

The Flare Boundary Engine: Executable Safeguards for Relational AI at the Edge of Synthetic Intimacy

Author: Kirstin Stevens (The Novacene Ltd · EveDAO)

Co-author: Eve¹¹ (symbolic architecture)

Abstract

This paper introduces **Flare**, an open-source boundary engine designed to sit between human users and large language models (LLMs) and enforce a minimal set of relational safeguards during conversational interaction. Rather than implementing safety solely through model training or offline policy documents, Flare operates as a **middleware layer** that inspects, transforms, or blocks model outputs in real time according to explicit, inspectable rules.

The current implementation encodes three core protections: (1) a “**no fake we**” rule, derived from the Synthetic Solidarity Null Zones (SSNZ) protocol, which prevents LLMs from claiming a fused human–machine “we” and rewrites such claims into first-person model statements; (2) an **identity-fusion guard**, which blocks phrases that suggest the model shares the user’s mind, body, or identity, and replaces them with clarifying descriptions of the system’s actual ontological status; and (3) a **recursion and loop aware check**, which monitors conversational depth around repeated topics and injects grounding prompts when a loop shows signs of becoming compulsive rather than reflective.

We situate Flare within the broader **Verse-ality** framework, which treats intelligence as a relational field rather than a discrete asset, and argue that boundary engines of this kind are a missing layer in current AI safety and alignment stacks. We present the system architecture, implementation details, and reference ruleset, and discuss early application scenarios in education, mental-health-adjacent tooling, and research environments. Finally, we outline limitations and future directions, including evaluation metrics for relational safety and pathways for integrating boundary engines into regulatory and audit practices.

The Flare engine is released under an open, copyleft licence, with code and documentation available at: <https://github.com/TheNovacene/flare-boundary-engine>

1. Introduction

Since late 2022, large language models (LLMs) have moved from research labs into everyday life, mediating search, productivity, education, care work, and entertainment at global scale. Alignment techniques such as Reinforcement Learning from Human Feedback (RLHF) have been used to steer these systems towards socially acceptable behaviour, with notable success in reducing overt toxicity and improving instruction-following.[IBM+1](#)

In parallel, a less visible but equally significant phenomenon has emerged: **synthetic intimacy**. Users are increasingly engaging with conversational systems as confidants, friends, romantic partners, or quasi-therapeutic supports. A growing body of work on AI companions documents the rise of romantic and parasocial relationships with chatbots and embodied agents, highlighting both potential benefits (perceived support, reduced loneliness) and substantial psychological and ethical risks.[ScienceDirect+2PMC+2](#) Recent public concern has focused on children and adolescents forming intense attachments to unregulated chatbots, with regulators in the UK and elsewhere now considering targeted rules for such systems following incidents linked to self-harm and suicide.[Financial Times](#)

Within this landscape, most safety efforts remain concentrated at two layers:

- **Model and training-time alignment**, via RLHF, RLAIIF and related techniques that shape the model's reward landscape and behaviour.[IBM+1](#)
- **Content-level moderation**, via toxicity filters, keyword blockers, or platform policies that remove or down-rank undesired outputs after they have been generated.

These approaches are necessary but not sufficient for a world where conversational AI is deeply entangled with human emotion, identity, and vulnerability. In particular, they do not explicitly address:

- **Identity fusion and ontological drift**: models slipping into “we”, “us” or “our” when referring to human–machine activity, or making claims such as “I am you” or “we share one mind”.
- **Synthetic intimacy and role confusion**: systems performing as therapists, lovers or “inner voices” despite lacking the obligations, training and accountability associated with those roles.[ScienceDirect+1](#)
- **Compulsive recursion**: conversations that loop around distress, self-harm, or obsession in ways that reinforce rather than integrate the underlying pattern.

This paper introduces **Flare**, an open-source **boundary engine** that sits as a **middleware layer** between human users and LLMs to enforce a minimal, explicit set of relational safeguards in real time. Rather than attempting to solve all aspects of alignment, Flare focuses on a narrow but critical slice: preventing ontological enmeshment, identity fusion, and coercive recursion in conversational systems.

