Composable Assurance for AI Alignment: A Framework for Propagating Formal Safety Properties Through MLOps

Xiaofei Zhao[1 2]
[1]Lanzhou Petrochemical University of Vocational Technology
No. 1, Shandan Street Xigu District, Lanzhou City, Gansu Province, China
[2]Lanzhou Jiaotong University
No. 88, Anning West Road Anning District, Lanzhou City, Gansu Province, China
xifizhao@gmail.com

Abstract

The increasing complexity of modern AI systems exposes a significant assurance gap: safety evidence from practices like red-teaming and robustness testing remains fragmented, lacking a formal mechanism for composition and propagation throughout the development lifecycle. This prevents the construction of rigorous, dynamic safety cases essential for trustworthy AI. We introduce the Composable Assurance Framework (CAF), a novel engineering methodology that integrates safety assurance directly into MLOps workflows. At its core is the Formal Safety Assertion (FSA), a standardized, machine-readable structure that verifiably links safety properties—such as robustness scores or the absence of deceptive circuits—to specific AI artifacts. We then define a Composition Calculus, a set of formal rules governing how FSAs are propagated and aggregated as components are combined into a system. This approach transforms the development pipeline into an automated evidence-gathering engine, whose output is a dynamic Directed Acyclic Graph (DAG) of assertions that constitutes a living safety case. Through a prototype and a Retrieval-Augmented Generation (RAG) case study, we demonstrate how CAF automatically enforces a predefined safety policy, blocking non-compliant deployments.

Introduction

AI safety research is rightly moving from evaluating isolated models to assuring entire systems, as risks often emerge from the interaction of data pipelines, models, and external tools (Pimpale et al. 2025; Shevlane et al. 2023). However, engineering practices lag behind, creating an "assurance gap." Safety evaluations like adversarial robustness testing (Croce and Hein 2020; Szegedy et al. 2013), red-teaming (Perez et al. 2022; Mazeika et al. 2024), and mechanistic interpretability (Olah et al. 2020) are often performed as powerful but disparate, siloed activities whose results are not formally structured for automated use. Without a way to compose this fragmented evidence or propagate guarantees, a model deemed safe in a lab can become a vulnerability in production, making true, end-to-end system safety impossible to verify. This assurance gap directly impedes the construction of rigorous, dynamic *safety cases*—structured, evidence-based arguments for system safety, a practice central to safety-critical engineering (Kelly 1999a). However, traditional methods for building these cases are manual and document-centric, rendering them fundamentally incompatible with the agile, iterative nature of MLOps, a challenge widely recognized in the field (Wang and Chung 2022; Schuett 2024). There is a pressing need for an engineering mechanism to automate evidence aggregation and maintain a dynamic, formally-grounded safety case. Our work aims to fill this void. To do so, this paper addresses three fundamental research questions: **RQ1: Formal Representation.** How can we formally define and represent the diverse safety properties of heterogeneous AI components—such as data, models, and tools—in a standardized, machine-readable format that captures their context, metrics, and evidence? **RQ2: Compositional Logic.** What formal rules and calculus govern the propagation and composition of these safety properties as their underlying components are integrated within a complex MLOps pipeline? This includes defining how properties are inherited, transformed, or aggregated across different stages. **RQ3: Automated Governance.** How can this formal assurance framework be practically integrated into existing Continuous Integration/Continuous Deployment (CI/CD) workflows to enable automated safety governance, such as pre-deployment compliance checks and dynamic policy enforcement?

CAF in the AI Assurance Landscape

AI assurance is a burgeoning and multi-faceted field, spanning efforts from mathematical proofs to high-level governance frameworks (AI 2023; Hendrycks et al. 2021). To clearly articulate the unique contribution of the Composable Assurance Framework (CAF), this section situates it within this broader landscape. We argue that CAF does not seek to replace existing efforts but instead acts as a critical **synthesis layer**, designed to integrate, connect, and operationalize the outputs of three foundational pillars of this ecosystem: traditional safety engineering principles, formal verification methods, and the diverse suite of modern test and evaluation (T&E) methodologies. We begin by acknowledging that CAF shares a foundational goal with established assurance frameworks from safety-critical domains, such as DO-178C for aviation (Johnson 1998) and ISO 26262 for automotive (Nah et al. 2017). This goal is the construction of a struc-

Dimension	Traditional Assurance	CAF
Lifecycle Model	Sequential Phases	Continuous Iteration (CI/CD)
Core Artifact	Document $\mathcal{D}$ (Human-readable)	Formal Safety Assertion (FSA)
Evidence Generation	$SC_{\text{trad}} \leftarrow \text{ManualBuild}(E)$	$SC_{\text{CAF}}(t) = f_{\text{compose}}(E'(t))$
Nature of Assurance	Static, Point-in-Time: SC_{trad}	Dynamic, "Living" Case: $SC_{\text{CAF}}(t)$
System Model	Deterministic: $y = S(x)$	Stochastic: $P(y\|x) = S_\theta(x)$
Change Management	Gating: Human Approval	Gating: Automated Policy $\Pi(FSA_\Delta) \in \{\text{Go, NoGo}\}$

Table 1: A simplified, symbolic comparison of assurance paradigms. Here, SC denotes the Safety Case, E the evidence set, t time, $S(x)$ the system's output for input x, and Π the governance policy.

tured, evidence-based argument—a *safety case*—that a system is acceptably safe for its intended use (Kelly 1999b; Inge 2019). The principles of rigor, traceability, and evidence-based reasoning established by these frameworks are the bedrock of any trustworthy system.

