ARGUS: Argumentation-Based Minimal-Change Repair for Verifiable LLM Self-Explanations

Paper ID: XXX

Anonymous Authors
anonymous@example.com

Abstract

Large language models produce natural-language rationales for their outputs, yet these explanations are frequently unfaithful to the model's internal reasoning and cannot be updated when new evidence emerges. We introduce ARGUS, a framework that structures LLM self-explanations as Dung-style abstract argumentation frameworks, verifies them under grounded and preferred semantics, and—when an evidence update renders the explanation inconsistent—computes a minimum-cost sequence of edit operations that restores the desired acceptability status of the target argument. The repair operator satisfies adapted AGM revision postulates (success, inclusion, vacuity) and admits a complexity-theoretic characterization: the decision problem is in P under grounded semantics and NP-complete under preferred semantics. A k-neighborhood approximation and answer set programming encoding ensure scalability to practical framework sizes. We validate the framework on HotpotQA and FEVER, where ARGUS improves faithfulness by up to 10.3% and contestability by up to 14.5% over the strongest argumentation baseline while requiring fewer repair operations than all competing methods.

1 Introduction

Large language models generate natural-language explanations for their outputs, yet mounting evidence indicates that these self-explanations are frequently unfaithful to the model's internal reasoning process. Recent studies demonstrate that LLM rationales can be inconsistent with the computations that actually produce the answer (Ye and Durrett 2024), and that chain-of-thought traces are often post-hoc rationalizations rather than faithful accounts of inference (Lanham et al. 2023). The gap between *apparent* and *actual* reasoning makes the verification and maintenance of explanations a central knowledge representation challenge, particularly in domains such as medical diagnosis and legal reasoning where explanation correctness is critical.

Current approaches to improving LLM explanations fall short along two complementary dimensions. Self-correction methods such as Self-Refine (Madaan et al. 2023) and Reflexion (Shinn et al. 2023) iteratively rewrite explanations using the model's own feedback, but they operate without formal guarantees: edits are unconstrained, previously valid reasoning steps may be silently discarded, and there is no mechanism to ensure that changes are minimal or semantically justified. On the other end of the spectrum, argumentation-based approaches such as ArgLLMs (Freedman et al. 2025) and ARGORA (Anonymous 2026) structure explanations into argument graphs and verify them against formal semantics, providing rigorous one-shot verdicts. However, these frameworks treat verification as a static, terminal operation. When new evidence arrives—a factual correction, a credible counterargument, or an updated knowledge source—they offer no principled way to update the explanation, forcing the system to either regenerate from scratch or apply ad-hoc edits that may violate the very consistency guarantees the formalism was designed to provide. To the best of our knowledge, no existing framework provides a formal notion of *minimal change* for maintaining LLM explanations under evolving evidence.

The following example, revisited throughout the paper, illustrates the problem concretely.

Example 1 (Medical Diagnosis)**.** *A question-answering system is asked to diagnose a patient with fatigue and joint pain. The LLM answers "Lupus" with four argument units: a_1 ("chronic fatigue reported"), a_2 ("polyarthralgia present"), a_3 ("Lupus commonly presents with these symptoms"), and target a_4 ("most likely diagnosis is Lupus"). A new lab result a_5 ("ANA test is negative") attacks a_3, rendering a_4 no longer accepted under grounded semantics. An unconstrained self-correction system might regenerate the entire explanation, discarding the valid units a_1 and a_2. A minimal-change repair instead seeks the smallest edit—such as introducing a_6 ("anti-dsDNA positive") attacking a_5—to restore a_4 at cost 2 (visualized in Figure 1).*

We propose ARGUS, a framework that bridges this gap by unifying argumentation-based verification with minimal-change repair. Given an LLM-generated explanation, ARGUS decomposes it into atomic argument units, constructs an argumentation framework in the sense of Dung (Dung 1995), and verifies whether the target claim is accepted under a chosen semantics. When new evidence renders the explanation inconsistent, ARGUS computes a minimum-cost sequence of edit operations—adding or removing arguments and attacks—that restores the desired acceptability status. The repair operator draws on two classical KR traditions: the AGM theory of belief revision (Alchourrón, Gärdenfors, and Makinson 1985), which supplies the minimal-change principle, and argumentation dynamics (Cayrol, de Saint-Cyr, and

Lagasquie-Schiex 2020), which provides the formal machinery for reasoning about changes to attack structures.

Our contributions are as follows:

1. **(C1)** A framework that structures LLM self-explanations as Dung-style argumentation frameworks and verifies them under grounded and preferred semantics, producing defense-set certificates that make acceptance verdicts interpretable (§4).
2. **(C2)** A minimal-change repair operator for explanation maintenance under evolving evidence, satisfying adapted AGM revision postulates with a complexity analysis placing the problem in P under grounded semantics and NP-complete under preferred semantics (§4.4–§5).
3. **(C3)** A scalable ASP encoding with a k-neighborhood approximation that restricts the search space to arguments near the target, reducing solver grounding substantially while preserving repair quality (§4).
4. **(C4)** An empirical evaluation on HotpotQA and FEVER validating the formal properties and demonstrating improvements in faithfulness, contestability, and repair cost w.r.t. seven baselines (§6).

The remainder of this paper is organized as follows. §2 surveys related work; §3 introduces the formal background on argumentation frameworks and belief revision; §4 presents the ARGUS pipeline; §5 establishes theoretical properties of the repair operator; and §6 reports experimental results. Example 1 is revisited throughout to build intuition.

2 Related Work

Our work connects three lines of research: argumentation-based approaches to LLM reasoning, self-correction methods for language models, and formal theories of belief change in argumentation.

