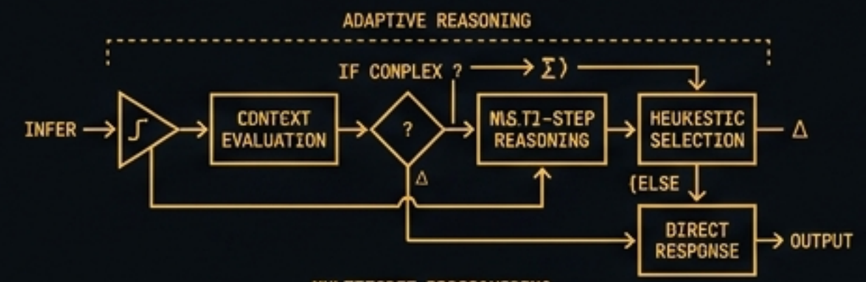
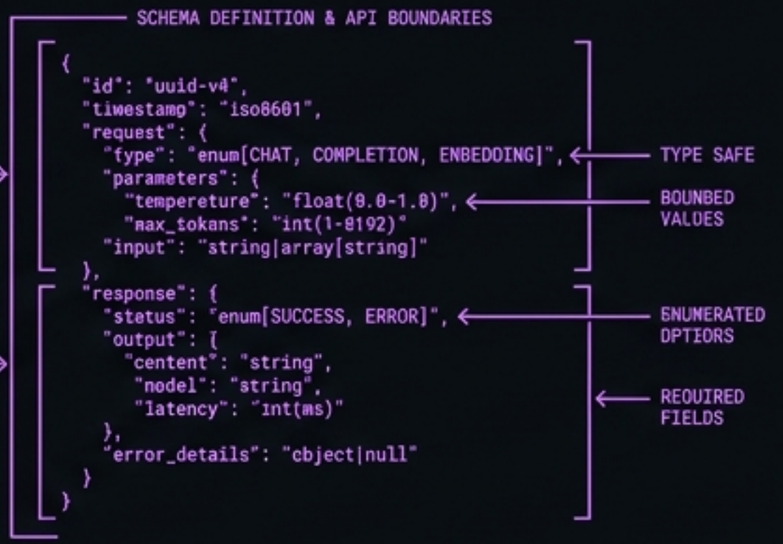
Engineering the Production LLM Interface





