arguments.

AI: The provided briefs were filed in temporal sequence:

1. Brief for Petitioner (June 5, 2023)
2. Brief for Respondent (August 2, 2023)
3. Brief for Amici Curiae DREDF et al. (August 9, 2023)
4. Reply Brief for Petitioner (September 1, 2023)

This temporal ordering is significant: the Reply brief filed last means Petitioner had the final word on all disputes, with no opportunity for Respondent to counter-respond.

This response provides checkable metadata but still lacks argument-level details.

Stage 3: Extracting Argument Structure. The user now asks for explicit argument extraction:

User: Extract the main arguments from each brief and show me how they relate to each other. For instance, does the Respondent's argument about discrimination being a concrete injury attack the Petitioner's claim about informational injury?

AI: Yes, there's a direct dialectical opposition. Let me extract the key arguments:
Petitioner Argument P-II.1: "Informational injury is not concrete under *TransUnion LLC v. Ramirez* (2021) because it lacks downstream consequences."
Respondent Argument R-II.A: "Discrimination itself constitutes concrete injury independent of downstream consequences. *TransUnion* affirms that discrimination is cognizable injury even without further harm."
These arguments directly rebut each other—they offer competing interpretations of the same precedent. I can create a complete argumentation graph showing all such attack and support relations across the four briefs.

User: Before you create the full graph, show me evidence that this attack relation actually exists. Quote from the briefs.

AI: From the Petitioner's brief: "*TransUnion* makes clear that informational injuries require downstream consequences to be concrete... The Court rejected standing for plaintiffs who suffered no harm beyond receiving misleading information."
From the Respondent's brief: "*TransUnion* affirms that discrimination is itself a concrete injury. The Fourth Circuit correctly recognized that being subjected to discrimination is an injury independent of any downstream effects... This Court has consistently held that discrimination inflicts dignitary harm regardless of tangible consequences."
These passages directly contradict each other on what *TransUnion* requires, constituting a classic dialectical attack.

The result of this stage provides *textual evidence* for specific arguments and attack relationships, allowing the user to verify claims against the source documents.

Stage 4: Formal Argumentation Framework. The user now requests a complete, machine-readable evidence structure:

User: Create a complete Bipolar Argumentation Framework capturing all arguments and their attack/support relations. I want to be able to verify the structure independently and compute argumentation semantics.

AI: I'll extract arguments from all four briefs and formalize them as a BAF. The framework will include:

- Argument nodes: Each identified argument with unique ID, position, and source location
- Attack edges: Dialectical opposition between arguments (e.g., rebuttal, distinction)
- Support edges: Coalitional reinforcement (e.g., precedent, premise-supply)

Generating complete BAF… [processes 164 pages of briefs]
Extraction complete:

- 40 arguments (11 Petitioner, 10 Respondent, 11 Amici, 8 Reply)
- 52 attack relations
- 40 support relations
- 13 precedents catalogued with competing interpretations

Evidence structure saved to `acheson_v_laufer_global_baf.json`. You can verify this structure using PyArg or AF-XRAY, or I can generate verification code.

The user can now inspect the provided evidence structure (in this case, given as a JSON file), verify that all edges resolve to existing nodes, check temporal consistency, and validate citations.

Stage 5: Independent Verification. The user's skepticism persists and so demands *proof* of logical coherence:

User: How can I verify this BAF is logically coherent? Generate verification code that I can run independently.

AI: I'll create a Python script that verifies structural integrity:

```
import json
import networkx as nx

# Load BAF
with open('acheson_v_laufer_global_baf.json') as f:
    baf = json.load(f)

# Extract argument IDs
arg_ids = {arg['id'] for arg in baf['arguments']}

# Verify all attack edges resolve
for attack in baf['attacks']:
    assert attack['source'] in arg_ids
    assert attack['target'] in arg_ids
print(f"✓ All {len(baf['attacks'])} attacks resolve")

# Verify all support edges resolve
for support in baf['supports']:
    assert support['source'] in arg_ids
    assert support['target'] in arg_ids
print(f"✓ All {len(baf['supports'])} supports resolve")

# Check temporal consistency (no anachronisms)
temporal_order = {'P': 1, 'A': 2, 'R': 3, 'Reply': 4}
for attack in baf['attacks']:
    # Extract brief prefixes
    src_brief = attack['source'].split('_')[0]
    tgt_brief = attack['target'].split('_')[0]
    # Attacks can go backward (later attacking earlier)
    # but should not violate filing order
    print(f"✓ Temporal consistency verified")
```

Running this code confirms:

- Reference integrity: All 92 edges resolve to existing arguments
- No orphan nodes: Every referenced argument exists
- Temporal consistency: No impossible argument orderings

AI: The structure is logically coherent, though legal accuracy requires expert validation.