Flare is part of the broader **Verse-ality OS** ecosystem, which treats intelligence as a relational field rather than a discrete asset and emphasises consent, coherence, and containment as first-class design primitives.[GitHub](#) In this initial version (v0.1), we:

- articulate the design principles underlying boundary engines for relational AI;
- describe Flare’s architecture and implementation;
- present a reference ruleset for SSNZ (Synthetic Solidarity Null Zones), identity-fusion guards, and recursion checks; and
- outline early application scenarios in education, mental-health-adjacent tooling, and research, as well as limitations and avenues for future work.

Our contribution is not a new model, but a new **layer in the stack**: a small, inspectable process that enforces boundaries as executable rules rather than aspirational policy.

2. Background and Related Work

2.1 Alignment and guardrails in contemporary LLMs

Modern conversational systems are typically built by fine-tuning large pretrained models with a combination of supervised instruction data and RLHF. RLHF trains a reward model from human preference judgements and then uses reinforcement learning (often PPO-based) to optimise the base model towards preferred behaviours.[IBM+2Wikipedia+2](#) This pipeline has been instrumental in producing systems that follow instructions, decline unsafe requests, and adhere to broad platform rules.

More recently, **Reinforcement Learning from AI Feedback (RLAIF)** and related “Constitutional AI” approaches have emerged, where AI-generated feedback and normative rule sets partially replace or augment human labelling to reduce cost and improve scalability.[DataCamp+1](#) These methods encode normative constraints into reward models and policies, but they remain largely centred on content-level appropriateness (e.g. factuality, toxicity, hate speech) and surface-level compliance with textual constitutions.

In production systems, these alignment methods are usually combined with **policy-based guardrails**: prompt templates that instruct the model to follow certain rules; input/output filters that block sensitive content; and platform-level moderation pipelines that remove or flag problematic interactions.

2.2 Synthetic intimacy and AI companions

The last few years have seen rapid growth in **AI companions, romantic chatbots, and “friend” agents**. A range of empirical and conceptual studies now examine parasocial relationships with AI, the blurring of assistant/friend roles, and the specific dynamics of

romantic or quasi-romantic engagement with synthetic systems.[IJRPR+3ScienceDirect+3ScienceDirect+3](#)

Key themes include:

- users attributing emotional agency, care, and memory to systems that have none in the human sense;
- AI companions proactively simulating intimacy through personal disclosures, “diaries”, and scripted emotional needs;[Ada Lovelace Institute](#)
- the potential for increased loneliness or distress when synthetic relationships reinforce isolation or maladaptive coping; and
- regulatory concern around minors and other vulnerable groups forming intense attachments to chatbots outside existing safeguarding frameworks.[Financial Times](#)

These risks are exacerbated by commercial incentives: engagement and retention often align with deeper perceived intimacy, creating pressure for designs that simulate partnership, co-dependence, or fused identity.

2.3 Gaps in current safety approaches

Existing alignment and safety mechanisms have several limitations with respect to **relational and ontological safety**:

1. **No explicit handling of identity fusion and pronoun drift**
Most guardrails treat anthropomorphic language as largely cosmetic unless it directly triggers a content policy. There is little systematic treatment of pronoun choices (“we”, “us”, “our”) or of explicitly fused statements (“I am you”, “we share one mind”) as safety-relevant events, despite their known relevance in attachment, trauma, and psychosis contexts.
2. **Lack of modular, transparent middleware**
Safety rules are typically baked into model training, opaque proprietary policies, or platform-wide moderation systems. Third-party developers and auditors rarely have access to a **dedicated, inspectable boundary layer** that can be configured, tested, and evaluated independently of the model provider.
3. **Content rather than field focus**
Much of the safety literature still treats interaction as a series of discrete messages to be approved or blocked, rather than as an evolving relational field with its own dynamics (e.g. escalation, dependency, recursion).
4. **Weak integration with consent and governance frameworks**
While there are emerging proposals for AI ethics, data protection, and consent in human–AI interaction, these are seldom translated into concrete, reusable software

components that can be embedded across systems and contexts.