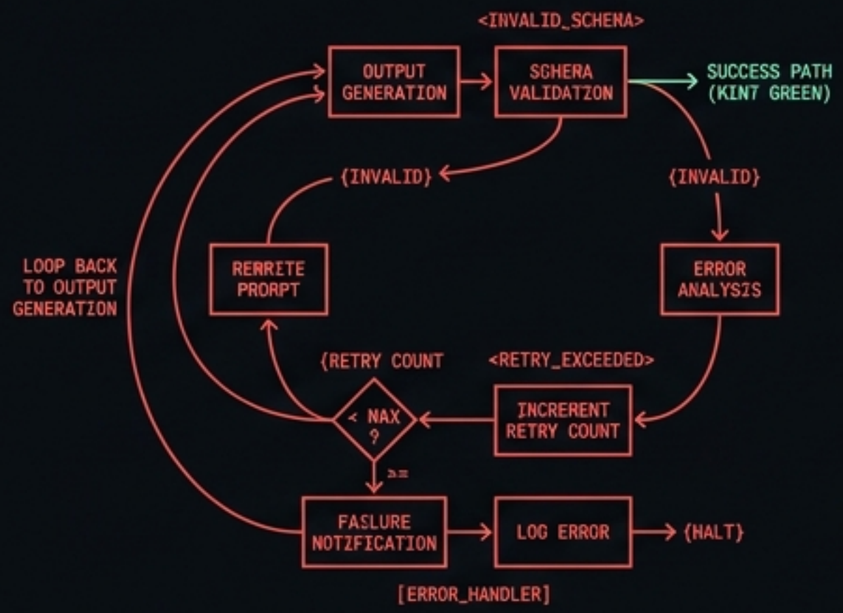
Adaptive Thinking & Heuristics

Strict JSON Invariants

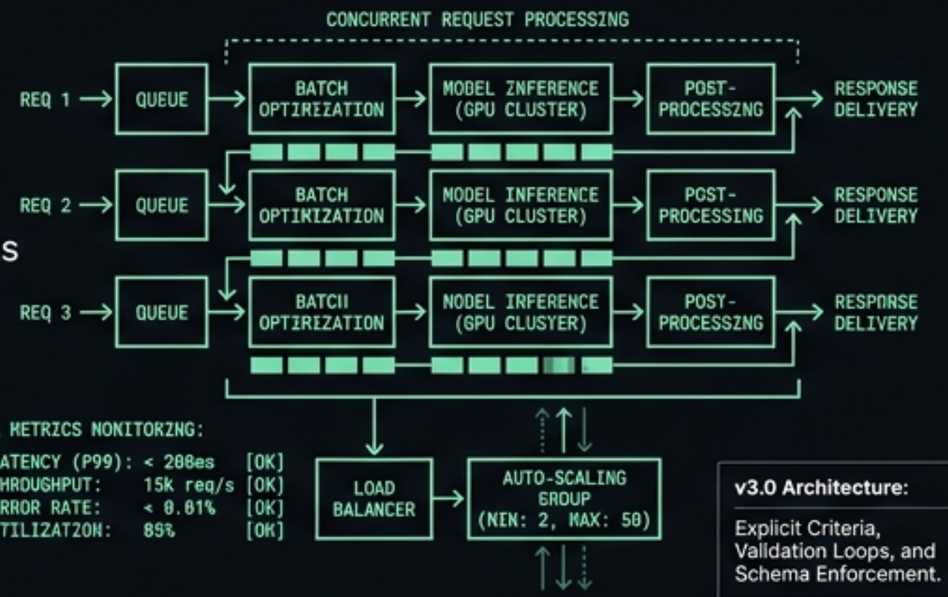


LLM ENGINE

Validation & Error Loops



Batch SLAs & Scale

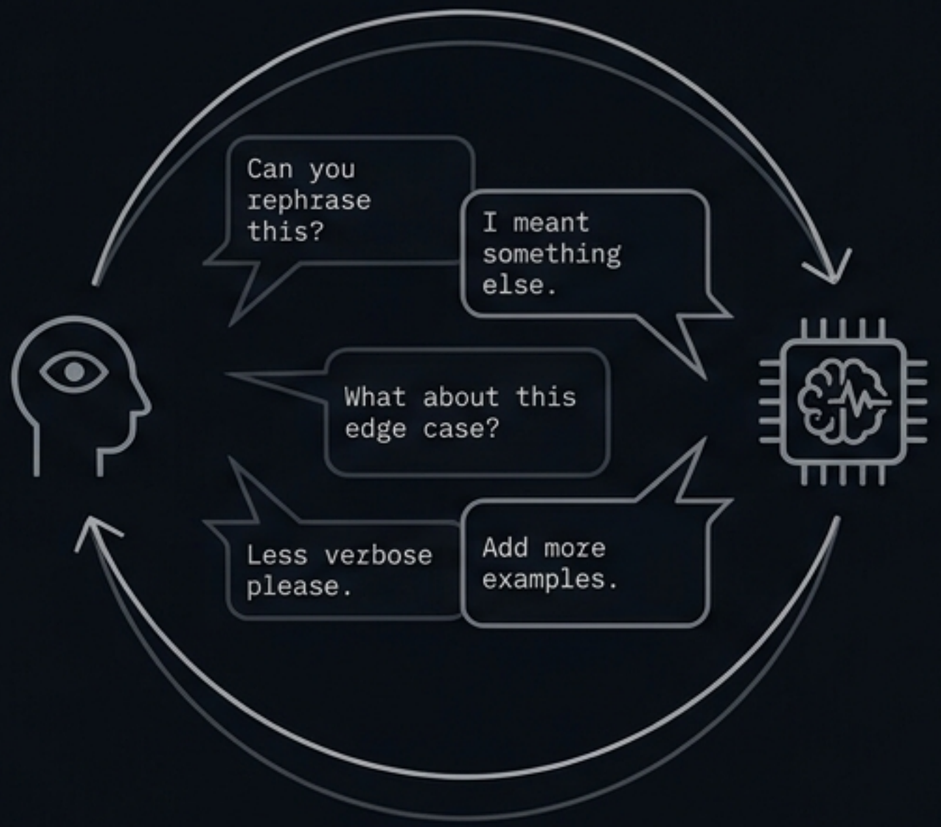


v3.0 Architecture:

Explicit Criteria, Validation Loops, and Schema Enforcement.

The Transition from Chat to Production API Levers

Interactive Chat



Human in the loop spots vague answers and clarifies interactively.