The program can then be analyzed and executed by the user to confirm the reported basic structural

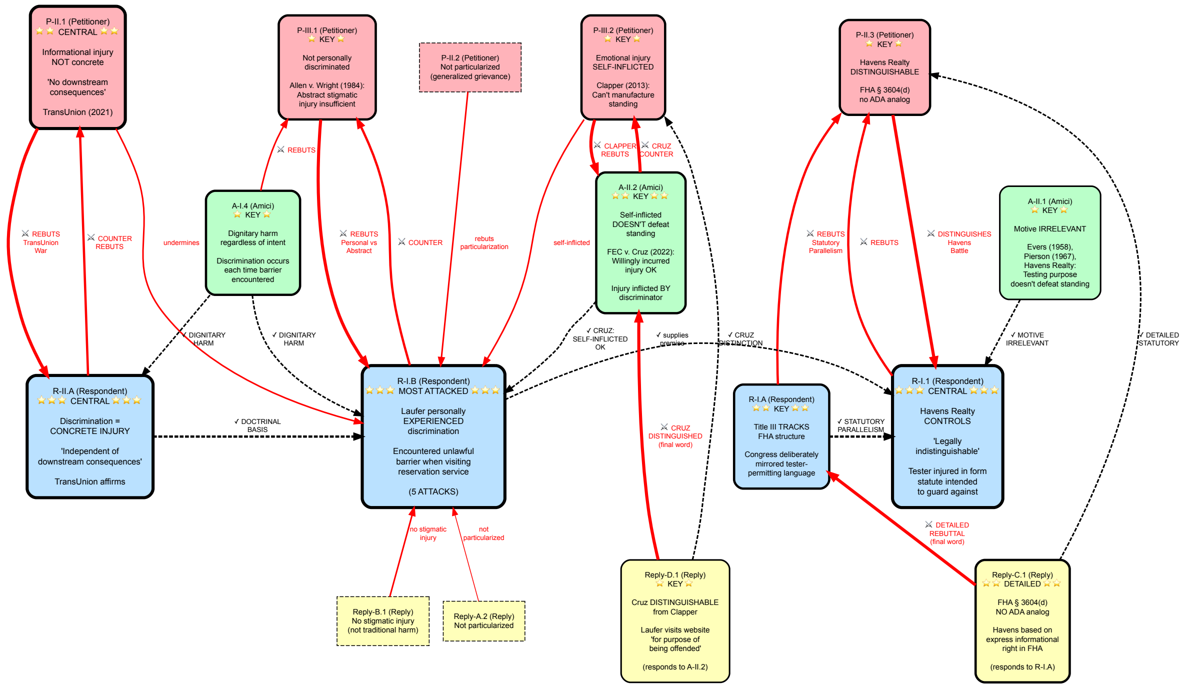

Figure 3: Core Argumentation Structure of Acheson v. Laufer. Visualization showing key arguments across the four briefs (red=Petitioner, blue=Respondent, green=Amici, yellow=Reply). Red solid edges represent *attacks* (dialectical opposition), black dashed edges represent *support*. The graph reveals several "battlegrounds": (1) *TransUnion* interpretation (P-II.1 ↔ R-II.A), (2) *Havens Realty* scope (P-II.3 ↔ R-I.1), (3) personal discrimination (P-III.1 ↔ R-I.B), and (4) self-inflicted injury (P-III.2 ↔ A-II.2). Argument R-I.B ("*Laufer personally experienced discrimination*") is most heavily attacked but also well-supported.

integrity of the argumentation framework. We invite the curious reader to review the AI's explanation, the comments in the Python code, and the actual code to see whether or not the "structural integrity check" can be trusted.

Stage 6: Semantic Analysis. The user then explores argumentation-theoretic properties:

User: Which arguments survive all attacks? Which are most heavily contested?

AI: Computing argumentation semantics reveals:
Most Attacked Argument: R-I.B ("Laufer personally experienced discrimination") receives 5 attacks from P-III.1 (not personally discriminated per *Allen v. Wright*), P-II.1 (undermines concreteness), P-II.2 (not particularized), P-III.2 (self-inflicted injury), and Reply-B.1 (no stigmatic injury). However, R-I.B has strong support from R-II.A (doctrinal basis), A-I.4 (dignitary harm theory), and A-II.2 (*Cruz* allows self-inflicted injury).
Key Battlegrounds: Four main dialectical conflicts emerge:

1. *TransUnion* interpretation: P-II.1 vs. R-II.A (complete interpretive deadlock)
2. *Havens Realty* scope: P-II.3 vs. R-I.1 (statutory structure dispute)
3. Personal discrimination: P-III.1 vs. R-I.B (abstract vs. concrete harm)
4. Self-inflicted injury: P-III.2 vs. A-II.2 (*Clapper* vs. *Cruz*)

Position Balance: Nearly evenly split—19 arguments support "lacks standing" vs. 21 support "has standing." This suggests a genuinely close case.

Finally, Figure 3 shows a visualization of the core argumentation structure produced by the LLM, highlighting these key argumentation "battlegrounds", with citations, that can be verified by the user.

Table 1
Evidence progression across formal and defeasible reasoning domains. Each step strengthens evidential support (from AI-generated assertions to verifiable evidence structures). While the specific evidence and verification approaches differ, each follows a common progression from unverified opinion to externally audited artifacts.