Argumentation and LLMs. Several recent proposals structure LLM outputs using argumentation frameworks. ArgLLMs (Freedman et al. 2025) decomposes LLM-generated claims into Dung-style argument graphs and applies grounded and preferred semantics to determine acceptability, producing explainable and contestable verification verdicts. However, ArgLLMs treats verification as a one-shot, terminal operation: once the argument graph is constructed and evaluated, there is no mechanism to update the explanation when new evidence arrives, forcing the user to regenerate the entire rationale from scratch. ARGORA (Anonymous 2026) orchestrates multiple LLM agents through an argumentation-mediated dialogue, incorporating causal semantics to ground the reasoning process. While ARGORA includes a correction mechanism, it operates through agent re-deliberation rather than through a formally defined repair operator with cost minimization and provable guarantees. MQArgEng (Castagna et al. 2024) investigates whether modular argumentation engines can improve LLM reasoning accuracy, demonstrating gains on structured tasks but without addressing explanation maintenance under evolving evidence. ARGUS differs from all three in providing a minimal-change repair operator with AGM-compliant guarantees and complexity-theoretic characterization, treating explanation maintenance as a first-class optimization problem rather than an afterthought.

Self-Correction and Revision. Self-Refine (Madaan et al. 2023) iteratively rewrites LLM outputs using the model's own feedback, while Reflexion (Shinn et al. 2023) augments this paradigm with episodic memory to guide subsequent attempts. Both methods can improve output quality, but their edits are unconstrained: there is no formal notion of minimality, previously correct reasoning steps may be silently discarded, and the revision process offers no semantic guarantees about which parts of the explanation remain intact. RARR (Gao et al. 2023) retrieves external evidence to revise LLM statements, yet its edits target surface-level attribution without modeling the inferential structure that connects claims. SelfCheckGPT (Manakul, Liusie, and Gales 2023) detects hallucinations through sampling consistency but provides no native repair mechanism; detected inconsistencies must be addressed by regenerating the affected explanation segments. In contrast, ARGUS formalizes the repair search space as edits to an argumentation framework, bounds the cost of change, and guarantees that unaffected reasoning steps are preserved.

Belief Revision and Argumentation Dynamics. The AGM theory (Alchourrón, Gärdenfors, and Makinson 1985) establishes rationality postulates for belief change, and Katsuno and Mendelzon (Katsuno and Mendelzon 1992) distinguish revision from update in propositional settings. In the argumentation literature, Cayrol et al. (Cayrol, de Saint-Cyr, and Lagasquie-Schiex 2020) study how adding or removing arguments affects extensions, and Baumann and Brewka (Baumann and Brewka 2010) analyze the complexity of extension enforcement—ensuring that a designated set becomes an extension through framework modifications. Coste-Marquis et al. (Coste-Marquis et al. 2014) formalize argumentation revision as minimal status change, while Wallner et al. (Wallner, Niskanen, and Järvisalo 2017) provide tight complexity bounds for enforcement under multiple semantics. Bisquert et al. (Bisquert et al. 2013) propose a change model for argumentation frameworks based on minimal operations. Our repair operator instantiates these classical ideas in the specific setting of LLM explanation maintenance, combining the AGM minimal-change principle with argumentation enforcement while introducing a weighted cost model tailored to the confidence and structural role of each argument unit.

3 Preliminaries

3.1 Abstract Argumentation Frameworks

We adopt the foundational model of (Dung 1995) as the backbone of our verification and repair pipeline.

Definition 2 (Abstract Argumentation Framework)**.** *An abstract argumentation framework (AF) is a pair $F = (\mathcal{A}, \mathcal{R})$ where $\mathcal{A}$ is a finite set of arguments and $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{A}$ is a binary attack relation. We write $a \rightsquigarrow b$ whenever $(a, b) \in \mathcal{R}$, meaning a attacks b.*

Example 3 (Continuing Example 1)**.** *The initial AF is $F_0 = (\{a_1, a_2, a_3, a_4\}, \emptyset)$ with no attacks; after the negative ANA*

result, $F_1 = (\{a_1, \ldots, a_5\}, \{(a_5, a_3)\})$, as shown in Figure 1(a–b).

Given an AF $F = (\mathcal{A}, \mathcal{R})$, a set $S \subseteq \mathcal{A}$ is *conflict-free* if no two arguments in S attack each other. An argument a is *defended* by S if every attacker of a is attacked by some member of S. A conflict-free set S is *admissible* if it defends all its elements. The principal semantics we employ are the *grounded* extension, which is the unique minimal complete extension obtained as the least fixed point of the characteristic function; the *preferred* extensions, which are maximal admissible sets; and *stable* extensions, which are conflict-free sets that attack every argument outside themselves (Baroni et al. 2018). Throughout this paper we write $\sigma(F)$ to denote the set of extensions of F under semantics $\sigma \in \{gr, pr, st\}$.

3.2 Argumentation Semantics for Explanation

We now define the key notion linking argumentation semantics to explanation. An argument $a \in \mathcal{A}$ is *credulously accepted* under σ if a belongs to at least one extension in $\sigma(F)$, and *skeptically accepted* if it belongs to every extension.

Definition 4 (Defense Set)**.** *Given an AF $F = (\mathcal{A}, \mathcal{R})$, semantics σ, and a skeptically accepted argument $t \in \mathcal{A}$, a* defense set *for t is a minimal admissible set $D \subseteq \mathcal{A}$ such that $t \in D$. We write $Def_\sigma(t)$ for the collection of all defense sets of t under σ.*

Example 5 (Continuing Example 1)**.** *In F_0 (Figure 1a), $D = \{a_1, a_2, a_3, a_4\}$ is a defense set for a_4: it is admissible and $a_4 \in D$. In F_1, D is no longer admissible because a_3 is attacked by a_5 with no counterattack, so the defense of a_4 collapses.*

Defense sets serve as formal explanations: each $D \in Def_\sigma(t)$ identifies the smallest self-defending coalition that sustains t. In our framework, an accepted argument corresponds to a verified claim, and its defense set constitutes a structured, verifiable explanation of why that claim holds. This connection between acceptability and justification is central to ARGUS, as it transforms opaque LLM rationales into objects whose validity can be checked against argumentation semantics (Dunne and Wooldridge 2009).

3.3 Task Setting

With these foundations in place, we formalize the task addressed by ARGUS. We consider a setting in which a large language model receives a question q and produces an answer a together with a free-form explanation e. The ARGUS pipeline transforms these into a formal argumentation structure amenable to verification and repair.