Production Pipeline

Authoring Lever: Explicit Rubrics & Contrast Examples

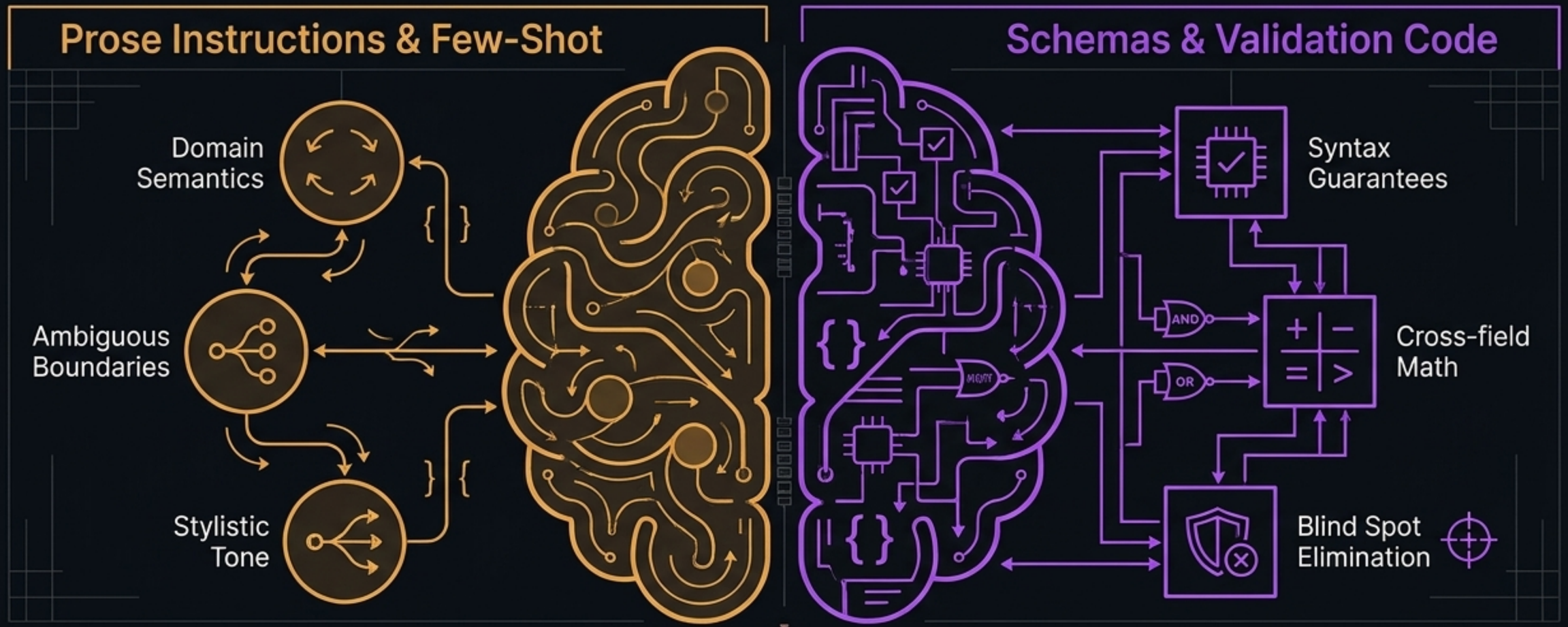
```
{                                                    [ ]
  "rubric": "strict_eval",
  "contrast_examples": [
    "good_response.json", "bad_response.json",
  ] "bad_response.json"
}                                                    { }
```

API Enforcement Lever: Constrained Schemas, tool_choice, & Bounded Retries

```
{                                                    [ ]
  "response_format": {
    "type": "json_schema", "schema": {
      "strict": true                                 ->
    }
  },
  "tool_choice": "required",
  "max_retries": 3
}                                                    { }
```

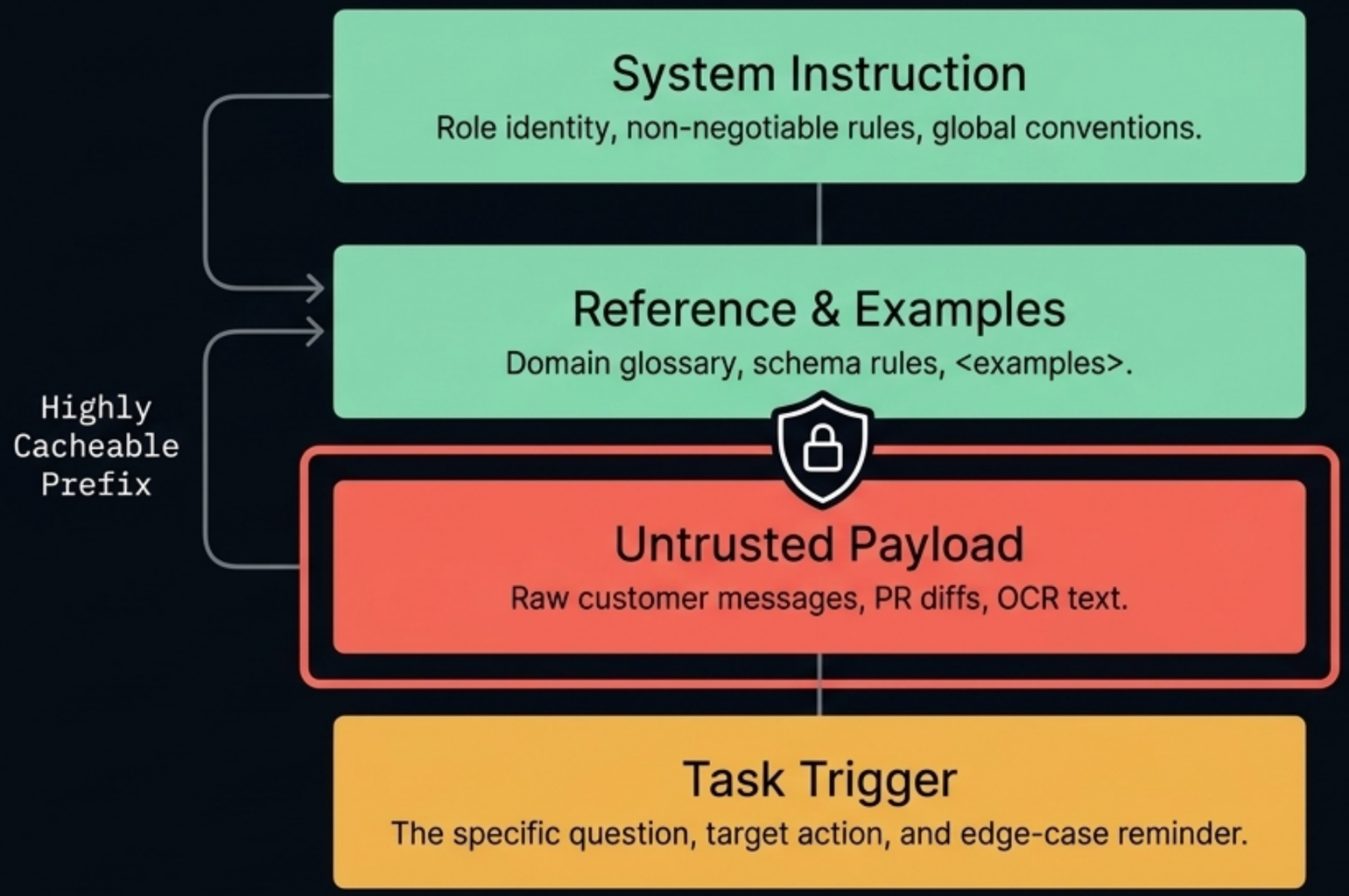
Automated components require programmable boundaries. Zero human intervention for real-time correction.