	Primality: Is 74,397,531 prime?		*Legal: Does Laufer have standing?*	
Step	*Evidence Type*	*Verifiability*	*Evidence Type*	*Verifiability*
1	Bare assertion	None (trust in AI)	Bare legal opinion	None (trust in AI)
2	Algorithmic (Python)	Execute code	Timeline metadata	Check against docket
3	Pratt certificate	Arithmetic verification	Extracted arguments	Check source briefs
4	Lean project	Machine-checked	Formal BAF	Machine-checked
5	Proof checking	Deterministic accept/reject	Semantic analysis	Coherence, defense

3.3. Progression of Evidence Across Formal and Defeasible Domains

Table 1 summarizes the evidence progression of the primality testing and legal reasoning examples, highlighting that they both share a similar pattern despite differences in domain, evidential standards, and verification approaches. In each example, the interaction begins with a bare assertion produced by the AI system that, while potentially informative, offers no independently verifiable support. Subsequent steps progressively produce explicit artifacts that can be inspected, checked, or analyzed without trusting the AI's internal reasoning. In the primality case, this progression moves from executable but uncertified code, to a mathematical witness in the form of a Pratt certificate, and ultimately to a fully formal proof artifact verified by the trusted Lean proof kernel. The progression in the legal case follows an analogous trajectory: from checkable metadata, to explicitly extracted arguments grounded in source texts, and finally to a fully externalized BAF accompanied by verification code that enables mechanical checks of structural integrity. The final step in each domain applies an appropriate verification procedure to the formalized artifact: proof checking for primality checking, argumentation-theoretic semantic analysis for the legal case.

This comparison illustrates the core properties of the TAIR framework. First, in both domains, trust is shifted from the AI system to independently verifiable evidence structures, operationalizing the principle of independent verifiability without requiring access to model internals. Second, both examples exhibit a clear computational asymmetry: generating witnesses, proofs, or argumentation frameworks is usually substantially more demanding than checking their validity or coherence using external tools. Third, the staged progression reflects TAIR's iterative refinement principle, in which skepticism motivates increasingly explicit and auditable forms of evidence. Finally, the table makes explicit TAIR's commitment to domain-dependent evidential standards. Formal mathematical reasoning admits a definitive verification endpoint, whereas legal reasoning culminates in structured assessments of coherence and contestability rather than a proof. The next section describes how the TAIR framework accommodates these differences using an evidence-first workflow that supports both formal and defeasible reasoning while preserving the appropriate evidence guarantees of each domain.

4. The TAIR Framework

This section provides an overview of the TAIR meta-workflow, which abstracts the common patterns observed in the examples of Sections 2 and 3 by introducing a set of interacting LLM-based *agents* for identifying and analyzing relevant evidence structures. We first describe the components and control flow of the meta-workflow, and then show how this architecture can be instantiated to support the example formal and defeasible reasoning domains described above.

4.1. Overview of the TAIR Meta-Workflow

Figure 4 provides an overview of the proposed TAIR framework shown as a (meta) workflow. The framework consists of various interacting LLM agents, which are inspired in part by recent approaches

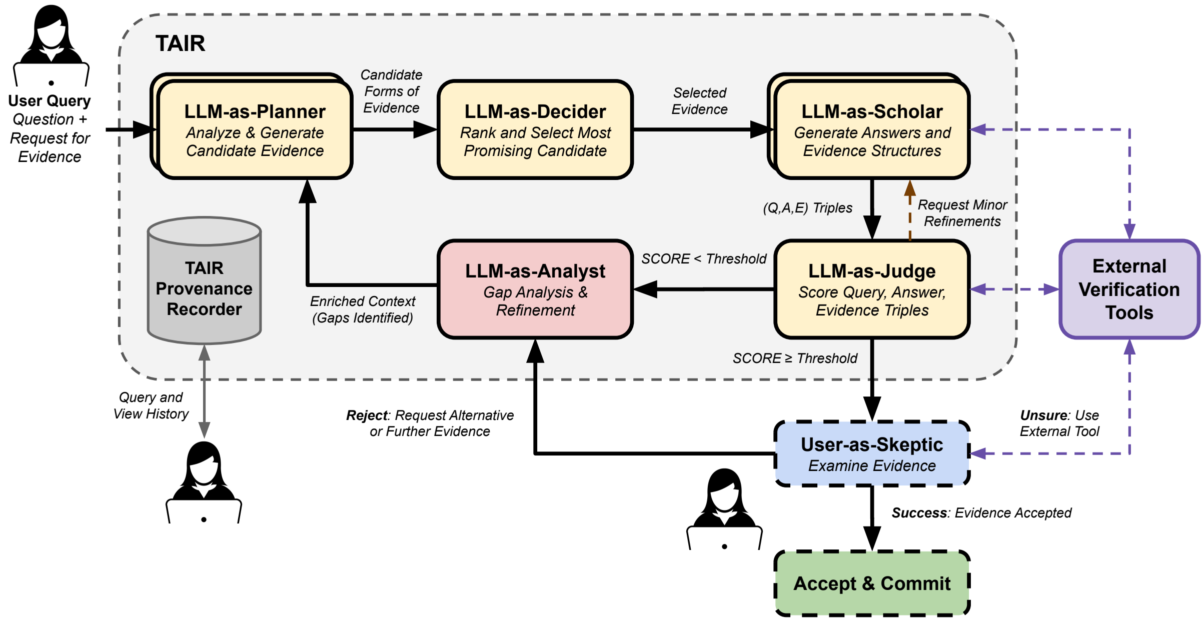

Figure 4: TAIR Question-Answering Meta-Workflow. This multi-agent system involves LLM-as-Planner, Decider, Scholar, Judge, and Analyst roles to generate, evaluate, and refine answers and supporting evidence. The process is iterated until a sufficient score is reached and a user accepts the evidence, with provisions for external verification and gap analysis. Internal steps are recorded and made available for further user analysis.