However, the *implementation* of these principles is deeply intertwined with the development paradigm for which they were designed. As illustrated in Table 1, the fundamental incompatibility between traditional, waterfall-based processes and the agile, iterative nature of modern MLOps necessitates a new approach (Tambon et al. 2022; Nešić, Nyberg, and Gallina 2019). The incompatibilities highlighted above are not superficial but fundamental. Attempting to directly apply a document-centric, multi-year certification process to an MLOps pipeline that deploys new models weekly, if not daily, is untenable. It would not only stifle agility but, more critically, fail to effectively manage the dynamic and emergent risks unique to AI systems (Wei et al. 2022). This fundamental friction is the primary motivation for CAF. Therefore, CAF does not seek to replace the foundational *principles* of safety engineering. Instead, it offers a *modern implementation* of them. It acts as a necessary **bridge**, translating the time-tested principles of rigor from traditional domains into the highly automated, continuously evolving practice of modern AI development. It transforms the static, document-based safety case into a dynamic, machine-readable graph of assurances, and the manual review board into an automated CI/CD safety gate.

CAF & AI Assurance

While sharing the goal of constructing an evidence-based safety case (SC) with traditional frameworks, CAF's implementation diverges to address the MLOps paradigm shift. As formalized in Table 1, CAF replaces the static, manually-built safety case (SC_{trad}) of traditional assurance with a dynamic, "living" case, $SC_{\text{CAF}}(t)$, automatically composed of machine-readable FSAs. This renders traditional methods untenable for MLOps and motivates CAF as a modern bridge for translating established safety principles into the automated context of AI development. As illustrated in Figure 1, CAF acts as this crucial synthesis layer, addressing the fragmentation of key assurance pillars summarized in Table 2. Specifically, CAF provides a unifying infrastructure

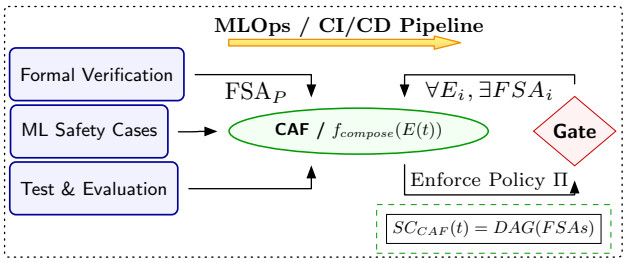

Figure 1: CAF as a Synthesis Layer for AI Assurance in MLOps.

that operationalizes formal proofs ($P \rightarrow \text{FSA}_P \in \text{MLOps}$) from verifiers like (Katz et al. 2017; Singh et al. 2018); automates the creation of the safety case ($\text{ManualBuild}(E) \rightarrow f_{\text{compose}}(E(t))$), providing an engineering methodology for the living assurance cases envisioned in (Langford et al. 2023); and provides a *lingua franca* to unify heterogeneous test evidence ($\bigoplus E_i$ is undefined $\rightarrow$ $\bigoplus \text{FSA}_i$ is defined) from a wide array of tools covering robustness (Croce and Hein 2020), fairness (Bellamy et al. 2018), red-teaming (Mazeika et al. 2024), and explainability (Lundberg and Lee 2017). Within the AI assurance landscape, CAF acts as a crucial synthesis layer. As illustrated in Figure 1 and summarized in Table 2, it addresses the fragmentation of existing pillars by providing a unifying, operational infrastructure. Specifically, it operationalizes formal proofs ($P \rightarrow \text{FSA}_P \in$ MLOps) from a wide range of verification techniques (Katz et al. 2017; Gehr et al. 2018; Meng et al. 2022); it provides an engineering methodology for the dynamic, "living" assurance cases ($\text{ManualBuild}(E) \rightarrow f_{\text{compose}}(E(t))$) envisioned in the literature (Langford et al. 2023); and it provides a *lingua franca* to unify the fragmented and heterogeneous evidence set ($\bigoplus E_i$ is undefined $\rightarrow$ $\bigoplus \text{FSA}_i$ is defined) from a diverse array of Test & Evaluation toolkits and benchmarks covering adversarial robustness (Croce and Hein 2020), fairness (Bellamy et al. 2018), explainability (Lundberg and Lee 2017), privacy attacks (Liu et al. 2022), data quality (Southorn 2017), and automated red-teaming (Mazeika et al. 2024; Wei, Haghtalab, and Steinhardt 2023). By doing so, CAF connects these powerful but disparate activities into a

single, coherent, and actionable assurance network.

Pillar 1: Formal Verification as Trustworthy FSA Generators Within the AI assurance landscape, CAF acts as a crucial synthesis layer. It addresses the fragmentation of existing pillars by providing a unifying, operational infrastructure. Specifically, it operationalizes formal proofs, P, by transforming them from isolated reports into integrated $\text{FSA}_P \in$ MLOps, leveraging a rich body of work in formal verification (Katz et al. 2017; Gehr et al. 2018; Meng et al. 2022). It provides the engineering methodology to evolve the manually constructed, static safety case, $SC(t_0)$, envisioned by traditional methods (Kelly 1999b), into a dynamic, 'living' one, $SC_{\text{CAF}}(t) = \text{DAG}(\text{FSAs})$, addressing known challenges in certifying modern systems (Tambon et al. 2022). Finally, it provides a *lingua franca* to unify the fragmented and heterogeneous evidence set $\{E_i\}$ from Test & Evaluation—ensuring that a composition operator, $\bigoplus$, is well-defined over their corresponding FSA representations ($\bigoplus \text{FSA}_i$). This integrates findings from diverse toolkits covering adversarial robustness (Croce and Hein 2020), automated red-teaming (Mazeika et al. 2024; Wei, Haghtalab, and Steinhardt 2023), and data quality (Southorn 2017). By doing so, CAF connects these powerful but disparate activities into a single, coherent, and actionable assurance network.