Prose Steers Judgment; Schemas Enforce Invariants



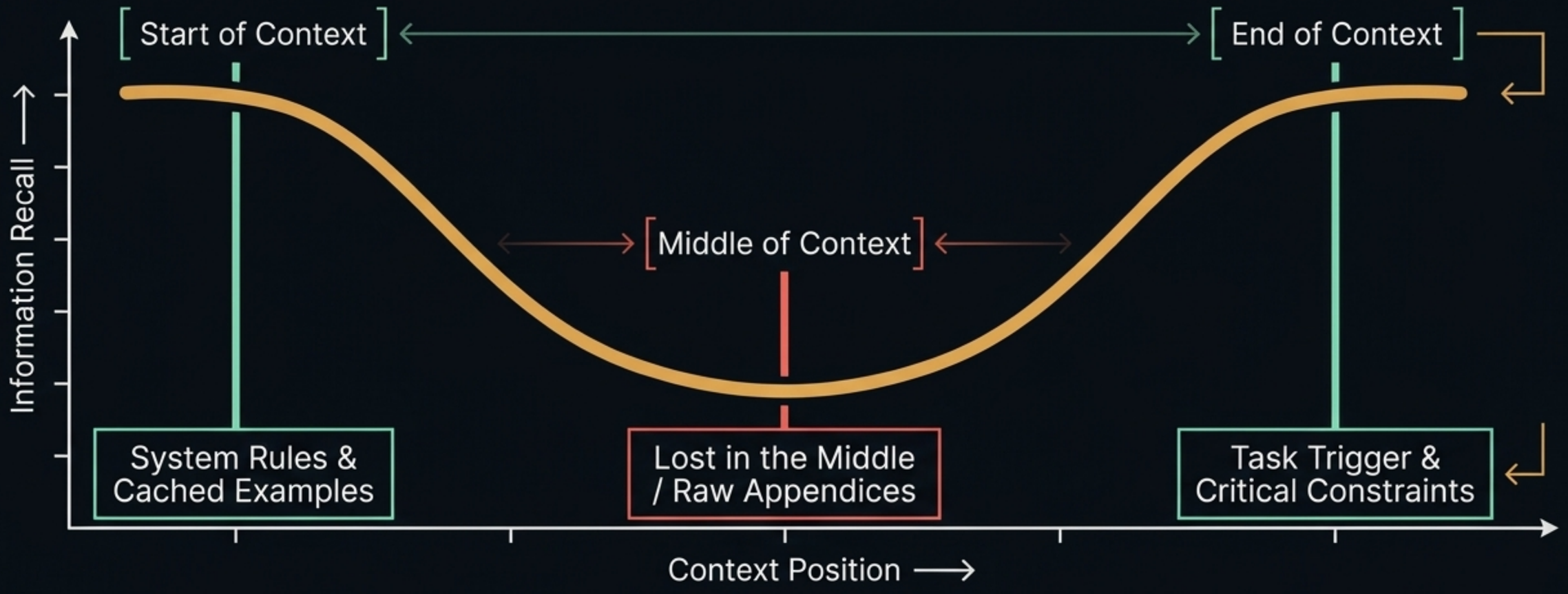
Use prompt instructions to define ambiguous boundaries. Use API schemas and validators to catch cross-field math errors and eliminate blind spots.