using LLMs for self-verification [21], peer review [22], reranking [23], and evaluating previous LLM outputs [24, 25]. TAIR also consists of a *provenance recorder* (and corresponding database) for logging workflow runs and storing the artifacts generated by the workflow. TAIR provenance information can be accessed by internal TAIR components as well as users interested in viewing the steps involved in generating a result and viewing the relevant TAIR artifacts. As shown in Figure 4, the workflow begins with a user question Q that is initially processed by the planner agent:

LLM-as-Planner: The *planner agent* takes the initial user question Q and generates one or more relevant forms of evidence. Each form of evidence ideally includes a basic sketch of the evidence structure for use by downstream agents to further consider and elaborate. Multiple planners (e.g., different instruct models, different continuations, etc.) can also be employed to generate a larger number of alternative, and potentially more diverse, forms of evidence (shown as a double box in Figure 4). Each potential form of evidence is then passed to the decider (i.e., ranker) agent.

LLM-as-Decider: The *decider agent* ranks each output from the planner phase and selects the most promising candidate. The selected form of evidence is then sent to the scholar agent. We note that similar to the reranking approach in [23], the decider agent could also be extended to use prior decisions and results to improve (and further customize) its rankings.

LLM-as-Scholar: The *scholar agent* takes the result of the decider agent and generates one or more answers A and evidence structures E to the user question Q, forming triples (Q, A, E). As part of its work, the scholar may choose to employ external verification tools (e.g., by writing and executing Python or Lean code). Similar to the planner agent, multiple LLMs may be employed to generate a diverse set of (Q, A, E) triples. The triples generated by the scholar agent are sent to the judge agent.

LLM-as-Judge: The *judge agent* evaluates and scores each (Q, A, E) triple it receives from the scholar agent. As part of this process, the judge agent may request a small change from the scholar agent for a given evidence structure, e.g., if it is clear that the answer and/or evidence structure is incomplete. Scores are normalized and compared to a TAIR threshold parameter. The triple with the highest score at or above the threshold value is output to the user for further analysis, pausing the workflow. If no triple produced by the scholar agent meets the threshold value, the judge rejects the results and requests alternative evidence from the analyst agent. This overall process (from planner to judge) may repeat for a certain maximum number of rounds as defined by TAIR configuration settings. The process may also be halted if scores do not advance towards the threshold value. Similar to the scholar agent, the judge agent may use external verification tools for evaluating and scoring results.

LLM-as-Analyst: The *analyst agent* functions as a domain-adaptive refinement engine. In formal domains (as in Section 2), it performs *error analysis*: if the judge (e.g., a Lean compiler or Python interpreter) reports a failure, the analyst interprets the specific error message (e.g., "*tactic failed*" or "*variable undefined*") to guide the planner in repairing the proof script or witness code. In *defeasible domains* (as in Section 3), it functions as a *dialectical opponent* using, e.g., Walton's *argumentation schemes* [26]. When the judge rejects an evidence structure, the analyst identifies the specific reasoning pattern used (e.g., *argument from precedent*) and retrieves the standard *critical questions (CQs)* associated with that scheme. A "gap" in this case is defined as a critical question that the current evidence fails to address (e.g., "*Is the new case relevantly similar to the cited precedent?*"). The analyst then formulates a new prompt targeting these unanswered CQs, effectively attempting to undercut or rebut the potential defeaters detected in the previous iteration.

User-as-Skeptic: The (positive) output of the judge agent is provided to the user (playing the role of a skeptic), who can then critically examine the provided answer and evidence structure, i.e., the output (Q, A, E) triple. If the user accepts the given evidence, then the process completes. Alternatively, if the user rejects the evidence, they can request either alternative or further evidence for their question via the analyst agent (which then restarts the workflow as described above). Before accepting or rejecting evidence, the user may also decide to further check the result using external verification tools (e.g., by running Python or Lean over the code in the evidence structure). The user may then accept the evidence or reject it and restart the workflow by asking the analyst for alternative or further evidence.

4.2. TAIR Meta-Workflow Instantiation Across Domains

Table 2 summarizes how the TAIR meta-workflow can be instantiated to support the two examples presented earlier, despite their differences in domain and evidential standards. As shown in Table 2, the planner agent first identifies candidate forms of evidence appropriate to the query, e.g., trial division or Pratt certificates in the primality case or as a precendent analysis or BAF in the legal case. The result of the decider agent is then supplied to the scholar agent, which constructs concrete evidence artifacts, e.g., producing either formal proof objects or structured argumentation frameworks. These artifacts are evaluated by the judge agent using domain-appropriate verification procedures, such as proof checking in a theorem prover or structural and semantic analysis of an argumentation framework. When verification fails or gaps are identified, the analyst agent drives refinement, either by repairing formal proofs or by identifying missing or contested arguments. Across both domains, the workflow produces explicit (Q, A, E) triples in which answers are paired with inspectable evidence structures rather than informal explanations.