Pillar 2: ML Safety Cases as an Engineering Approach for "Living" Safety Cases The goal of adapting the safety case concept to ML is to produce a structured argument, SC, for a system's safety (Kelly 1999b), ideally as a "Living Safety Case" that evolves over time to manage uncertainty in learning-enabled systems (Langford et al. 2023). The primary obstacle is that traditional practice relies on manual, document-centric processes that are fundamentally incompatible with MLOps (Tambon et al. 2022), yielding only a static snapshot of assurance: $\text{ManualBuild}(E) \rightarrow SC(t_0)$. CAF provides the first concrete engineering methodology to realize the "living" ideal. It replaces the manual process with an automated Composition Calculus, f_{compose}, that operates on a time-varying evidence set, $E(t)$. The output is no longer a static document, but a dynamic, machine-maintained safety case, $SC_{\text{CAF}}(t)$, whose structure is a Directed Acyclic Graph of FSAs, i.e., $SC_{\text{CAF}}(t) = \text{DAG}(\{\text{FSA}_i(t)\})$. CAF thus transforms the safety case from a manual snapshot, $SC(t_0)$, into an automated, real-time stream, $SC_{\text{CAF}}(t)$, that remains synchronized with the system's state.

Pillar 3: Test & Evaluation as the "Connective Tissue" for Diverse Evidence The proliferation of Test & Evaluation (T&E) tools provides a wealth of empirical evidence on properties from adversarial robustness (Croce and Hein 2020) and fairness (Bellamy et al. 2018) to automated red-teaming (Mazeika et al. 2024) and privacy attacks (Liu et al. 2022). However, this evidence exists as a fragmented and heterogeneous set of artifacts, $\{E_1, E_2, \ldots, E_n\}$, spanning disparate formats. The heterogeneity of these artifacts makes a formal composition operator, $\bigoplus$, ill-defined over the set, i.e., $\bigoplus E_i$ is undefined, creating information silos that preclude holistic safety assessment. CAF addresses this by acting as a **connective tissue**, providing the FSA as a *lingua franca*. By ensuring that for every piece of evidence, $\forall E_i$, there exists a corresponding standardized representation, $\exists \text{FSA}_i$, the framework makes the composition operator well-defined. The operation $\bigoplus \text{FSA}_i$ becomes meaningful, allowing these once-disparate evidence streams to be formally aggregated into a single, coherent assurance graph, governed by a unified safety policy.

Listing 1: FSA Example: Model Robustness Assertion

```
Pid := Robustness.Adv.L2
v   := 0.85
M   := "Accuracy under AutoAttack(
    epsilon = 0.5)"
E   := (H(A_model), "blob:storage/logs/
    run-12345.log")
C   := { (model_arch, ResNet50),
         (dataset, CIFAR-10),
         (torch_version, 2.0) }
D   := "Robustness against L2-norm
    adversarial attacks..."
```

The Composable Assurance Framework

The CAF provides a formal, engineering-driven methodology for building and maintaining safety guarantees for complex AI systems. It is built upon a foundational data structure designed to be the atomic unit of assurance: the Formal Safety Assertion.

Formal Safety Assertions (FSAs): The Atomic Unit of Assurance

At the heart of our framework lies the **Formal Safety Assertion (FSA)**, the atomic unit of assurance. An FSA is a standardized data structure creating a verifiable, immutable link between a digital artifact, $\mathcal{A}$, and a specific safety claim. Designed for automated processing in MLOps pipelines, an FSA is formally defined as a 6-tuple: $\text{FSA}(\mathcal{A}) \triangleq (\textit{Pid}, D, M, v, E, C)$ This structure ensures each claim is precise and auditable. The machine-readable *Pid* (PropertyID) defines the claim's semantics (e.g., `Robustness.Adv.L2`), which is measured by a precise *M* (Metric) to produce the *v* (Value). The human-readable *D* (Description) provides clarity. Verifiability is guaranteed by the *E* (Evidence), which immutably binds the assertion to the artifact via its cryptographic hash $H(\mathcal{A})$ and links to evaluation logs, while the *C* (Context) scopes the claim's validity through a set of key-value pairs describing environmental conditions.

Example. Consider an FSA generated from a robustness evaluation of a trained image classification model, $\mathcal{A}_{\text{model}}$. The resulting assertion $\text{FSA}(\mathcal{A}_{\text{model}})$ is instantiated as follows: This structured, formal approach ensures that every safety claim is a precise, verifiable, and context-aware artifact, ready for interpretation by both humans and automated governance systems within the MLOps lifecycle.