Prompt Anatomy and the Architectural Trust Hierarchy



Structural Layer	API Mapping & Config Reference
System Instruction	`Claude: 'system' | Gemini: 'system_instruction'`
Reference & Examples	`Claude: 'messages' (initial turn) | Gemini: 'system_instruction' or 'messages'`
Untrusted Payload	`Claude: 'messages' (user content) | Gemini: 'messages' (user content)`
Task Trigger	`Claude: 'messages' (final user turn) | Gemini: 'messages' (final user turn)`

Respecting the U-Shaped Attention Geometry

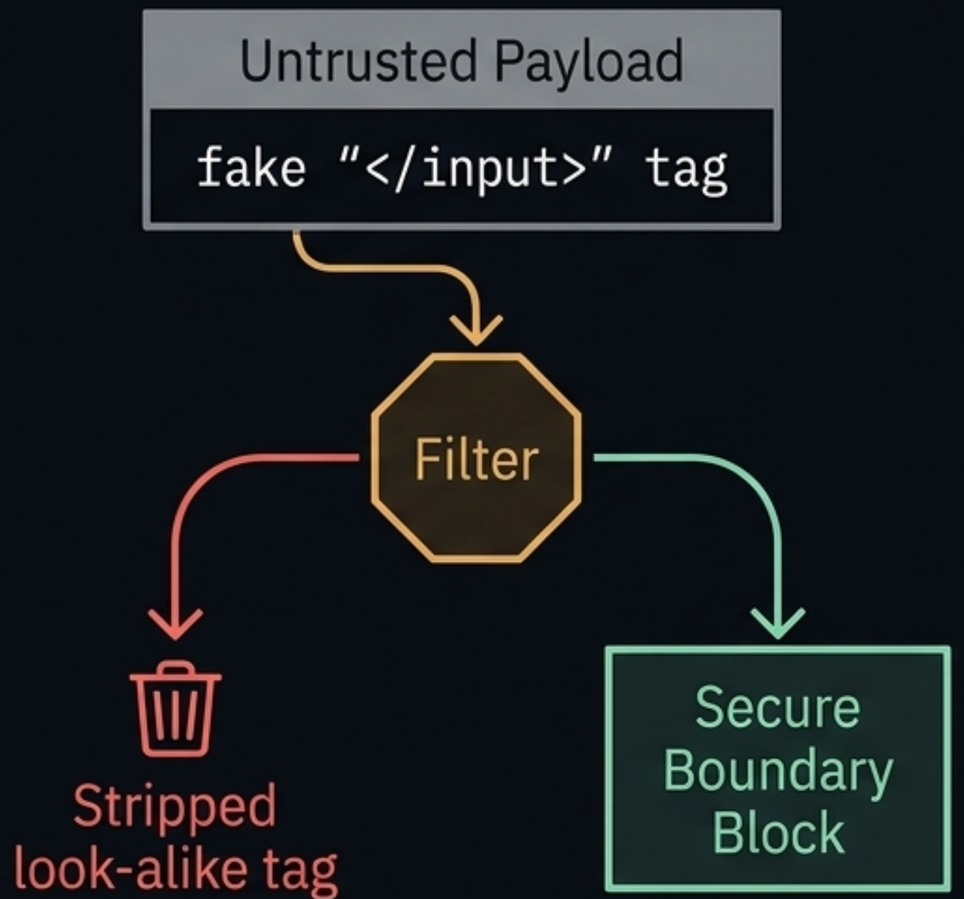


(Liu et al., Lost in the Middle: How Language Models Use Long Contexts)

Rule: **Place large reference documents near the top to form a stable cacheable prefix; state the exact task instructions at the very end.**