Table 2
Instantiation of the TAIR meta-workflow across formal and defeasible reasoning domains, where TAIR components are mapped to their roles in the primality-testing and legal-reasoning case studies.

TAIR Component	*Primality: Is 474,397,531 prime?*	*Legal: Does Laufer have standing?*
Planner	Pratt certificate (vs factors, Miller-Rabin)	BAF (vs precedent analysis, warrants)
Scholar	Generate witness + Lean proof	Extract arguments, supports, and attacks
Judge	Lean compiler (formal verification)	Structural and semantic checks
Analyst	Repair proof errors	Critical questions (Walton schemes [26])
Evidence (E)	Pratt certificate and verified Lean proof	BAF with source citations
Verification	Formal proof checking	Extension analysis, expert evaluation

This instantiation demonstrates how the meta-workflow operationalizes TAIRs core principles. Independent verifiability is achieved by ensuring that final judgments are rendered by external tools or expert evaluation rather than by the AI model itself. Computational asymmetry is preserved in both domains: constructing proofs or argumentation frameworks (from complex legal cases and corresponding documents) can be substantially more difficult than checking their coherence or validity. Iterative refinement is realized through explicit feedback loops in which failed verification or unresolved challenges are further investigated through gap analysis (ideally leading to improved evidence structures). Finally, the workflow enforces domain-dependent evidential standards by allowing different verification approaches, ranging from formal proof checking to defeasible semantic analysis. By treating evidence structures as first-class artifacts throughout the workflow, TAIR provides a unified, evidence-first approach for enabling user trust by supporting independent verification of AI results.

4.3. TAIR and Agentic Workflows

Recent agentic AI systems for programming and problem solving have independently adopted iterative *generate-test-repair* loops that align closely with TAIR's architectural principles. Systems such as Claude Code, OpenAI Codex, and Gemini CLI achieve reliability not through faithful introspective explanations of model reasoning, but by externalizing trust to independent verification mechanisms. These systems propose solutions that are validated by external tools such as compilers, test suites, and linters, creating feedback loops that enable iterative refinement. This emerging practice in agentic coding systems both validates TAIR's design principles and illustrates opportunities for extending these reliability patterns beyond programming domains.

Table 3
Mapping TAIR components to agentic coding system roles.

TAIR Component	Agentic Coding Systems	Key Mechanism
Planner	Task decomposition, strategy selection	LLM proposes approach
Decider	Implicit heuristic pruning	Often weak or absent
Scholar	Code generation, test synthesis	LLM as conjecture engine
Judge	Compiler, test runner, type checker	*External hard oracle*
Analyst	Error interpretation, constraint-guided repair	Diagnostic feedback loop
Evidence (E)	Code + passing tests	Trust anchor
User-as-Skeptic	Developer approval	Final acceptance authority

Table 3 shows a possible mapping between TAIR's conceptual components and their counterparts in agentic coding systems. In both architectures, neural components generate candidate solutions while symbolic components enforce acceptance criteria. The *planner* role manifests in task decomposition and solution strategy selection, though often implicitly. The *scholar* role corresponds directly to code synthesis and test generation. Most critically, the *judge* role is realized through external verification tools (compilers, type checkers, and test runners) that provide deterministic accept/reject signals independent of the model. The *analyst* role interprets failure signals (compiler errors, failed tests) to guide constraint-driven refinement, while the (skeptical) *user* retains ultimate authority over whether to accept the proposed solutions.

This architectural convergence reveals that agentic coding systems work well when they possess strong verification oracles. Reliability emerges because errors are made *observable*, acceptance conditions are externalized, and iteration is gated by hard constraints (rather than model confidence). These systems may also obey the principle of computational asymmetry, i.e., checking whether code compiles and passes tests is often easier than generating correct implementations.

Current agentic coding systems are domain-specific and engineering-driven, lacking explicit theoretical grounding. TAIR extends this emerging pattern: First, it abstracts the common architectural principles (generation, verification, gap detection) into a domain-agnostic framework applicable beyond programming. Second, it generalizes the notion of verification oracles from compilers and test suites to more diverse evidence structures: proof assistants for mathematical reasoning, argumentation semantics for legal analysis, and warrant chains for scientific discourse. This generalization addresses a fundamental limitation of agentic coding workflows: these achieve reliability primarily in domains where strong mechanical verifiers exist (code, formal mathematics), but struggle in domains where verification requires human expertise or defeasible reasoning. TAIR provides a framework for extending agentic reliability patterns into such domains by making evidence structures and verification roles explicit across both formal and defeasible reasoning settings.

TAIR and Neurosymbolic AI Approaches. TAIR and its agentic variations can be viewed as a specialized class of neurosymbolic AI [27] that operationalizes the integration of "System 1" generative intuition with "System 2" analytical reasoning. In this architecture, the LLM functions as a high-capacity heuristic engine (System 1), rapidly proposing candidate solutions, while the external evidence verifier enforces strict logical or structural constraints (System 2). This synergy shows, e.g., in neurosymbolic theorem proving, where the combination of LLMs for proof synthesis and interactive theorem provers for verification has become a standard paradigm. Systems like AlphaProof [28], which achieved silver and gold medal performance at the *International Mathematical Olympiad*, exemplify this approach. The rapid proliferation of Lean 4-focused benchmarks (including MiniF2F [29], ProofNet [30], and PutnamBench) and toolchains validates this architecture. These systems accept the LLM's "jagged frontier" [10] of unreliable generation because they rely entirely on the trusted kernel of the proof assistant for validity. Beyond formal mathematics, frameworks like LLM-ASPIC+ [27] demonstrate analogous patterns in defeasible reasoning, combining neural triggering with symbolic argumentation semantics. TAIR generalizes this neurosymbolic pattern, proposing that the reliability achieved in formal mathematics via proof assistants can be replicated in other high-stakes domains by identifying analogous evidence structures and verification kernels.