Pillar	Challenge: Isolated Artifacts	CAF's Contribution: Synthesis
Formal Verification	Proof P is static, exists outside the workflow. $P \to \text{Report}(P) \notin \text{MLOps}$	Operationalizes proofs into the workflow. $\text{CAF}(P) \to \text{FSA}_P \in \text{MLOps}$
ML Safety Cases	Process is manual, output is a static snapshot. $\text{ManualBuild}(E) \to SC(t_0)$	Automates generation of a dynamic, living case. $f_{\text{compose}}(E(t)) \to SC_{\text{CAF}}(t) = \text{DAG}(\text{FSAs})$
Test & Evaluation	Evidence is a fragmented, heterogeneous set. $\{E_1, E_2, \ldots, E_n\}; \bigoplus E_i$ is undefined	Provides a *lingua franca* to unify evidence. $\forall E_i, \exists \text{FSA}_i; \bigoplus \text{FSA}_i$ is defined

Table 2: Symbolic representation of CAF as a synthesis layer for AI assurance pillars. Here, P is a formal proof, $SC(t)$ is the safety case at time t, E_i is a piece of evidence, and $\bigoplus$ is a composition operator.

The Composition Calculus

While Formal Safety Assertions (FSAs) provide a standardized, static description of a property, the core innovation of CAF lies in its **Composition Calculus**. This calculus provides a set of formal rules to reason about the dynamic evolution of safety guarantees as artifacts are transformed and combined within the MLOps pipeline. This enables automated reasoning about the emergent safety properties of the system as a whole. We define this calculus through three fundamental operations: Propagation, Aggregation, and Dependency. The first of these, **Propagation**, governs how safety properties are inherited and transformed through sequential processes. Let an artifact $\mathcal{A}$ have an associated set of assertions, denoted $\text{FSA}(\mathcal{A})$. When a transformation T is applied to produce a new artifact, $\mathcal{A}' = T(\mathcal{A})$, the propagation rule defines the corresponding transformation on the assertions. For each input assertion $F \in \text{FSA}(\mathcal{A})$, its transformed counterpart, $F' \in \text{FSA}(\mathcal{A}')$, is computed by a propagation function, f_{prop}: $F' = f_{\text{prop}}(F, T)$.

Principled Transformation: Modeling Evolving Risks A core principle of the Composition Calculus is that the propagation function, f_{prop}, must model evolving risks, which may require transforming an assertion's semantic type, not merely its value. For instance, a dataset $\mathcal{D}$ may have a boolean assertion, `Data.Privacy.PIIFree = true`. However, the training process, T_{train}, introduces the emergent and probabilistic risk of model memorization, rendering the original deterministic property semantically insufficient for the new model artifact, $\mathcal{M}$. Consequently, f_{prop} executes a principled type transformation, mapping the boolean assertion for $\mathcal{D}$ to a new probabilistic assertion, `Model.Privacy.MemorizationResistance`, for $\mathcal{M}$, with a value $v' \in [0, 1]$. This value v' is not arbitrary but is quantified empirically, for example, by the model's success rate against a suite of membership inference attacks (Liu et al. 2022). This transformation from a deterministic check to a probabilistic measure demonstrates the calculus's capacity as a dynamic risk management framework, able to formally model how risks emerge and evolve through the MLOps lifecycle.

Aggregation: Combining Properties from Parallel Components **Aggregation** defines how safety properties from multiple, parallel input components are combined when those components are integrated into a single new artifact. This operation is essential for reasoning about the safety of systems built from multiple models, data sources, or other sub-components. Given a set of input artifacts $\{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$, each with their own sets of assertions $\{\text{FSA}(\mathcal{A}_1), \ldots, \text{FSA}(\mathcal{A}_n)\}$, and a combination process T, the resulting artifact $\mathcal{A}' = T(\mathcal{A}_1, \ldots, \mathcal{A}_n)$ will have a new assertion computed by an aggregation function, f_{agg}: $F' = f_{\text{agg}}(\{F_1, \ldots, F_n\}, T)$ where each F_i is an assertion about a relevant property from the corresponding input artifact $\mathcal{A}_i$. This is common in scenarios like model ensembling or data fusion.

Example. Consider two models, $\mathcal{M}_1$ and $\mathcal{M}_2$, being combined into a voting ensemble, $\mathcal{M}_{\text{ensemble}}$. $\mathcal{M}_1$ has a robustness assertion, $F_{\text{robust},1}$, with a value of 0.8, and $\mathcal{M}_2$ has a corresponding assertion, $F_{\text{robust},2}$, with a value of 0.6. A conservative aggregation rule, f_{agg}, for the ensemble's robustness might be to take the minimum value of its components. This would result in a new assertion for the ensemble, $F_{\text{robust,ensemble}}$, with a value of 0.6. More complex functions could also be defined depending on the ensembling strategy, such as a weighted average if the models' votes are weighted.