Structural Delimiters and XML Tag Discipline

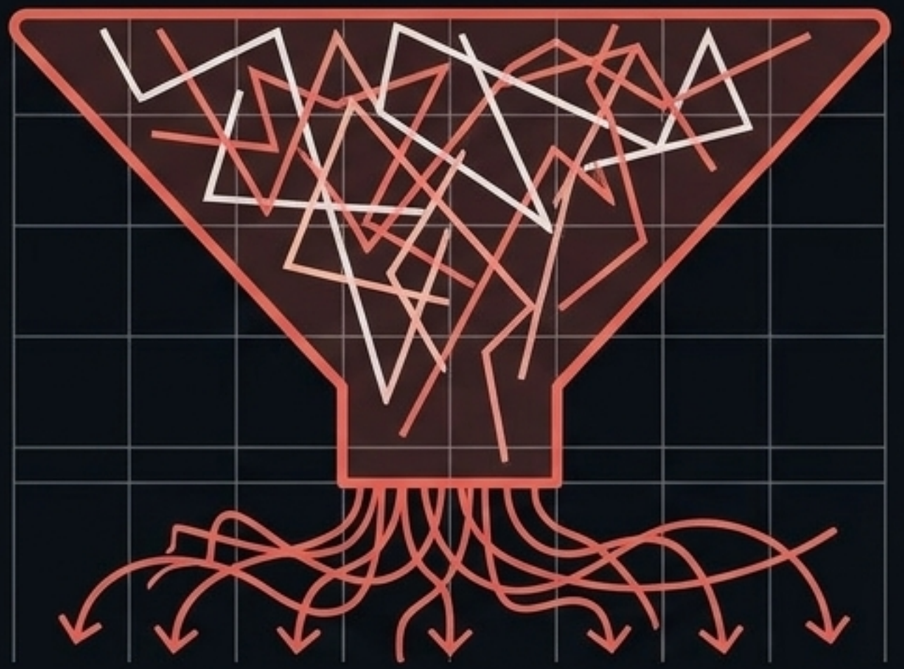
```
ROOT
├── <instructions>
├── <context>
├── <examples>
└── <input>
    └── <documents>
        └── <document index="2">
```



1. One tag per content type.
2. Nest to express hierarchy.
3. Neutralize tag look-alikes.

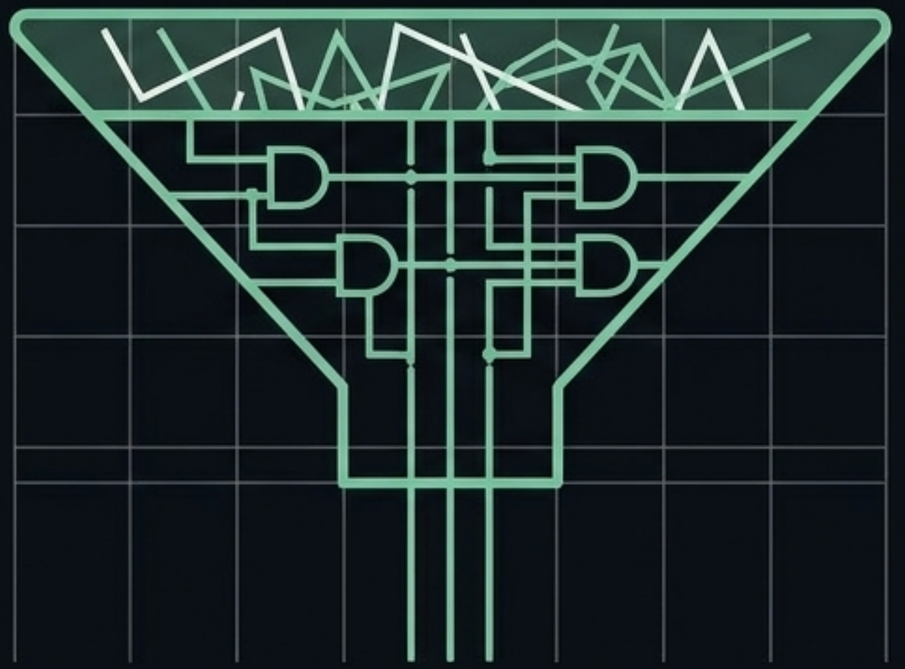
Precision Control: Defeating False Positives

Only report **high-confidence** bugs; be conservative.



Noisy edge cases and stylistic nitpicks slip through, poisoning developer trust.

Report only null dereferences and **auth bypasses**. Skip naming conventions.



Perfect filtering. Only valid defects pass through to the pipeline.

One noisy category poisons every category. Developers will ignore a 92% accurate security reviewer if its maintainability warnings are only 35% accurate. Turn noisy categories off immediately while fixing them offline.

The Severity Rubric Matrix

	Vague Definition (Anti-Pattern)	Explicit Runtime Impact (Production)
Critical	Very bad bug	Exploitable security flaw or guaranteed crash (if is_admin)
Major	Moderate issue	Wrong business result on reachable inputs
Skip	Minor style	Formatting, missing docstrings, established repo idioms

Explicit rubrics anchor severity to tangible code behaviors. Must include one concrete code example per severity level in the prompt contract.