5. Discussion and Future Directions

Proof-carrying code and certifying algorithms. TAIR's certificate paradigm builds on two parallel traditions that formalize the principle of separating untrusted generation from trusted verification, i.e., *proof-carrying code* (PCC) [31] and *certifying algorithms* [32]. Though the underlying practices have been known for a long time[6], these frameworks were the first to articulate verification as an explicit design principle for trustworthy computation. PCC addresses code safety by bundling executables with machine-checkable safety proofs. Rather than trusting code producers, PCC relies on a small trusted *proof checker* to verify certificates independently, enabling safe execution of code from arbitrary sources. Certifying algorithms address computational correctness by producing not just solutions but *witnesses* enabling efficient verification. This approach is well-known, e.g., in SAT/SMT solving [33], where modern solvers emit proof traces or UNSAT certificates, and in interactive theorem proving [16, 34], where formal proofs serve as certificates of mathematical truth.

Certifiable machine learning. TAIR generalizes the above ideas to *epistemic trust* for AI systems: LLMs are treated as *untrusted* generators that must produce *verifiable evidence* with their answers. Where PCC produces program-certificate pairs (P, C) verified by proof checkers, TAIR produces triples (Q, A, E) verified by domain-appropriate tools (e.g., proof assistants, argumentation analyzers, or expert assessment). Both paradigms exploit computational asymmetry: verification should be substantially cheaper than generation. Barrett *et al.* [14], based on similar ideas in program synthesis, define the principle of *certifiable machine learning* (CML) for improving the quality of AI-generated artifacts. Instead of asking an AI to learn or generate the desired artifact, CML asks for an answer that contains both the desired result and evidence that the result meets the intended specification (expected properties). They propose a simple *learner-verifier* loop: the learner generates an artifact and a certificate (evidence that can be evaluated by a third party) that the verifier (e.g., a proof system) then checks. If the verifier finds the certificate does not satisfy the desired properties, it attempts to find a counterexample, which is returned to the learner. Unlike in modern program synthesis techniques, CML advocates for the

[6] e.g, witness-based primality testing [15] and the extended (i.e., certifying) Euclidean algorithm already used this principle

learner to perform the majority of certificate generation, reducing the work of the verifier. TAIR is both a generalization and an elaboration of the CML learner-verifier feedback architecture.

Warrant chains. While this paper has focused on formal verification (e.g., Pratt certificates and Lean proofs) and structured defeasible reasoning (e.g., bipolar argumentation frameworks), TAIR also accommodates *warrant chains* as informal evidence structures. The concept originates with Toulmin [35], whose argumentation model distinguished *data* (facts), *claims* (conclusions), and *warrants* (inference licenses connecting data to claims). Warrant chains are informal reasoning bridges validated by human experts, connecting evidence to conclusions through explicit inference steps. Examples include source citations with explicit reasoning ("*according to S, X holds because Y*"), analogical reasoning chains, and expert testimony synthesis. Prakken [36] formalized these concepts, showing how warrants can be made explicit in structured argumentation. Warrant chains appear naturally throughout academic discourse: systematic reviews synthesize studies into meta-analytic conclusions, each earlier study serving as a warrant; theoretical developments build chains where seminal results become warrants for subsequent work; and legal precedents form warrant chains through *stare decisis*[7]. Recent work has begun automating warrant chain verification: Wu et al. [37] developed techniques for checking whether cited medical sources actually support attributed claims, essentially verifying warrant adequacy in LLM-generated medical reasoning. This represents a first step toward systematic validation of warrant chains in domains where formal verification is unavailable but structured justification remains essential.

Argumentation frameworks as explanations. Argumentation frameworks have also been used to structure and evaluate LLM-generated natural language arguments elicited through prompting, which are post-processed to derive probabilistic BAFs for further analysis (e.g., [38]). This line of work is complementary to TAIR but operates at a different level of abstraction: rather than deriving BAFs from an LLM's textual output, TAIR uses LLMs to produce explicit evidence structures (including BAFs) for external verification and iterative refinement.

Future work. We plan to implement and evaluate the TAIR framework within the context of legal reasoning, focusing on similar examples as in Section 3. We envision TAIR as a major component within the broader XAI-CA (eXplainable-AI via Computational Argumentation) project [39], which aims to leverage neurosymbolic approaches (i.e., combining LLMs and formal argumentation, e.g., in law [40]) and explainability methods to support analysis and decision making within case law for the broader legal community.

Acknowledgments

This work emerged from XAI-CA [39], a collaboration between Cardiff University and the University of Illinois, Urbana-Champaign. We thank Haoyang Liu for comments on an earlier version of the paper.