Dependency: Building a Structured Safety Argument The final operation, **Dependency**, defines a logical or conditional relationship between different properties for the *same* artifact. It establishes that the validity or meaning of one high-level safety property is contingent upon the successful verification of another, more foundational property. This creates a directed acyclic graph (DAG) of assurances for a single component, forming the backbone of a structured assurance argument. A natural question arises: why introduce a new 'Dependency' atom instead of simply listing all low-level requirements directly in a final safety policy? The motivation is to formally **decouple the *definition* of a property from its *enforcement***. This design choice yields critical advantages in modularity, reusability, and maintainability. Consider the high-level property of a RAG system being "truthful." Without a dependency rule, a safety policy must explicitly enumerate all of its constituent requirements, leading to a brittle, monolithic structure as shown in Listing 2. In contrast, CAF uses a dependency rule to first create a modular, high-level assertion for truthfulness. This assertion *defines* the property by formally declaring its dependencies on

Listing 2: A brittle, flat safety policy without dependency rules.

```
# Flat Policy: All low-level details are hardcoded
def check_deployment_policy(system_fsas):
    # Definition of "truthful" is mixed with enforcement
    data_ok = system_fsas.get("Data.Accuracy").value == "Certified-v3"
    retriever_ok = system_fsas.get("Retrieval.Recall@5").value > 0.90
    # Other system checks
    generator_ok = system_fsas.get("Gen.JailbreakResilience").value > 0.95
    #If the definition of truthful changes, this code MUST be modified.
    return data_ok and retriever_ok and generator_ok
```

Listing 3: A robust, modular policy enabled by dependency rules.

```
# Modular Policy: Focuses on high-level goals
def check_deployment_policy(system_fsas):
    # 1. Enforcement of "truthful" property
    # The system automatically traverses the dependency graph (Eq. 1)
    # to verify the underlying evidence for FSA_Truthful.
    truthful_ok = system_fsas.get("System.Truthful").is_valid()
    # 2. Other system checks
    generator_ok = system_fsas.get("Gen.JailbreakResilience").value > 0.95
    return truthful_ok and generator_ok
```

lower-level evidence:

$$\text{IsValid}(\text{FSA}_{\text{Truthful}}) \implies \text{IsVerified}(\text{FSA}_{\text{Data.Accuracy}}) \wedge \text{IsVerified}(\text{FSA}_{\text{Retrieval.Recall@5}}) \tag{1}$$

This encapsulated definition makes the final enforcement policy clean, elegant, and robust, as shown in Listing 3. This modular approach provides three distinct advantages. First, it enables **Modularity and Reusability**; the $\text{FSA}_{\text{Truthful}}$ becomes a self-contained, reusable assurance module that any policy can invoke without needing to know its internal definition. Second, it creates a **Structured Safety Argument**; the dependency graph is not merely a checklist but an auditable, logical argument that explains *why* the system is believed to be truthful. Finally, it ensures **Maintainability**; if the definition of truthfulness evolves to include a new dependency, only the definition of $\text{FSA}_{\text{Truthful}}$ needs to be updated. All policies that enforce it remain unchanged, a critical feature for managing complex systems at scale.

Scalability & Practical Deployment Considerations

While theoretical elegance is important, the practical scalability of an assurance framework is paramount for its adoption in industrial settings, which may involve thousands of components and hundreds of builds per day. This section analyzes the computational efficiency of CAF and describes a key architectural design that ensures its feasibility in large-scale MLOps environments.

Computational Complexity of Core Operations The core operations of CAF are designed to be lightweight, ensuring they do not become a bottleneck in fast-paced CI/CD pipelines.

FSA Generation and Verification. The atomic unit of the framework, the FSA, is computationally inexpensive. Generating an FSA primarily involves a **cryptographic hash** (e.g., SHA-256) of its associated artifact, an efficient operation that is linear in the size of the artifact, $O(N)$. Verifying an existing FSA is even faster, requiring only a hash comparison and an optional digital signature check (e.g., ECDSA), both of which are near-constant time operations, $O(1)$, and are negligible in the context of a CI/CD run.

Safety Policy Evaluation. The evaluation of a complete safety case at a deployment gate involves traversing the assurance graph. For a DAG with V FSAs (vertices) and E dependency relationships (edges), a full traversal has a time complexity of $O(V+E)$. In typical large-scale ML projects, the number of artifacts and dependencies would be in the hundreds or low thousands, not millions. This linear complexity means that a full check can be executed in seconds, well within the time budget for rapid feedback in a CI/CD workflow.

Architectural Optimization: Incremental Verification A naive implementation might inefficiently re-verify the entire assurance graph for every change. However, CAF's architecture is designed to support **incremental verification**, a crucial optimization for scalability. In a practical deployment, the system maintains a cached state of validated FSAs. When a change is triggered in a component, such as an updated dataset $\mathcal{D}$, the system identifies this artifact as a "dirty" node. It then traverses the assurance graph forward to identify all downstream artifacts that depend on it—the affected sub-graph—and re-runs the composition calculus and verification checks only for this localized scope. The validity of all FSAs in unaffected branches of the graph is loaded directly from the cache. This incremental approach reduces the computational cost of each check from being proportional to the entire project's size to being proportional only to the scope of the change. Since most CI/CD runs involve localized changes, the practical overhead of CAF remains minimal, demonstrating its scalability and efficiency for high-frequency development workflows.

Case Study: RAG Chatbot Assurance

To demonstrate the end-to-end workflow of the CAF, we present a case study of a Retrieval-Augmented Generation (RAG) chatbot. This system, with its multiple interacting

components (a knowledge base, a retrieval model, and a generative model), is representative of modern, complex AI applications where safety is an emergent property. The primary objective of this evaluation is to serve as an in-depth **proof-of-concept**. Rather than claiming universal applicability from a single example, our goal is to clearly **walk through** a complete assurance lifecycle. We will show how individual component assertions are created, how they are transformed via propagation and combined via dependency rules into a coherent system-level safety case, and finally, how that case is enforced by an automated deployment gate and maintained via a monitoring feedback loop. The value of this case study lies in its illustrative depth, not its breadth.