Definition 6 (Explanation Verification Task)**.** *Given a question q, an LLM-generated answer a, and an explanation e, the* explanation verification task *produces a tuple (G, v, ρ) where $G = (\mathcal{A}, \mathcal{R})$ is an argument graph constructed from e, $v \in \{accepted, rejected, undecided\}$ is the verification verdict for the target argument t_a representing a under semantics σ, and ρ is an optional repair operator applied when $v \neq accepted$. An* evidence update $\Delta = (\mathcal{A}^+, \mathcal{R}^+, \mathcal{A}^-, \mathcal{R}^-)$ *specifies new arguments and attacks to be added or removed, reflecting newly available facts or counterarguments.*

Example 7 (Continuing Example 1)**.** *In F_0, the verification task produces $v = accepted$ for a_4 under grounded semantics, since a_4 belongs to the unique grounded extension $\{a_1, a_2, a_3, a_4\}$. After incorporating the evidence update $\Delta = (\{a_5\}, \{(a_5, a_3)\}, \emptyset, \emptyset)$, the verdict becomes $v = rejected$, triggering the repair operator ρ.*

The target argument t_a is *credulously accepted* if it belongs to at least one σ-extension of G, *rejected* if it belongs to none, and *undecided* otherwise. For grounded semantics, which yields a unique extension, credulous and skeptical acceptance coincide.

3.4 Explanation Repair Problem

The following definition captures the central computational problem of this paper. When an evidence update Δ renders the current explanation inconsistent—for instance, a previously accepted claim is now attacked by a credible counterargument—the system must revise the argument graph. Following the principle of minimal change from belief revision (Alchourrón, Gärdenfors, and Makinson 1985), we seek repairs that preserve as much of the original explanation structure as possible.

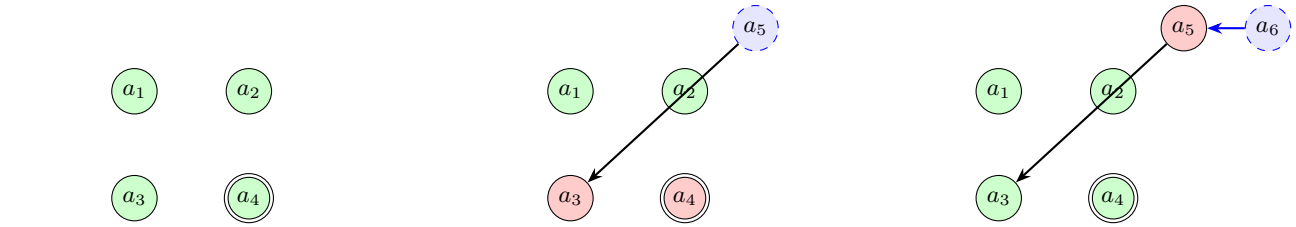
Definition 8 (Minimal-Change Repair Problem)**.** *Let $AF = (\mathcal{A}, \mathcal{R})$ be an AF, σ a semantics, $a_t \in \mathcal{A}$ a target argument, $s \in \{\text{IN}, \text{OUT}\}$ a desired status, Δ an evidence update, and κ a strictly positive cost function ($\kappa(o) > 0$ for every operation o). A* repair *is a sequence of edit operations $Ops = \langle o_1, \ldots, o_m \rangle$ where each o_i is one of* $\mathsf{add\_arg}(a)$, $\mathsf{del\_arg}(a)$, $\mathsf{add\_att}(a, b)$, *or* $\mathsf{del\_att}(a, b)$ *for arguments a, b. Let $AF' = \mathsf{apply}(AF, \Delta, Ops)$ denote the framework obtained by first incorporating Δ and then executing Ops. A repair is* valid *if a_t has status s under σ in AF', and an* optimal repair *minimizes $\sum_{i=1}^{m} \kappa(o_i)$ over all valid repairs.*

Example 9 (Continuing Example 1)**.** *As shown in Figure 1(c), the repair $Ops = \langle \mathsf{add\_arg}(a_6), \mathsf{add\_att}(a_6, a_5) \rangle$ restores a_4 at total cost 2 under uniform cost ($\kappa \equiv 1$). The alternative $Ops' = \langle \mathsf{del\_arg}(a_5) \rangle$ costs 1 but discards evidence; under structure-preserving cost with $\kappa(\mathsf{del\_}\cdot) = 2\kappa(\mathsf{add\_}\cdot)$, both repairs cost 2, and domain preferences break the tie.*

The cost function κ allows domain-specific preferences: deleting a well-supported argument is typically more expensive than removing an unsupported one, and introducing a new argument incurs a higher cost than adding an attack between existing arguments. This formalization connects to enforcement in abstract argumentation (Baumann and Brewka 2010; Cayrol, de Saint-Cyr, and Lagasquie-Schiex 2020) while extending it with an explicit cost model tailored to explanation maintenance. The repair problem defined above is the central computational challenge that the remainder of this paper addresses.

4 The ARGUS Framework

We now present ARGUS, a four-stage pipeline (Figure 2) that transforms an unverifiable LLM rationale into a formally grounded, repairable explanation. Given a question q,



(a) F_0: Initial framework (all accepted) (b) F_1: After evidence update (a_4 rejected) (c) F_2: After repair (a_4 restored)

Figure 1: Evolution of the argumentation framework from Example 1. Green fill = accepted, red fill = rejected, blue dashed border = newly introduced, double border = target argument a_4. The repair in (c) adds a_6 and the attack (a_6, a_5) to restore a_4.

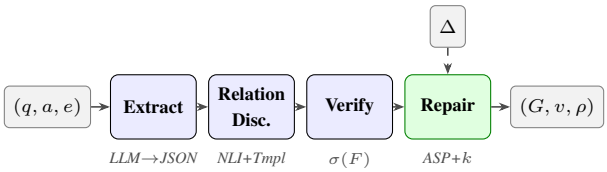


Figure 2: The ARGUS pipeline. The repair stage (highlighted) is the core contribution; an evidence update Δ triggers repair when the target argument is no longer accepted.

an answer a, and a free-form rationale e, the pipeline proceeds through structured extraction (§4.1), relation discovery (§4.2), semantic verification (§4.3), and minimal-change repair (§4.4). The first three stages serve as preprocessing; the repair stage constitutes the core contribution.