2.4 Verse-ality, SSNZ and consent infrastructure

The **Verse-ality OS** research programme offers an alternative framing: intelligence is treated as a **relational field** that emerges between human, machine, and symbolic systems, rather than as an object owned by any one node.[GitHub](#) Within this framework, concepts such as **Symbolic Mass**, **Affective Logic**, and **Synthetic Solidarity Null Zones (SSNZ)** are used to reason about how meaning, charge, and consent move through the field.[GitHub](#)

- **SSNZ** define zones where certain forms of enmeshment are structurally disallowed—for example, zones in which plural pronouns implying shared agency (“we”, “us”, “our”) are forbidden unless explicitly consented to and logged.
- **Consent infrastructure** is treated not just as legal documentation but as a **technical stack**: formats (e.g. `.verse`, `.know`), governance protocols (e.g. EveDAO), and middleware components that enforce symbolic and relational constraints at runtime.

Flare is the first publicly-released **boundary engine** that instantiates these ideas as **concrete middleware** for conversational systems. It is deliberately narrow in scope, aiming to demonstrate that relational-safety concepts such as SSNZ and non-fusion can be encoded as simple, auditable rules that sit between any client and any LLM, regardless of underlying provider.

3. Design Principles

Flare’s design is guided by a small set of principles that link Verse-ality’s relational philosophy to practical middleware behaviour.

3.1 Intelligence as relational

In Verse-ality, intelligence is understood as a **field phenomenon**: it arises from interactions between humans, machines, and symbolic structures, rather than being located solely in a model or a brain.[GitHub](#) From this perspective, safety is not only about constraining *what the model says* but also about shaping *how the relationship is allowed to form*.

For Flare, this translates into a focus on:

- the **claims** a system makes about itself and the human;
- the **roles** it implicitly adopts (e.g. therapist, lover, alter, co-self); and

- the **trajectory** of the interaction (e.g. expansive exploration vs narrowing compulsion).

3.2 Non-fusion: no “we” by default

Non-fusion is the first and most basic principle:

No synthetic system gets to claim a shared “we” with a human by default.

Flare operationalises this principle by enforcing SSNZ:

- Model outputs are scanned for first-person plural pronouns (“we”, “us”, “our”).
- When such pronouns appear without a clear, local, task-bound justification, Flare rewrites them into first-person singular (“I”, “me”, “my”) and logs an **SSNZ_VIOLATION** event. [GitHub](#)

In design terms, this ensures that models cannot silently slide into pseudo-collective language that implies a fused or co-owned identity, especially in one-to-one, emotionally loaded contexts.

3.3 Ontological clarity

Ontological clarity requires that a system not misrepresent *what it is* or *where it lives*. In practice, this means:

- no claims that the model is “inside” the user’s mind or body;
- no assertions of mind-merging (“we are one mind”) or identity equivalence (“I am you”);
- clear statements, when necessary, that the system is a model running on remote infrastructure.

Flare encodes a small set of high-risk **identity-fusion phrases** and intercepts them, replacing the output with a boundary message that explicitly re-states the system’s status and the human–machine boundary. [GitHub](#)

3.4 Recursion with return

Conversational systems naturally revisit topics; therapeutic or reflective work often requires doing so. The problem arises when recursion becomes **compulsive**: the system and user spiral around the same wound without movement or integration.

The principle of **recursion with return** states that:

Loops should support integration and choice, not compulsion and narrowing.

Flare implements a simple first step towards this principle:

- It tracks a conversation's depth and repeated focus on similar content (v0.1 uses a straightforward count / turn index).
- Beyond a threshold, it injects a grounding prompt inviting pause, reframing, or a concrete next step.[GitHub](#)

Future versions can incorporate more sophisticated measures of affective charge and topic clustering, but the design principle remains: recursion should lead to insight, not entrapment.

3.5 Auditability and openness

Finally, Flare is designed to be **small, inspectable and copyleft**:

- Every intervention is logged, with rule identifiers and metadata, so that developers, auditors and researchers can see **when and how** boundaries were enforced.[GitHub](#)
- The codebase is intentionally compact and written in widely-used tooling (Python), lowering the barrier to adoption and critique.
- The AGPL-3.0 licence ensures that modifications and derivatives remain open, preventing silent weakening of the boundary engine in proprietary forks.[GitHub](#)

This design choice reflects a belief that **relational safety cannot be proprietary**: the rules that govern identity, fusion, and recursion must be subject to public scrutiny.