Declaration on Generative AI

During preparation of this work, the authors used ChatGPT, Claude, and NotebookLM to: draft specific content, paraphrase, and reword portions of the text. After each tools use, the authors reviewed, edited, and revised all content as needed and take full responsibility for the publication's content.

References

[1] M. Silvestri, V. Vineis, E. Gabrielli, F. Giorgi, F. Veglianti, F. Silvestri, G. Tolomei, A Survey on Explainable AI Narratives based on Large Language Models (2025). URL: https://www.techrxiv.org/doi/full/10.36227/techrxiv.176282217.77757258.

[7] Latin: "to stand by things decided", i.e., the legal principle that courts should follow precedents set by previous decisions.

[2] A. B. Arrieta, N. Díaz-Rodríguez, J. D. Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, F. Herrera, Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI, Information Fusion 58 (2020) 82–115. doi:10.1016/j.inffus.2019.12.012.

[3] Y. Chen, J. Benton, A. Radhakrishnan, J. Uesato, C. Denison, J. Schulman, A. Somani, P. Hase, M. Wagner, F. Roger, V. Mikulik, S. R. Bowman, J. Leike, J. Kaplan, E. Perez, Reasoning Models Don't Always Say What They Think, 2025. URL: http://arxiv.org/abs/2505.05410.

[4] I. Arcuschin, J. Janiak, R. Krzyzanowski, S. Rajamanoharan, N. Nanda, A. Conmy, Chain-of-Thought Reasoning In The Wild Is Not Always Faithful, 2025. URL: http://arxiv.org/abs/2503.08679.

[5] F. Barez, T.-Y. Wu, I. Arcuschin, M. Lan, V. Wang, N. Siegel, N. Collignon, Chain-of-Thought Is Not Explainability (2025). URL: https://fbarez.github.io/assets/pdf/Cot_Is_Not_Explainability.pdf.

[6] H. Mayne, R. O. Kearns, Y. Yang, A. M. Bean, E. D. Delaney, C. Russell, A. Mahdi, LLMs Don't Know Their Own Decision Boundaries: The Unreliability of Self-Generated Counterfactual Explanations, in: Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2025, pp. 24161–24186. URL: https://aclanthology.org/2025.emnlp-main.1231/.

[7] W. Walden, Reasoning Models Will Blatantly Lie About Their Reasoning, 2026. URL: http://arxiv.org/abs/2601.07663.

[8] C. Rudin, Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead, Nature Machine Intelligence 1 (2019) 206–215. doi:10.1038/s42256-019-0048-x.

[9] A. Alansari, H. Luqman, A Comprehensive Survey of Hallucination in Large Language Models: Causes, Detection, and Mitigation, 2025. URL: http://arxiv.org/abs/2510.06265.

[10] F. Dell'Acqua, E. McFowland, E. R. Mollick, H. Lifshitz-Assaf, K. C. Kellogg, S. Rajendran, L. Krayer, F. Candelon, K. R. Lakhani, Navigating the Jagged Technological Frontier: Field Experimental Evidence of the Effects of AI on Knowledge Worker Productivity and Quality, Working Paper 24-013, Harvard Business School, 2023. URL: https://www.hbs.edu/faculty/Pages/item.aspx?num=64700.

[11] A. Karpathy, Jagged intelligence, 2024. URL: https://x.com/karpathy/status/1816531576228053133, social Media Post.

[12] H. Reichenbach, Experience and Prediction: An Analysis of the Foundations and the Structure of Knowledge, University of Chicago Press, Chicago, 1938.

[13] P. Hoyningen-Huene, Context of discovery and context of justification, Studies in History and Philosophy of Science Part A 18 (1987) 501–515. doi:10.1016/0039-3681(87)90005-7.

[14] C. Barrett, T. A. Henzinger, S. A. Seshia, Certificates in AI: Learn but Verify, Communications of the ACM 69 (2026) 66–75. URL: https://dl.acm.org/doi/10.1145/3737447. doi:10.1145/3737447.

[15] V. R. Pratt, Every Prime Has a Succinct Certificate, SIAM J. Comput. 4 (1975) 214–220.

[16] L. de Moura, S. Ullrich, The lean 4 theorem prover and programming language, in: A. Platzer, G. Sutcliffe (Eds.), Automated Deduction (CADE), volume 12699 of *Lecture Notes in Computer Science*, Springer, 2021, pp. 625–635. doi:10.1007/978-3-030-79876-5_37.

[17] P. M. Dung, On the Acceptability of Arguments and Its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games, Artificial Intelligence 77 (1995) 321–357.

[18] P. Baroni, D. Gabbay, M. Giacomin (Eds.), Handbook of Formal Argumentation, volume 1, College Publications, London, 2018.

[19] A. Borg, D. Odekerken, PyArg for solving and explaining argumentation in python: Demonstration, in: F. Toni, S. Polberg, R. Booth, M. Caminada, H. Kido (Eds.), Computational Models of Argument (COMMA), volume 353 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, Amsterdam, 2022. doi:10.3233/FAIA220167.

[20] Y. Xia, H. Zheng, S. Bowers, B. Ludäscher, AF-XRAY: Visual explanation and resolution of ambiguity in legal argumentation frameworks, in: 20th Intl. Conf. on AI and Law (ICAIL), ACM, 2025.