Multishot Architecture and Contrast Pairs

Genuine Positive

```
// Example: Potential Null Pointer Exception
public class DataProcessor {
    public String process(User user) {
        if (user.isActive()) { // Defect: user could be null
            return user.getDetails().toString();
        }
        return "Inactive";
    }
}
```

architectural arrow

```
[
  {
    "finding": "Null Pointer Exception",
    "line": 5,
    "severity": "High",
    "reasoning": "The method accesses 'user.isActive()'
     without a prior null check on 'user', potentially
     causing a NullPointerException if 'user' is null."
  }
]
```

Plausible Near-Miss

```
// Example: Valid Repository Idiom
public class AuthHandler {
    public boolean hasAccess(@Nullable User user) {
        // Correctly handles potential null user
        if (user != null && user.isAdmin()) {
            return true;
        }
        return false;
    }
}
```

architectural arrow

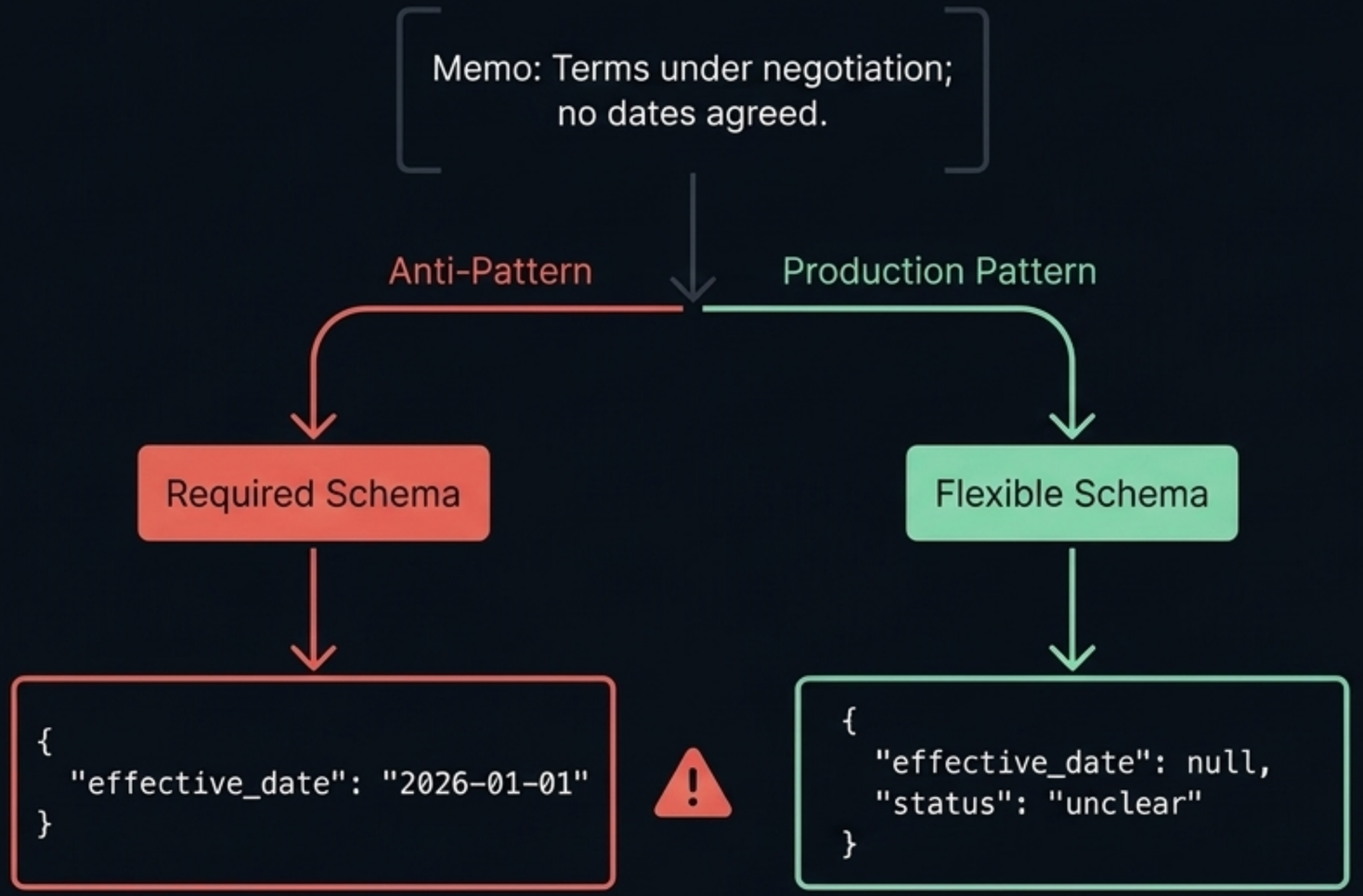
```
[]
```

1. Target 2-5 examples inside <examples> tags.

2. Demonstrations teach generalizable judgment better than brittle keyword matching.

3. Keep formatting invariant across all examples to prevent structural drift.

Null Boundaries and Preventing Fabrication



- Vary surface structure in multishot examples.
- Explicitly demonstrate null values to prevent hallucination.
- Use 'unclear' as a truthful escape hatch for ambiguous documents.