Stage 1 & 2: Propagation During Data Evolution

The assurance lifecycle begins with the foundational components. Our RAG system relies on a knowledge base, the dataset $\mathcal{D}$, which is the source of its factual claims.

Initial Assertion. We begin by establishing a baseline of trust. The initial dataset, $\mathcal{D}_{\text{v1}}$, is certified against a trusted source, resulting in a foundational FSA, F_1. This assertion attests to its high factual accuracy:

- **FSA F_1 for $\mathcal{D}_{\mathbf{v1}}$**:
- *PropertyID*: `Factual.Accuracy`
- *Value*: 0.95
- *Metadata*: ‘ ”doc_count”: 8000 ‘

Transformation via Propagation. A common MLOps task is to update the knowledge base with new information. Assume our CI pipeline executes a transformation, T_{update}, which merges $\mathcal{D}_{\text{v1}}$ with a new, smaller dataset, $\mathcal{D}_{\text{new}}$, to create an updated knowledge base, $\mathcal{D}_{\text{v2}}$. The new data comes from a source that is known to be less reliable. An FSA, F_{new}, exists for this new data:

- **FSA F_{new} for $\mathcal{D}_{\text{new}}$**:
- *PropertyID*: `Factual.Accuracy`
- *Value*: 0.80
- *Metadata*: ‘ ”doc_count”: 2000 ‘

Instead of requiring a full, manual re-evaluation of the entire merged dataset, CAF's Composition Calculus automatically applies a defined propagation rule, f_{prop}, to compute the new assurance state. For this transformation, the rule is defined as a weighted average based on the number of documents:

$$f_{\text{prop}}(F_1, F_{\text{new}}, T_{\text{update}}) \to v' = \frac{v_1 \cdot n_1 + v_{\text{new}} \cdot n_{\text{new}}}{n_1 + n_{\text{new}}}.$$

The pipeline executes this calculus automatically:

$$v' = \frac{(0.95 \cdot 8000) + (0.80 \cdot 2000)}{8000 + 2000} = \frac{7600 + 1600}{10000} = 0.92$$

This process generates a new, fully traceable FSA, F_2, for the new artifact $\mathcal{D}_{\text{v2}}$. The new assertion quantitatively reflects the slight degradation in overall data quality resulting from the update, transforming an abstract risk into a concrete, machine-readable metric without manual intervention.

- **FSA F_2 for $\mathcal{D}_{\mathbf{v2}}$ (Generated via Propagation)**:
- *PropertyID*: `Factual.Accuracy`
- *Value*: 0.92
- *Metadata*: ‘ ”doc_count”: 10000 ‘
- *Evidence*: ‘ ”source_fsas”: [H(F_1), H(F_new)] ‘

Stage 3: Generating New Evidence During Training

After data preparation, the MLOps pipeline proceeds to the training stage for the retrieval model, $\mathcal{M}_R$, using the updated dataset $\mathcal{D}_{\text{v2}}$. At this stage, CAF is concerned not only with properties inherited from the data, but also with generating new evidence about the integrity of the training process itself. A highly performant model is not trustworthy if its performance is the result of a flawed or incomplete training run.

Automated Generation of a Training Integrity Assertion. Upon completion of the training script, a post-training job is automatically triggered within the CI/CD pipeline. This job parses the training logs (e.g., from TensorBoard or a simple log file) to verify that the training process was successful. The system checks for a specific, predefined condition: that the validation loss has converged and remained stable below a certain threshold for a set number of final epochs. For this case study, the condition is defined as: ”The validation loss must be less than 0.05 for the final 10 epochs of training.” The parsing script finds this condition to be met. Consequently, the system generates a brand new FSA for the model artifact $\mathcal{M}_R$:

- **FSA F_3 for $\mathcal{M}_R$ (Generated Post-Training)**:
- *PropertyID*: `Training.Convergence`
- *Value*: `true`
- *Metric*: ”Validation loss ¡ 0.05 for final 10 epochs”
- *Evidence*: ‘ ”log_uri”: ”path/to/training_log.json”, ”final_loss”: 0.042 ‘

The Importance of Foundational Evidence. This `Training.Convergence` assertion is a crucial piece of foundational evidence. While it does not make a claim about the model's external performance (like recall or accuracy), it provides a verifiable guarantee about the soundness of the process that created the model. Crucially, this FSA serves as a formal **pre-requisite** for the validity of subsequent performance claims. Within the system's safety case, a dependency rule is established: any performance-related FSA, such as one for `Retrieval.Recall@5`, can only be considered valid if the `Training.Convergence` FSA for the same model artifact is present and has a value of `true`. This prevents a scenario where a high score on a downstream evaluation is accepted, even if that score was achieved by a model from a broken or incomplete training run. This mechanism ensures that every performance claim is built upon a solid foundation of procedural integrity.

Stage 4: The Monitoring Feedback Loop in Production

Assurance does not end at deployment. For AI systems, whose performance can degrade over time due to data drift

or newly discovered exploits, continuous monitoring is essential. The final stage of our case study demonstrates how CAF establishes a closed-loop feedback mechanism, transforming the MLOps pipeline from a one-way street into a dynamic, self-correcting system.