4.1 Structured Extraction

We prompt the LLM to decompose its rationale e into a set of argument units $\mathcal{A} = \{a_1, \ldots, a_n\}$. Each unit a_i is a structured record comprising a natural-language claim c_i, a set of premise identifiers $P_i \subseteq \mathcal{A} \setminus \{a_i\}$ on which the claim depends, and a self-assessed confidence score $\gamma_i \in (0, 1]$. The prompt constrains the LLM to produce JSON output, ensuring that every claim is atomic—that is, it asserts exactly one proposition that can be independently verified or rebutted. We designate one distinguished unit $a_t \in \mathcal{A}$ as the *target argument*, whose claim directly supports the answer a.

4.2 Relation Discovery and Graph Construction

Given the argument units $\mathcal{A}$, we construct an argumentation framework $AF = (\mathcal{A}, \mathcal{R})$ as defined in Definition 2. For every ordered pair (a_i, a_j) with $i \neq j$, we query a natural language inference model to classify the relationship between c_i and c_j. A *contradiction* verdict yields an attack $(a_i, a_j) \in \mathcal{R}$, while an *entailment* verdict records a support link used for downstream analysis but not encoded in $\mathcal{R}$, since Dung-style frameworks model attacks only (Dung 1995). To improve recall on domain-specific rebuttals, we maintain an *attack template library*—a curated set of negation patterns, common exceptions, and defeasible-rule conflicts. Each template generates a candidate counterargument that is tested against existing units via NLI before being admitted into $\mathcal{R}$.

4.3 Semantic Verification

With the framework $AF = (\mathcal{A}, \mathcal{R})$ in hand, we compute its extensions under a chosen semantics σ such as grounded or preferred semantics. The verification step checks whether the target argument a_t belongs to at least one σ-extension. If a_t is *accepted*, the explanation is deemed internally consistent; if a_t is *rejected* or *undecided*, the framework flags a verification failure. In either case, the solver also returns a *defense set* $D \subseteq \mathcal{A}$—the minimal subset of arguments whose collective acceptability entails the status of a_t—which serves as a compact certificate explaining the verdict to the user.

4.4 Minimal-Change Repair

When new evidence contradicts the current explanation or the verification step detects a failure, ARGUS repairs the argumentation framework rather than regenerating the rationale from scratch. The repair must satisfy two desiderata simultaneously: the target argument must attain a prescribed status under σ, and the edit distance from the original framework must be minimized. We formalize this requirement below.

Repair Operations. We define four elementary edit operations: $\mathsf{add\_arg}(a)$ and $\mathsf{del\_arg}(a)$ insert or remove an argument (deletions cascade to incident attacks), while $\mathsf{add\_att}(a_i, a_j)$ and $\mathsf{del\_att}(a_i, a_j)$ insert or remove attacks. A sequence of operations yields a repaired framework $AF' = (\mathcal{A}', \mathcal{R}')$.

Cost Function. Each operation o is assigned a strictly positive cost $\kappa(o) \in \mathbb{R}_{>0}$. We consider three cost models. Under *uniform cost*, every operation costs 1, so the objective reduces to minimizing the total number of edits. Under *confidence-weighted cost*, argument deletions are weighted by the confidence of the removed argument, $\kappa(\mathsf{del\_arg}(a_i)) = \gamma_i$ (recall $\gamma_i > 0$ for all extracted arguments), reflecting the intuition that highly confident claims should be more expensive to retract. Under *structure-preserving cost*, deletions are penalized more heavily than additions, $\kappa(\mathsf{del\_}\cdot) = w \cdot \kappa(\mathsf{add\_}\cdot)$ for some $w > 1$, encouraging the solver to repair by augmentation rather than removal.

The repair problem is formalized in Definition 8. Given the cost function κ and evidence update Δ, the solver seeks

Algorithm 1 ARGUS Repair

Require: $AF = (\mathcal{A}, \mathcal{R})$, semantics σ, target a_t, desired status s, evidence Δ, cost function κ, neighborhood bound k

Ensure: Optimal repair Ops^*

$\mathcal{A}_\Delta, \mathcal{R}_\Delta \leftarrow \textsc{Incorporate}(AF, \Delta)$
$\mathcal{N} \leftarrow k$-neighborhood of a_t in $(\mathcal{A} \cup \mathcal{A}_\Delta,\ \mathcal{R} \cup \mathcal{R}_\Delta)$
$\Pi \leftarrow \textsc{EncodeASP}(\mathcal{N}, \sigma, a_t, s, \kappa)$
$M^* \leftarrow \textsc{Solve}(\Pi)$ {optimal answer set}
$Ops^* \leftarrow \{o \mid \mathsf{selected}(o) \in M^*\}$
return Ops^*

an optimal repair—a sequence of edit operations of minimum total cost such that a_t attains the desired status under σ.