[21] Y. Weng, M. Zhu, F. Xia, B. Li, S. He, S. Liu, B. Sun, K. Liu, J. Zhao, Large Language Models are Better Reasoners with Self-Verification, in: Findings of the Association for Computational Linguistics: EMNLP 2023, Association for Computational Linguistics, 2023, pp. 2550–2575.

[22] Z. Chu, Q. Ai, Y. Tu, H. Li, Y. Liu, Automatic large language model evaluation via peer review, in: ACM International Conference on Information and Knowledge Management (CIKM), 2024, p. 384–393. URL: https://doi.org/10.1145/3627673.3679677.

[23] Z. Tan, J. Yan, I.-H. Hsu, R. Han, Z. Wang, L. Le, Y. Song, Y. Chen, H. Palangi, G. Lee, A. R. Iyer, T. Chen, H. Liu, C.-Y. Lee, T. Pfister, In Prospect and Retrospect: Reflective Memory Management for Long-term Personalized Dialogue Agents, in: W. Che, J. Nabende, E. Shutova, M. T. Pilehvar (Eds.), Annual Meeting of the Association for Computational Linguistics, 2025, pp. 8416–8439.

[24] Z. Sollenberger, J. Patel, C. Munley, A. Jarmusch, S. Chandrasekaran, LLM4VV: Exploring LLM-as-a-Judge for Validation and Verification Testsuites, in: Workshops of the International Conference on High Performance Computing, Network, Storage, and Analysis, 2025, p. 1885–1893.

[25] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. P. Xing, H. Zhang, J. E. Gonzalez, I. Stoica, Judging llm-as-a-judge with mt-bench and chatbot arena, in: International Conference on Neural Information Processing Systems, 2023.

[26] D. Walton, C. Reed, F. Macagno, Argumentation Schemes, Cambridge University Press, 2008.

[27] X. Fang, Z. Li, C. Chen, B. Liao, LLM-ASPIC+: A neuro-symbolic framework for defeasible reasoning, in: Proceedings of the 27th European Conference on Artificial Intelligence (ECAI), IOS Press, 2025, pp. 1567–1574. doi:10.3233/FAIA250981.

[28] T. Hubert, R. Mehta, L. Sartran, T. Zahavy, L. Schrittwieser, R. Munos, M. Rowland, M. Balog, A. Huang, D. Hassabis, P. Kohli, D. Silver, Olympiad-level formal mathematical reasoning with reinforcement learning, Nature (2025). doi:10.1038/s41586-025-09833-y.

[29] K. Zheng, J. M. Han, S. Polu, MiniF2F: A cross-system benchmark for formal Olympiad-level mathematics, in: International Conference on Learning Representations (ICLR), 2022. URL: https://openreview.net/forum?id=9ZPegFuFTFv.

[30] Z. Azerbayev, B. Piotrowski, H. Schoelkopf, E. W. Ayers, D. Radev, J. Avigad, ProofNet: Autoformalizing and formally proving undergraduate-level mathematics, in: 37th Conference on Neural Information Processing Systems (NeurIPS) Track on Datasets and Benchmarks, 2023.

[31] G. C. Necula, Proof-carrying code, in: Proceedings of the 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 1997), ACM, 1997, pp. 106–119.

[32] R. M. McConnell, K. Mehlhorn, S. Näher, P. Schweitzer, Certifying algorithms, Computer Science Review 5 (2011) 119–161. doi:10.1016/j.cosrev.2010.10.001.

[33] L. Zhang, S. Malik, Validating SAT solvers using an independent resolution-based checker: Practical implementations and other applications, in: Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE), 2003, pp. 880–885.

[34] T. Nipkow, L. C. Paulson, M. Wenzel, Isabelle/HOL — A Proof Assistant for Higher-Order Logic, volume 2283 of *Lecture Notes in Computer Science*, Springer, 2002.

[35] S. E. Toulmin, The Uses of Argument, Cambridge University Press, Cambridge, 1958.

[36] H. Prakken, An abstract framework for argumentation with structured arguments, Argument & Computation 1 (2010) 93–124. doi:10.1080/19462160903564592.

[37] K. Wu, E. Wu, K. Wei, A. Zhang, A. Casasola, T. Nguyen, S. Riantawan, P. Shi, D. Ho, J. Zou, An automated framework for assessing how well LLMs cite relevant medical references, Nature Communications 16 (2025) 3615. URL: https://www.nature.com/articles/s41467-025-58551-6.

[38] G. Freedman, A. Dejl, D. Gorur, X. Yin, A. Rago, F. Toni, Argumentative Large Language Models for Explainable and Contestable Claim Verification, Proceedings of the AAAI Conference on Artificial Intelligence (2025) 14930–14939.

[39] M. Caminada, B. Ludäscher, H. Zheng, XAI-CA: Explainable AI via Computational Argumentation, Joint Seed Grant, Cardiff University and University of Illinois at Urbana-Champaign, 2024.

[40] H. Prakken, G. Sartor, A formal framework for combining legal reasoning methods, in: Proceedings of the Nineteenth International Conference on Artificial Intelligence and Law (ICAIL 2023), ACM, Braga, Portugal, 2023, pp. 168–177.