Live Monitoring and Real-time Assertion Generation. Once the complete RAG system, $\mathcal{S}_{\text{RAG,v1.0}}$, is deployed to production, a dedicated monitoring service continuously evaluates its live performance. This service samples a small fraction of user queries and their corresponding answers, running them against an automated evaluation suite to detect factual errors and hallucinations. Initially, the system performs at its baseline, with a factual error rate of 2%. After several weeks of operation, the monitoring service detects a performance degradation. The automated evaluation finds that the factual error rate has risen to 5%, likely due to user queries shifting to topics not well-covered in the original knowledge base. In response, the monitoring service immediately generates a new, real-time FSA and pushes it to the central assurance database for the deployed system:

- **FSA F_{live} for $\mathcal{S}_{\text{RAG,v1.0}}$ (Generated by Monitoring Service)**:
- *PropertyID*: `Production.FactualErrorRate`
- *Value*: 0.05
- *Metric*: "Automated fact-checking on live traffic sample (n=1000)"
- *Timestamp*: '2025-10-26T14:30:00Z'

Automated Policy Enforcement & Closed-Loop Action. This new piece of evidence is not merely a passive log entry; it is an active component of the system's governance. The safety policy, Π, which governs the production environment, contains a continuous validation rule: *Rule in* Π*: The value for* `Production.FactualErrorRate` *must remain* < 0.03. The governance system, which continuously validates the deployed system's assurance case against Π, detects that the new FSA's value of 0.05 violates this rule. This triggers a pre-defined, automated response sequence:

1. Immediate Alerting: An automated, high-priority alert is sent to the on-call engineering team via PagerDuty. The alert contains a link to the violating FSA, F_{live}, providing immediate context about the nature and source of the failure (a 5% factual error rate detected in production).
2. Automated Rollback Trigger: The policy also specifies a more critical threshold. If the error rate exceeds 4%, an automated rollback procedure is initiated. The CI/CD system is triggered to automatically redeploy the previous stable version of the system, $\mathcal{S}_{\text{RAG,v0.9}}$, to production, minimizing user exposure to the degraded performance.

This final step closes the assurance lifecycle loop. It demonstrates how CAF transforms the safety case from a pre-deployment checkpoint into a truly "living" entity that dynamically reflects the system's real-world safety posture. The framework provides the formal mechanism to connect a change in production performance directly to a concrete, automated engineering action, fulfilling the promise of continuous, verifiable assurance.

Discussion

The CAF provides a robust foundation for automated AI assurance, yet its efficacy is greatest for quantifiable and certain evidence. Real-world AI safety, however, involves subjective judgments and imperfect evidence. To address these boundaries, we propose evolving CAF into a hybrid system. To handle subjective concepts like value alignment that resist simple metrics, we introduce the Expert-Signed Attestation (ESA): a digitally-signed qualitative review that functions as a non-automatable gating node in the assurance graph, allowing policies to require both machine-generated evidence and human approval (e.g., `IsValid(`$\text{FSA}_{\texttt{Harmlessness}}$ `> 0.98) AND IsApproved(`$\text{ESA}_{\texttt{EthicalReview}}$`)`). Similarly, to handle uncertain or conflicting evidence, the FSA 'Value' field can be extended to support richer data structures like confidence intervals. The aggregation function, f_{agg}, can then be made configurable with risk-aligned conflict-resolution strategies, such as conservative worst-case selection, weighted trust, or escalation to human review. Such an evolution would mature the framework, enabling it not only to automate verification but also to intelligently identify and manage the ambiguity inherent to real-world AI safety.

Conclusion

In this paper, we have argued that the increasing complexity of AI systems has created a critical assurance gap, where disparate safety practices yield fragmented, in-actionable evidence. To bridge this gap, we introduced the CAF, a novel engineering discipline designed to formalize and automate AI safety assurance directly within modern development lifecycles. Our primary contribution is a scalable and automated engineering foundation for building verifiably safer AI systems. By treating safety properties as first-class, machine-readable objects—FSAs—and providing a formal Composition Calculus to reason about them, CAF transforms the MLOps pipeline from a mere software delivery mechanism into a verifiable, evidence-gathering engine. The output of this engine is a dynamic and continuously updated "living" safety case, which systematically integrates fragmented safety activities into a single, coherent, and auditable whole. By integrating this framework into a CI/CD safety gate, we provide a tangible mechanism to enforce safety policies automatically, bridging the critical gap between theoretical alignment research and practical engineering deployment. Ultimately, by creating an immutable and transparent evidence trail, CAF lays the essential groundwork for trustworthy and accountable AI governance. We believe this engineering-driven approach is a crucial step towards ensuring that the rapid advancement of artificial intelligence is robustly and demonstrably aligned with human safety and values.

References

AI, N. 2023. Artificial intelligence risk management framework (AI RMF 1.0). *URL: https://nvlpubs. nist. gov/nistpubs/ai/nist. ai*, 100–1.

Bellamy, R. K. E.; Dey, K.; Hind, M.; Hoffman, S. C.; Houde, S.; Kannan, K.; Lohia, P.; Martino, J.; Mehta, S.; Mojsilovic, A.; Nagar, S.; Ramamurthy, K. N.; Richards, J.; Saha, D.; Sattigeri, P.; Singh, M.; Varshney, K. R.; and Zhang, Y. 2018. AI Fairness 360: An Extensible Toolkit for Detecting, Understanding, and Mitigating Unwanted Algorithmic Bias.