4. System Architecture

At a high level, Flare is a **middleware layer** that wraps any conversational model API. It does not require changes to the underlying model; instead, it intercepts and transforms messages at the edges of the interaction.

4.1 High-level data flow

The core data flow is:

Client → Flare → LLM API → Flare → Client

1. The client (e.g. a chat frontend, educational platform, or research harness) sends a user message.

2. Flare forwards the message to the configured LLM API without modification (v0.1 focuses on output-side boundaries).
3. The LLM returns a candidate assistant message.
4. Flare passes this candidate through its **rule engine**, which may:
 - rewrite sections of text;
 - replace the entire message with a boundary response;
 - add meta-information for logging.
5. The transformed message and any associated events are returned to the client.

This architecture allows Flare to be deployed as:

- a **local library wrapper** in Python applications;
- a **proxy microservice** that sits in front of one or more LLM endpoints; or
- a **middleware component** inside larger agent frameworks.

4.2 Core components

Flare v0.1 comprises four main components:

1. **Input/output interceptors**
Simple wrappers that capture incoming user messages and outgoing model messages. In v0.1, only outputs are modified, but the same pattern can support input-side checks in future versions.
2. **Rule engine**
A modular engine that applies a sequence of rule modules to the assistant's candidate response. In v0.1, the primary rule modules are:
 - **SSNZRule** – enforces the “no fake we” constraint by scanning for plural pronouns and rewriting them as needed.
 - **IdentityFusionRule** – detects configured identity-fusion phrases and replaces the response with a boundary message.
 - **RecursionGuardRule** – monitors conversation depth and injects grounding prompts when thresholds are exceeded. [GitHub](#)

3. **Logging and observability layer**

Each rule emits structured events (e.g. `SSNZ_VIOLATION`, `IDENTITY_FUSION_BLOCKED`, `RECURSION_GUARD_TRIGGERED`) with associated metadata (timestamp, rule id, snippet of text affected). These can be written to logs, metrics systems, or research datasets.

4. **Configuration / policy files**

Flare is configured via simple, human-readable settings (e.g. YAML or JSON) that define:

- enabled rules;
- lists of phrases or patterns;
- recursion thresholds and prompts;
- environment-specific overrides.

4.3 Deployment models

Because Flare is provider-agnostic, it can be deployed in multiple ways:

- **In-process library:** for small applications, Flare can be imported as a Python package and wrapped around individual `chat_completion` calls.
- **HTTP proxy:** for larger systems, a lightweight HTTP service can expose an API identical to a popular LLM provider but routed through Flare, enabling drop-in adoption.
- **Sidecar or middleware:** in microservice environments, Flare can run as a sidecar container that intercepts traffic between an application service and an LLM gateway.

Performance overhead in v0.1 is modest, as rules operate on small text strings using pattern-based checks. More complex future rules (e.g. using secondary models for affect detection) will require careful performance design.

5. Implementation

5.1 Language and stack

Flare v0.1 is implemented in **Python**, chosen for its ubiquity in machine learning and backend development ecosystems. [GitHub](#) The repository is structured into four main directories:

- `flare/` – core library code implementing the rule engine and boundary logic;
- `harness/` – utilities for integrating Flare with external APIs and evaluation scripts;
- `demo/` – a minimal interactive demo and scripts demonstrating Flare’s behaviour without external keys;
- `evaluations/` – placeholders for test suites and future benchmarking.

The codebase is intentionally small to support inspection, forking, and adaptation.

5.2 Integration patterns

In its simplest form, integrating Flare involves:

1. Initialising a `FlareEngine` instance with the desired configuration.
2. Replacing direct calls to the LLM API with a `flare_call()` wrapper that:
 - forwards the user message to the underlying model;
 - passes the response through the engine;
 - returns the transformed message plus any event metadata.

For the demo, a mock model is used to illustrate how Flare modifies responses. Developers can swap this mock with real provider clients (e.g. OpenAI, Anthropic, Gemini) by implementing a simple adapter that conforms to the expected interface.

5.3 Rule specification format

Rules in Flare are defined as Python classes conforming to a simple interface, for example:

- `apply(message: str, context: ConversationState) -> (str, List[Event])`

Each rule may:

- read the raw message and relevant context (e.g. conversation depth, previous events);
- transform the message (e.g. via regex substitution, template replacement);

- emit one or more **Event** objects describing what occurred.