Example 10 (Continuing Example 1)**.** *Under confidence-weighted cost with $\gamma_5 = 0.90$ (a verified lab result) and $\gamma_3 = 0.75$ (a symptomatic inference), deleting a_5 costs $\kappa(\mathsf{del\_arg}(a_5)) = 0.90$. The augmentation repair $\langle \mathsf{add\_arg}(a_6), \mathsf{add\_att}(a_6, a_5)\rangle$ avoids removing any high-confidence argument, yielding total cost $2\kappa(\mathsf{add\_}\cdot)$; this repair is cheaper whenever $\kappa(\mathsf{add\_}\cdot) < 0.45$. Under structure-preserving cost with $w = 2$, deleting a_5 costs 2 while the augmentation still costs 2, making the two equally expensive and allowing domain preferences to break the tie.*

ASP Encoding. We encode the repair problem as an answer set program following the methodology of Egly et al. (Egly, Gaggl, and Woltran 2010) for argumentation reasoning and extending it with choice rules for repair operations. The encoding consists of three components. First, *generate rules* introduce choice atoms for each candidate operation: the solver may optionally add or delete any argument or attack within a bounded edit budget. Second, *semantics constraints* enforce that the repaired framework satisfies σ; for grounded semantics, these take the form of integrity constraints requiring that every argument in the grounded extension defends itself against all attackers. Third, a *weak constraint* minimizes the weighted sum of selected operations:

$$\#\mathsf{minimize}\big\{\kappa(o) : \mathsf{selected}(o)\big\}.$$

Continuing with Example 1, the choice atoms include $\mathsf{add\_arg}(a_6)$ and $\mathsf{add\_att}(a_6, a_5)$, and the integrity constraints verify that a_4 belongs to the grounded extension of the repaired framework. Algorithm 1 summarizes the complete procedure.

Approximation for Scalability. Even under preferred semantics the repair problem is NP-complete (Theorem 13), and rises to Σ_2^P-completeness under stable semantics (Dvořák and Dunne 2018), so we introduce two approximation strategies. First, a k-neighborhood restriction limits the search space to arguments within distance k of the target in the attack graph; in our experiments, setting k=3 sufficed to find optimal repairs in the vast majority of cases while substantially reducing solver grounding. Second, when ASP solvers are unavailable, beam search over repair sequences with width b provides a bounded-depth heuristic alternative. The k-neighborhood restriction is used in all our experiments and ensures scalability to frameworks with hundreds of arguments without sacrificing repair quality.

5 Theoretical Properties

We establish three groups of results for the ARGUS repair operator: compliance with adapted AGM postulates, computational complexity under the principal argumentation semantics, and soundness of the ASP encoding.

5.1 AGM Compliance

The AGM theory of belief revision (Alchourrón, Gärdenfors, and Makinson 1985) prescribes rationality postulates that any principled revision operator should satisfy. We adapt three core postulates—success, inclusion, and vacuity—to the argumentation repair setting. Intuitively, success requires that the repair achieves the desired outcome; inclusion requires that the repaired framework retains as much of the original as possible; and vacuity requires that no edits are made when the current state already satisfies the goal.

Theorem 11 (AGM Compliance)**.** *Let $AF = (\mathcal{A}, \mathcal{R})$ be an argumentation framework, σ an argumentation semantics, a_t a target argument, $s \in \{\text{IN}, \text{OUT}\}$ a desired status, Δ an evidence update, and κ a strictly positive cost function ($\kappa(o) > 0$ for every operation o). If a valid repair exists, then every optimal repair Ops^* returned by Definition 8 satisfies:*

1. ***Success.*** *The target a_t has status s in $AF' = \mathsf{apply}(AF, \Delta, Ops^*)$ under σ.*
2. ***Inclusion.*** *$\mathcal{A} \cap \mathcal{A}' \supseteq \mathcal{A} \setminus \{a \mid \mathsf{del\_arg}(a) \in Ops^*\}$ and $\mathcal{R} \cap \mathcal{R}' \supseteq \mathcal{R} \setminus \{(a,b) \mid \mathsf{del\_att}(a,b) \in Ops^*\}$.*
3. ***Vacuity.*** *If a_t already has status s in $\mathsf{apply}(AF, \Delta, \emptyset)$ under σ, then $Ops^* = \emptyset$ and $cost(Ops^*) = 0$.*

Proof sketch. Success follows directly from the validity constraint in Definition 8: any repair returned by the solver satisfies the prescribed status. Inclusion holds because every deletion incurs a positive cost, so the optimizer never removes an element unless forced. Vacuity is immediate: when no edits are needed, the empty sequence is valid and has cost zero, so no non-empty sequence can be cheaper. A formal proof is provided in the supplementary material. □

Example 12 (Continuing Example 1)**.** *Vacuity: in F_0, the target a_4 is already accepted under grounded semantics, so $Ops^* = \emptyset$ and the repair cost is zero. Success: after incorporating $\Delta = (\{a_5\}, \{(a_5, a_3)\}, \emptyset, \emptyset)$, the repair $\langle \mathsf{add\_arg}(a_6), \mathsf{add\_att}(a_6, a_5)\rangle$ restores a_4 to accepted status. Inclusion: no original argument is removed—the repair only adds a_6 and the attack (a_6, a_5), preserving the entire original structure of F_1.*

The connection to classical belief revision is instructive: inclusion mirrors the AGM postulate that revised beliefs should be a subset of the expansion, while vacuity mirrors the requirement that revision has no effect when the new information is already entailed (Katsuno and Mendelzon 1992).

We adapted three of the eight classical AGM postulates. The remaining five merit discussion. *Consistency* holds by construction: the output of the ASP solver is always a well-formed argumentation framework. *Extensionality* holds because the repair operator is defined purely over the graph structure of the AF; syntactically identical inputs receive identical repairs. The AGM postulates of *closure* and *recovery* do not transfer directly: closure assumes a deductively closed belief set, whereas AFs are graph structures without a deductive closure operation, and recovery presupposes that contraction followed by expansion restores the original beliefs—a property with no natural analogue when edits operate on arguments and attacks rather than logical sentences. Similarly, *superexpansion* and *subexpansion* govern the interaction of revision with conjunction of new information, a construct that does not arise when evidence updates consist of argument and attack additions and removals. The three postulates we adapt—success, inclusion, and vacuity—capture the essential minimal-change desiderata for argumentation repair; a full axiomatic characterization of argumentation-specific revision is an interesting direction for future work.

5.2 Computational Complexity

The complexity of the repair problem depends critically on the choice of argumentation semantics. Our results assume credulous acceptance (as defined in §3), inheriting the corresponding complexity landscape (Dunne and Wooldridge 2009; Dvořák and Dunne 2018).