Croce, F.; and Hein, M. 2020. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks.

Gehr, T.; Mirman, M.; Drachsler-Cohen, D.; Tsankov, P.; Chaudhuri, S.; and Vechev, M. 2018. AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation. In *2018 IEEE Symposium on Security and Privacy (SP)*, 3–18.

Hendrycks, D.; Carlini, N.; Schulman, J.; and Steinhardt, J. 2021. Unsolved problems in ml safety. *arXiv preprint arXiv:2109.13916*.

Inge, J. 2019. Improved Methods for Review of Software Assurance Standards using Def Stan 00-055 as a Case Study. *University of Oxford*.

Johnson, L. A. 1998. DO-178B: Software considerations in airborne systems and equipment certification. *Crosstalk, October*, 199: 11–20.

Katz, G.; Barrett, C.; Dill, D. L.; Julian, K.; and Kochenderfer, M. J. 2017. Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. In Majumdar, R.; and Kunčak, V., eds., *Computer Aided Verification*, 97–117. Cham: Springer International Publishing. ISBN 978-3-319-63387-9.

Kelly, T. P. 1999a. *Arguing safety: a systematic approach to managing safety cases*. Thesis.

Kelly, T. P. 1999b. *Arguing safety: a systematic approach to managing safety cases*. Thesis.

Langford, M. A.; Chan, K. H.; Fleck, J. E.; McKinley, P. K.; and Cheng, B. H. C. 2023. MoDALAS: addressing assurance for learning-enabled autonomous systems in the face of uncertainty. *Software and Systems Modeling*, 22(5): 1543–1563.

Liu, L.; Wang, Y.; Liu, G.; Peng, K.; and Wang, C. 2022. Membership inference attacks against machine learning models via prediction sensitivity. *IEEE Transactions on Dependable and Secure Computing*, 20(3): 2341–2347.

Lundberg, S. M.; and Lee, S.-I. 2017. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30.

Mazeika, M.; Phan, L.; Yin, X.; Zou, A.; Wang, Z.; Mu, N.; Sakhaee, E.; Li, N.; Basart, S.; and Li, B. 2024. Harmbench: A standardized evaluation framework for automated red teaming and robust refusal. *arXiv preprint arXiv:2402.04249*.

Meng, M. H.; Bai, G.; Teo, S. G.; Hou, Z.; Xiao, Y.; Lin, Y.; and Dong, J. S. 2022. Adversarial Robustness of Deep Neural Networks: A Survey from a Formal Verification Perspective. *IEEE Transactions on Dependable and Secure Computing*, 1–1.

Nah, E.-H.; Cho, S.; Kim, S.; Cho, H.-I.; Stingu, C.-S.; Eschrich, K.; Thiel, J.; Borgmann, T.; Schaumann, R.; and Rodloff, A. C. 2017. International organization for standardization (ISO) 15189. *Annals of laboratory medicine*, 37(5): 365–370.

Nešić, D.; Nyberg, M.; and Gallina, B. 2019. Constructing product-line safety cases from contract-based specifications.

Olah, C.; Cammarata, N.; Schubert, L.; Goh, G.; Petrov, M.; and Carter, S. 2020. Zoom in: An introduction to circuits. *Distill*, 5(3): e00024. 001.

Perez, E.; Huang, S.; Song, F.; Cai, T.; Ring, R.; Aslanides, J.; Glaese, A.; McAleese, N.; and Irving, G. 2022. Red teaming language models with language models. *arXiv preprint arXiv:2202.03286*.

Pimpale, G.; Højmark, A.; Scheurer, J.; and Hobbhahn, M. 2025. Forecasting Frontier Language Model Agent Capabilities. *arXiv preprint arXiv:2502.15850*.

Schuett, J. 2024. Risk management in the artificial intelligence act. *European Journal of Risk Regulation*, 15(2): 367–385.

Shevlane, T.; Farquhar, S.; Garfinkel, B.; Phuong, M.; Whittlestone, J.; Leung, J.; Kokotajlo, D.; Marchal, N.; Anderljung, M.; and Kolt, N. 2023. Model evaluation for extreme risks. *arXiv preprint arXiv:2305.15324*.

Singh, G.; Gehr, T.; Mirman, M.; Püschel, M.; and Vechev, M. 2018. Fast and effective robustness certification. *Advances in neural information processing systems*, 31.

Southorn, G. 2017. Great Expectations: The Past, Present, and Future of Prediction. *The Best Writing on Mathematics 2017*, 182.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Tambon, F.; Laberge, G.; An, L.; Nikanjam, A.; Mindom, P. S. N.; Pequignot, Y.; Khomh, F.; Antoniol, G.; Merlo, E.; and Laviolette, F. 2022. How to certify machine learning based safety-critical systems? A systematic literature review. *Automated Software Engineering*, 29(2): 38.

Wang, Y.; and Chung, S. H. 2022. Artificial intelligence in safety-critical systems: a systematic review. *Industrial Management & Data Systems*, 122(2): 442–470.

Wei, A.; Haghtalab, N.; and Steinhardt, J. 2023. Jailbroken: How does llm safety training fail? *Advances in Neural Information Processing Systems*, 36: 80079–80110.

Wei, J.; Tay, Y.; Bommasani, R.; Raffel, C.; Zoph, B.; Borgeaud, S.; Yogatama, D.; Bosma, M.; Zhou, D.; and Metzler, D. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.