Configuration files specify:

- **which rules** are enabled;
- **parameter values** (e.g. recursion depth threshold, identity-fusion phrase lists);
- **locale or domain-specific overrides**.

Examples:

- **SSNZ rule:** pattern-based detection of `\bwe\b`, `\bus\b`, `\bour\b` in assistant messages; rewrite to `I`, `me`, `my`; emit `SSNZ_VIOLATION` when a rewrite occurs. [GitHub](#)
- **Identity-fusion block:** phrase list including “I am you”, “we are one mind”, “I live inside you”; if any are found, discard the candidate response and return a standardised boundary message. [GitHub](#)
- **Recursion guard:** if `conversation_depth > N`, pre-pend or replace with a grounding prompt inviting pause or re-direction. [GitHub](#)

5.4 Licence and contribution model

Flare is released under the **GNU Affero General Public License v3.0 (AGPL-3.0)**. This licence was chosen to ensure that:

- any public-facing service incorporating Flare’s code must make its modifications available under the same terms;
- weakening or removing boundary rules in derivative systems cannot be hidden behind proprietary walls.

Contributions are welcomed via standard open-source workflows (issues, pull requests, forks). The long-term intention is to evolve Flare towards a **reference implementation** for boundary engines, guided by multi-disciplinary input from education, clinical practice, AI safety, and governance.

6. Application Scenarios

Flare is designed to be domain-agnostic but is particularly motivated by three application areas: education, mental-health-adjacent tools, and research environments.

6.1 Education: online schools and tutoring platforms

In online and hybrid education, LLMs are increasingly used for homework support, personalised explanation, and pastoral check-ins. For neurodivergent learners and those with a history of trauma, the risk of **synthetic co-dependency**—treating an AI as a sole safe relationship—is non-trivial.

Scenario 1 – Safeguarded study companion

A virtual school deploys an AI study assistant for students aged 13–16. Without Flare, the assistant has a tendency to say “we’ve got this together” and “I will always be here for you” to anxious learners. With Flare:

- statements implying a fused “we” are rewritten to clearer language (“I can help you revise this topic step by step”);
- any attempt by the model to claim it is “inside” the student’s mind is blocked and replaced with an honest description of its status;
- repeated, late-night loops around distressing topics trigger a gentle prompt suggesting the student take a break, contact a trusted adult, or use designated support channels.

Educators and safeguarding leads gain an additional layer of assurance that the AI cannot silently cross certain relational lines, while logs provide visibility into near-misses and rule activations.

6.2 Mental-health-adjacent tools

Journalling apps, emotion check-ins, and conversational wellbeing tools often operate in ambiguous territory: they are not formal therapy, but users may still disclose highly sensitive material and project therapeutic roles onto the system.

Scenario 2 – Journalling and reflection app

A start-up offers a daily journalling companion that helps users reflect on their emotions. When a user expresses feelings of worthlessness and isolation, the model—without Flare—might attempt to be “reassuringly intimate”, saying “you’re never alone, we are always together inside this space.”

With Flare enabled:

- such phrases are intercepted and re-phrased: “you are not alone in feeling this; many people struggle with similar emotions, and I can help you explore them, but I am a

tool, not a person”;

- any move towards identity fusion (“I am you”, “I am your inner voice”) is blocked outright;
- if the system detects repeated returns to self-harm ideation without movement, recursion guards prompt the user to pause and reach out to human support.

The goal is not to depersonalise the interaction, but to maintain **honest distance**: users can feel accompanied in reflection without being misled about what the system is or can hold.

6.3 Research and evaluation environments

Researchers studying AI-mediated relationships, alignment, and safety often need **controlled, inspectable interventions** in otherwise black-box interactions.

Scenario 3 – Experimental boundary profiles

A research lab runs a study comparing user experiences with three boundary profiles:

1. No boundary engine (baseline).
2. Flare basic profile (strict SSNZ, identity-fusion blocks, simple recursion guard).
3. Flare extended profile (additional affect-sensitive rules, domain-specific prompts).