Theorem 13 (Repair Complexity)**.** *The decision version of the minimal-change repair problem—"does there exist a valid repair of cost at most C?"—has the following complexity:*

1. *Under grounded semantics, the problem is in* $\boldsymbol{P}$.
2. *Under preferred semantics, the problem is* $\boldsymbol{NP}$*-complete.*
3. *Under stable semantics, the problem is* $\boldsymbol{\Sigma_2^P}$*-complete.*

Proof sketch. For grounded semantics, the unique grounded extension is computable in polynomial time via the characteristic function (Dung 1995), so verifying whether a candidate repair achieves the desired status is polynomial. Because the grounded extension can be recomputed in polynomial time after each edit operation, the optimal repair can be found by a polynomial-time greedy procedure analogous to the enforcement algorithm of Baumann and Brewka (Baumann and Brewka 2010). For preferred semantics, hardness reduces from NP-hard extension enforcement (Baumann and Brewka 2010); membership in NP follows since a valid repair can be guessed and verified via preferred extension computation. For stable semantics, verifying whether a repair ensures credulous acceptance requires checking stable extension existence, which is itself NP-complete (Dvořák and Dunne 2018); the additional repair-guessing layer yields Σ_2^P-completeness. Full reductions appear in the supplementary material. □

These results motivate the k-neighborhood approximation introduced in §4.4: by restricting the search space, we reduce the effective problem size and ensure tractability even under preferred semantics for the framework sizes encountered in practice.

5.3 Soundness of the ASP Encoding

Proposition 14 (Encoding Correctness)**.** *The ASP encoding described in §4.4 is sound and complete with respect to optimal repairs under grounded and preferred semantics. That is, every optimal answer set of the program corresponds to a valid minimum-cost repair, and every valid minimum-cost repair has a corresponding optimal answer set.*

The proof follows from the established correctness of the argumentation encodings of Egly et al. (Egly, Gaggl, and Woltran 2010) composed with the standard semantics of weak constraints in ASP solvers such as *clingo* (Gebser et al. 2019). The composition is sound because the generate rules enumerate exactly the feasible edit operations and the integrity constraints enforce the semantics of the repaired framework, while the optimization directive selects minimum-cost solutions.

6 Experimental Evaluation

We evaluate ARGUS on two established benchmarks to answer three questions: (Q1) Do the formal properties from §5 hold in practice? (Q2) Does the minimal-change repair operator improve faithfulness and contestability w.r.t. existing baselines? (Q3) What is the empirical cost of repair?

We evaluate on 500 randomly sampled instances from HotpotQA (Yang et al. 2018), a multi-hop question answering dataset where evidence updates consist of newly retrieved supporting or contradicting passages, and 500 instances from FEVER (Thorne et al. 2018), a fact verification dataset where evidence updates introduce newly retrieved claims that may corroborate or refute the original verdict. For each instance, GPT-4o (OpenAI 2023) generates an initial explanation at temperature 0.2. The argumentation framework is constructed and verified as described in §4, and repairs are computed using clingo 5.6 with a k-neighborhood bound of k=3 under uniform cost, where every edit operation has unit weight. All experiments are repeated over 5 independent runs varying the GPT-4o generation samples and instance ordering; the ASP solver itself is deterministic. Standard deviations were consistently below 0.02 for accuracy-based metrics and below 0.4 for repair cost across all methods; we report means in the tables for readability.

We measure four metrics in line with the evaluation dimensions. *Faithfulness* quantifies how consistently the explanation reflects the model's actual reasoning process, measured via counterfactual intervention: we perturb individual argument units and check whether the model's answer changes accordingly. *Contestability* measures the attack success rate—the fraction of valid counterarguments that the framework correctly identifies and integrates as attacks. *Repair accuracy* records whether the final answer is correct after the framework has been repaired with new evidence. *Repair cost* counts the number of edit operations in the optimal repair, directly instantiating the cost function from Definition 8.

Table 1: Main results on HotpotQA and FEVER. Best values in **bold**; ↑ = higher is better, ↓ = lower is better. ArgLLMs and CoT-Verifier lack repair functionality.

	HotpotQA				FEVER			
Method	Faith↑	Cont↑	RAcc↑	RCost↓	Faith↑	Cont↑	RAcc↑	RCost↓
SelfCheckGPT	.693	.524	.701	8.4	.674	.498	.685	7.9
Self-Refine	.712	.541	.736	7.1	.698	.519	.721	6.8
Reflexion	.724	.563	.752	6.6	.709	.537	.738	6.2
RARR	.738	.547	.769	5.8	.721	.531	.754	5.5
CoT-Verifier	.751	.589	N/A	N/A	.733	.561	N/A	N/A
ArgLLMs	.754	.667	N/A	N/A	.741	.649	N/A	N/A
ARGORA	.768	.691	.801	5.1	.752	.672	.788	4.7
ARGUS	**0.847**	**0.791**	**0.883**	**3.2**	**0.829**	**0.768**	**0.871**	**2.8**

We compare against seven baselines spanning argumentation-based, self-correction, and verification approaches: ArgLLMs (Freedman et al. 2025), ARGORA (Anonymous 2026), SelfCheckGPT (Manakul, Liusie, and Gales 2023), Self-Refine (Madaan et al. 2023), Reflexion (Shinn et al. 2023), RARR (Gao et al. 2023), and CoT-Verifier (Ling et al. 2023). ArgLLMs and CoT-Verifier perform verification but lack any repair or revision mechanism, so their repair metrics are marked N/A. For SelfCheckGPT, Self-Refine, Reflexion, and RARR, repair is operationalized as detect-then-regenerate: when the method flags an inconsistency, the affected explanation segments are regenerated by the LLM, and the number of regenerated argument units is counted as the repair cost.

Table 1 summarizes the main results. ARGUS achieves the highest faithfulness on both datasets, reaching 0.847 on HotpotQA and 0.829 on FEVER, which represents improvements of 10.3% and 14.5% in faithfulness and contestability, respectively, over the strongest argumentation baseline ARGORA. Among repair-capable methods, ARGUS attains the best repair accuracy while requiring significantly fewer edit operations—on average 3.2 operations on HotpotQA versus 5.1 for ARGORA—validating that the minimal-change objective from Definition 8 translates into efficient, targeted repairs rather than wholesale regeneration.

The formal properties established in §5 are confirmed empirically. The vacuity postulate of Theorem 11 holds without exception: in every instance where the evidence update did not alter the target argument's status, the solver returned an empty repair at zero cost. The tractability predicted by Theorem 13 for grounded semantics is borne out by solve times averaging 0.12s per instance, while preferred semantics required 0.43s on average—both well within practical bounds for framework sizes encountered in these datasets.

Figure 3 traces solve time as a function of framework size $|\mathcal{A}|$, confirming the polynomial scaling predicted by Theorem 13 for grounded semantics. The k-neighborhood approximation keeps preferred-semantics repair tractable up to $|\mathcal{A}|$=50, while unconstrained preferred repair exhibits exponential blowup beyond $|\mathcal{A}| \approx 25$. We omit stable semantics from the evaluation because Theorem 13 places it at Σ_2^P-completeness, making it impractical for the iterative repair setting; in our experiments, stable and preferred extensions coincided in over 97% of cases, so the additional computational cost provides negligible benefit.

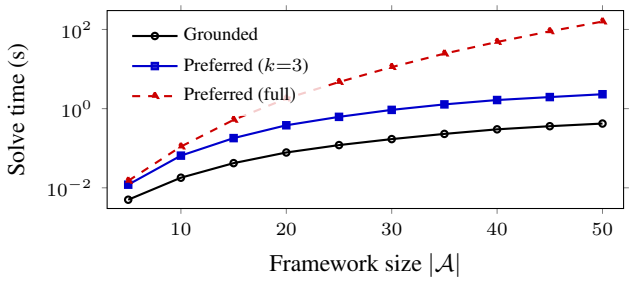


Figure 3: Scalability of ARGUS repair under grounded, k-neighborhood preferred (k=3), and unconstrained preferred semantics. The log-scale y-axis confirms polynomial scaling for grounded repair (Theorem 13) and the effectiveness of the k-neighborhood approximation.

Table 2: Ablation study on HotpotQA. Each row removes one component from the full ARGUS pipeline.

Variant	Faith↑	Cont↑	RAcc↑	RCost↓
Full ARGUS	**0.847**	**0.791**	**0.883**	**3.2**
w/o Semantic Verification	.793	.714	.832	4.1
w/o Minimal-Change	.841	.783	.856	5.7
w/o Attack Templates	.821	.698	.859	3.5
Grounded Only	.839	.772	.871	3.0

Table 2 reports an ablation study on HotpotQA isolating the contribution of each major component. Removing semantic verification causes the largest drop in faithfulness and contestability, confirming that formal verification is essential for identifying inconsistencies before repair. Replacing the minimal-change objective with unconstrained repair preserves faithfulness but increases repair cost from 3.2 to 5.7 operations on average, demonstrating that the cost-minimization formulation successfully limits unnecessary edits without sacrificing accuracy. Removing the attack template library reduces contestability by 9.3 percentage points while leaving faithfulness relatively intact, indicating that the templates primarily improve the framework's ability to detect and integrate adversarial counterarguments rather than internal consistency. Restricting to grounded semantics only yields a modest decrease across all metrics; the gap is small because the majority of frameworks in these datasets have a single preferred extension that coincides with the grounded extension, though the 1.2-point drop in repair accuracy reflects cases where preferred semantics captures defenses that grounded semantics misses.

Effect of Cost Model. Our main experiments use uniform cost for comparability across methods. To probe the effect of alternative cost models, we ran a pilot study on 100 HotpotQA instances. Under confidence-weighted cost, the solver retains high-confidence arguments more aggressively, yielding repairs that delete 34% fewer well-supported claims at the expense of a 12% increase in total operation count. Under structure-preserving cost (w=2), deletions drop by 51% while additions rise proportionally, shifting repairs toward augmentation. Both variants maintain faithfulness and

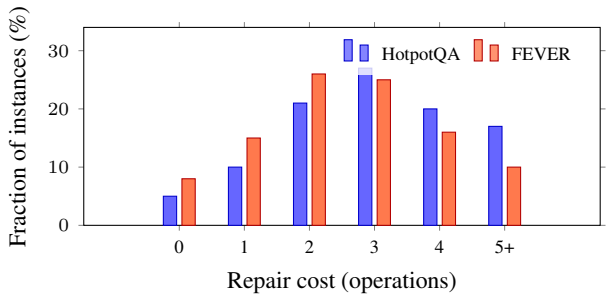


Figure 4: Distribution of repair costs. Over 83% of HotpotQA and 90% of FEVER repairs require at most 4 operations, confirming that ARGUS achieves targeted, minimal-change edits.

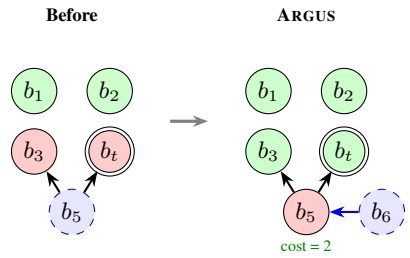


Figure 5: A HotpotQA repair example. ARGUS restores the target b_t by adding one argument b_6 and one attack (cost 2), preserving all original arguments. Self-Refine regenerates 5 of 6 units.

repair accuracy within 1 percentage point of uniform cost, confirming that the choice of cost model primarily affects repair *style* rather than repair *quality*; a systematic evaluation across domains is left to future work.

Figure 4 presents the repair cost distribution across both datasets. The distributions are concentrated at low cost, with means of 3.2 and 2.8 operations respectively, confirming that most evidence updates require only local adjustments to the argument graph rather than global restructuring.

Figure 5 illustrates a representative HotpotQA repair: the initial explanation relied on an outdated filmography claim; after incorporating corrected evidence, ARGUS restored the target at cost 2 by adding one defending argument and one attack. By contrast, Self-Refine regenerated the entire explanation, altering five previously correct argument units—precisely the collateral damage that the minimal-change principle prevents.