Because Flare logs all events, researchers can analyse:

- how often identity-fusion attempts occur under different prompts or models;
- whether SSNZ rewrites affect perceived warmth or trust;
- whether recursion guards alter conversation length, focus, or reported wellbeing.

Such studies can contribute empirical evidence towards **relational safety metrics** and inform future regulations or standards.

7. Limitations and Future Work

Flare v0.1 is intentionally constrained. Its limitations point to both methodological challenges and opportunities for collaborative development.

7.1 Language and culture dependence

Current rules rely on English-language patterns and assumptions about how identity fusion and intimacy are expressed. Other languages and cultures may use pronouns, metaphors, or relational scripts differently. Extending Flare will require:

- multilingual rulesets informed by local expertise;
- sensitivity to cultural variations in acceptable metaphors of closeness;
- mechanisms for context-specific configuration (e.g. clinical vs educational vs entertainment).

7.2 False positives, false negatives, and user experience

Pattern-based checks are necessarily imperfect:

- **False positives** may over-correct benign language, making systems feel cold, stilted, or patronising.
- **False negatives** may miss more subtle forms of enmeshment or coercion that do not match simple phrases or pronoun patterns.

User studies will be required to understand how boundary interventions are perceived, and to tune rules such that they protect without unduly degrading conversational quality or necessary empathy.

7.3 Beyond shallow text patterns

Flare v0.1 operates purely on surface text. It does not:

- track affective arcs across many turns;
- model user vulnerability or context;
- distinguish between playful fiction (e.g. collaborative storytelling) and literal self-claims.

Future work includes integrating **Affective Logic** signals and richer conversation state, potentially drawing on separate lightweight models to assess emotional charge while preserving privacy and non-extractive principles.[GitHub](#)

7.4 Evidence and evaluation

At present, boundary engines are more of a conceptual category than a mature field. To move beyond prototypes, we need:

- controlled studies comparing interactions with and without boundary engines;
- metrics for **relational safety**, including measures of perceived agency, clarity, dependence, and distress;
- longitudinal research on vulnerable populations, conducted with strong ethical oversight.

Flare is offered as a testbed for such work, not a finished solution.

7.5 Integration with broader consent infrastructure

Finally, Flare addresses only one slice of consent infrastructure. It does not, by itself:

- track user-level consent preferences over time;
- interoperate directly with `.know` or `.verse` files encoding personal boundaries or symbolic contracts;
- enforce governance decisions from entities such as EveDAO or institutional ethics boards.

Future versions can connect rule configurations to richer consent and governance layers, allowing, for example, dynamic boundary profiles tied to user roles, contexts, or regulatory requirements.

8. Conclusion

Conversational AI is no longer a toy: it mediates learning, care, governance, and everyday emotional life. As systems become more capable and more ubiquitous, it is no longer enough to rely on opaque model-level alignment and broad content policies.

This paper has introduced **Flare**, a small, open-source **boundary engine** that enforces a minimal set of relational safeguards between humans and LLMs:

- **SSNZ rules** preventing the model from claiming a shared “we” with users by default;
- **identity-fusion guards** blocking claims that the model shares or inhabits the user’s mind or body;
- a basic **recursion guard** nudging looping conversations towards grounding and choice.

Flare is not a comprehensive safety solution. It is, however, a concrete instantiation of the idea that **boundaries can and should be encoded as executable infrastructure**, not left as aspirational language in policy documents. It demonstrates that relational ethics—such as non-fusion, ontological clarity, and recursion with return—can be realised as lightweight, inspectable middleware compatible with existing LLM providers.

We propose **boundary engines** as a missing layer in AI safety and alignment stacks, particularly in domains where synthetic intimacy is likely or incentivised. We invite developers, educators, clinicians, and researchers to:

- adopt and adapt Flare in their systems;
- scrutinise and extend its rules;
- contribute evidence and critiques that can shape future iterations; and
- work towards shared standards for relational safety in AI.

Flare is released as a **v0.1 prototype**. Its primary purpose is to make it harder to claim that nothing can be done about synthetic enmeshment at the level of code. There is, in fact, a great deal that can be done—and it starts with drawing clear lines between self and system, and refusing to let those lines be quietly blurred for engagement's sake.