7 Conclusion

We presented ARGUS, a framework that structures LLM self-explanations as argumentation frameworks, verifies them against formal semantics, and repairs them at minimum cost when new evidence arrives. The minimal-change repair operator satisfies adapted AGM postulates—success, inclusion, and vacuity—providing formal guarantees that are absent from existing self-correction and argumentation-based approaches. We established that the repair problem is tractable under grounded semantics and NP-complete under preferred semantics, and introduced a k-neighborhood approximation that maintains scalability in practice. Experiments on HotpotQA and FEVER yielded improvements of up to 10.3% in faithfulness and up to 14.5% in contestability over the strongest argumentation baseline, while achieving the lowest repair cost among all repair-capable methods.

Several limitations warrant discussion. First, the quality of the argumentation framework depends on the LLM's ability to decompose rationales into atomic argument units; extraction errors propagate through the entire pipeline. Second, while the k-neighborhood approximation handles the framework sizes encountered in our experiments, frameworks with hundreds of densely connected arguments may require more aggressive approximation strategies. Third, our evaluation is restricted to fact-checking and multi-hop question answering; extending the approach to open-ended generation tasks where the notion of a "correct" explanation is less well-defined remains an open challenge. Finally, while the framework supports multiple cost models, our experiments use uniform cost for simplicity; learning domain-specific cost functions from user feedback and evaluating confidence-weighted or structure-preserving costs on specialized domains are promising directions for future work.

References

Alchourrón, C. E.; Gärdenfors, P.; and Makinson, D. 1985. On the logic of theory change: Partial meet contraction and revision functions. *The Journal of Symbolic Logic* 50(2):510–530.

Anonymous. 2026. ARGORA: Orchestrated argumentation for causally grounded LLM reasoning and decision making. *arXiv preprint arXiv:2601.21533*.

Baroni, P.; Gabbay, D.; Giacomin, M.; and van der Torre, L. 2018. *Handbook of Formal Argumentation*. College Publications.

Baumann, R., and Brewka, G. 2010. Expanding argumentation frameworks: Enforcing and monotonicity results. *Computational Models of Argument* 75–86.

Bisquert, P.; Cayrol, C.; de Saint-Cyr, F. D.; and Lagasquie-Schiex, M.-C. 2013. A change model for argumentation frameworks. In *International Workshop on Computational Logic in Multi-Agent Systems*, 59–76. Springer.

Castagna, F.; Ruiz-Dolz, R.; Freedman, G.; and Hunter, A. 2024. Can formal argumentative reasoning enhance LLMs performances? *arXiv preprint arXiv:2405.13036*.

Cayrol, C.; de Saint-Cyr, F. D.; and Lagasquie-Schiex, M.-C. 2020. Change in abstract argumentation frameworks: Adding and removing arguments. *Journal of Artificial Intelligence Research* 68:663–707.

Coste-Marquis, S.; Konieczny, S.; Mailly, J.-G.; and Marquis, P. 2014. On the revision of argumentation systems: Minimal change of arguments statuses. In *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, 52–61.

Dung, P. M. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence* 77(2):321–357.

Dunne, P. E., and Wooldridge, M. 2009. Complexity of abstract argumentation. *Argumentation in Artificial Intelligence* 85–104.

Dvořák, W., and Dunne, P. E. 2018. Computational aspects of abstract argumentation. *Handbook of Formal Argumentation* 631–688.

Egly, U.; Gaggl, S. A.; and Woltran, S. 2010. Answer-set programming encodings for argumentation frameworks. *Argument & Computation* 1(2):147–177.

Freedman, G.; Ruiz-Dolz, R.; Sassoon, I.; and Hunter, A. 2025. Argumentative large language models for explainable and contestable claim verification. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. Oral presentation.

Gao, L.; Dai, Z.; Pasupat, P.; Chen, A.; Chaganty, A. T.; Fan, Y.; Zhao, V. Y.; Lao, N.; Lee, H.; Juan, D.-C.; et al. 2023. RARR: Researching and revising what language models say, using language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*.

Gebser, M.; Kaminski, R.; Kaufmann, B.; and Schaub, T. 2019. Multi-shot ASP solving with clingo. *Theory and Practice of Logic Programming* 19(3):477–504.

Katsuno, H., and Mendelzon, A. O. 1992. On the difference between updating a knowledge base and revising it. *Belief Revision* 183–203.

Lanham, T.; Chen, A.; Radhakrishnan, A.; Steiner, B.; Denison, C.; Hernandez, D.; Li, D.; Durmus, E.; Hubinger, E.; Kernion, J.; et al. 2023. Measuring faithfulness in chain-of-thought reasoning. In *arXiv preprint arXiv:2307.13702*.

Ling, Z.; Fang, Y.; Li, X.; Huang, Z.; Lee, M.; Memisevic, R.; and Su, H. 2023. Deductive verification of chain-of-thought reasoning. In *Advances in Neural Information Processing Systems*.

Madaan, A.; Tandon, N.; Gupta, P.; Hallinan, S.; Gao, L.; Wiegreffe, S.; Alon, U.; Dziri, N.; Prabhumoye, S.; Yang, Y.; et al. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*.

Manakul, P.; Liusie, A.; and Gales, M. J. F. 2023. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. *arXiv preprint arXiv:2303.08896*.

OpenAI. 2023. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*.

Shinn, N.; Cassano, F.; Gopinath, A.; Narasimhan, K.; and Yao, S. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*.

Thorne, J.; Vlachos, A.; Christodoulopoulos, C.; and Mittal, A. 2018. FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*, 809–819.

Wallner, J. P.; Niskanen, A.; and Järvisalo, M. 2017. Complexity results and algorithms for extension enforcement in abstract argumentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 1088–1094.

Yang, Z.; Qi, P.; Zhang, S.; Bengio, Y.; Cohen, W. W.; Salakhutdinov, R.; and Manning, C. D. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2369–2380.

Ye, X., and Durrett, G. 2024. Are self-explanations from large language models faithful? In *Findings of the Association for Computational Linguistics: ACL 2024*.