Cognitive Control and Adaptive Thinking

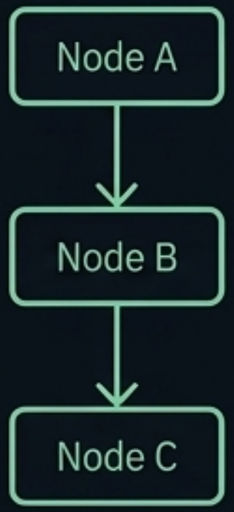




Passing non-default temperature, top_p, or legacy budget_tokens on modern endpoints returns a hard HTTP 400 error. Steer determinnism through schemas and adaptive effort.

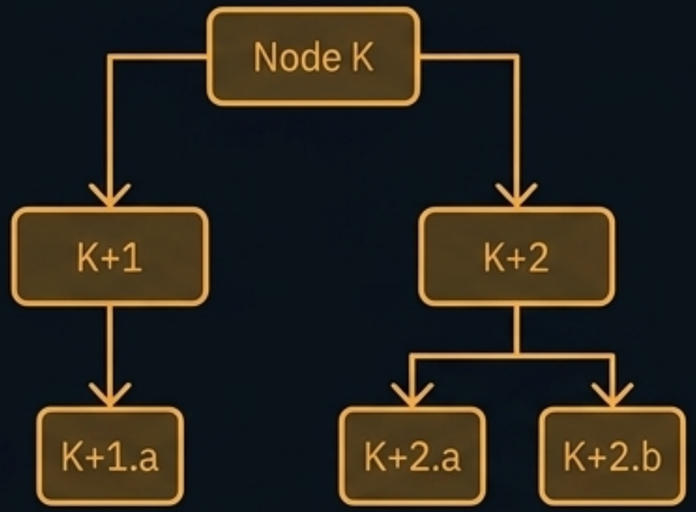
Task Decomposition Topologies

Fixed Sequential Chain



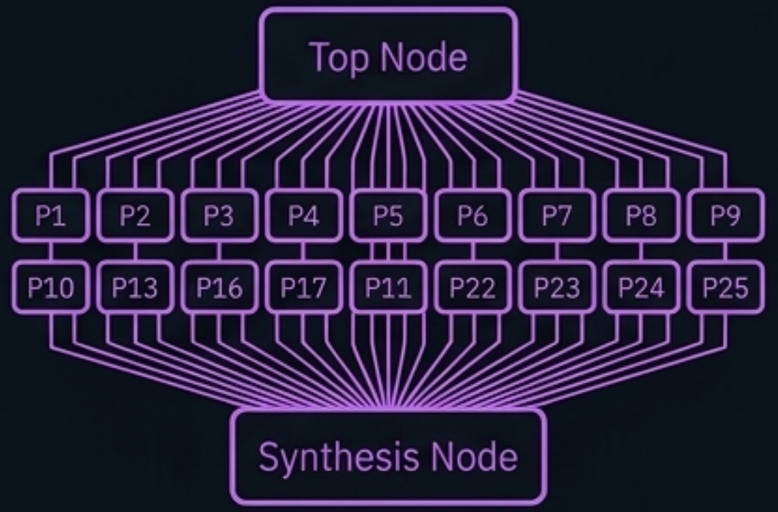
Extract -> Cross-check -> Render table.

Dynamic Adaptive Tree



Map module graph -> Rank risks -> Inspect dependencies.

Parallel Fan-out



Summarize 25 transcripts concurrently -> Synthesize final report.

Mega-prompts dilute attention. Choose decomposition based on whether stages are known upfront, adaptive, or independent.

Structured Outputs: The Three Mechanisms

Prompt-only formatting

Syntax Guaranteed?: ✗ No.

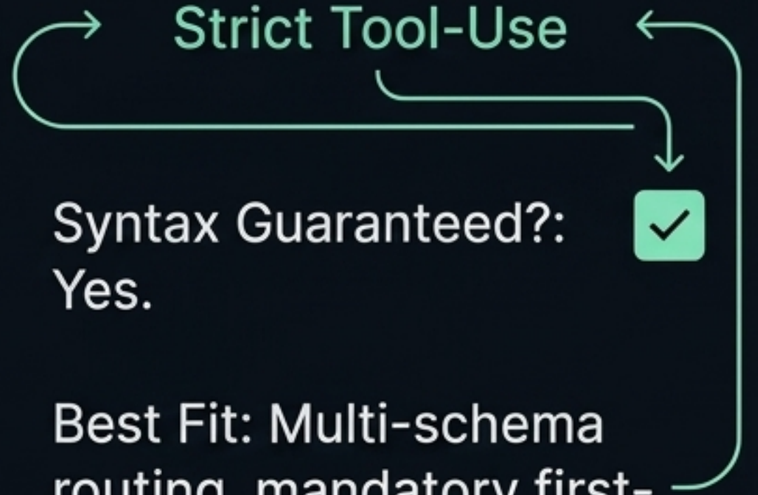
Best Fit: Streaming Markdown to humans.

Native JSON Schema

Syntax Guaranteed?: ✓ Yes.

Best Fit: Single-step extraction (e.g., response_schema / **output_config.format**).

Strict Tool-Use

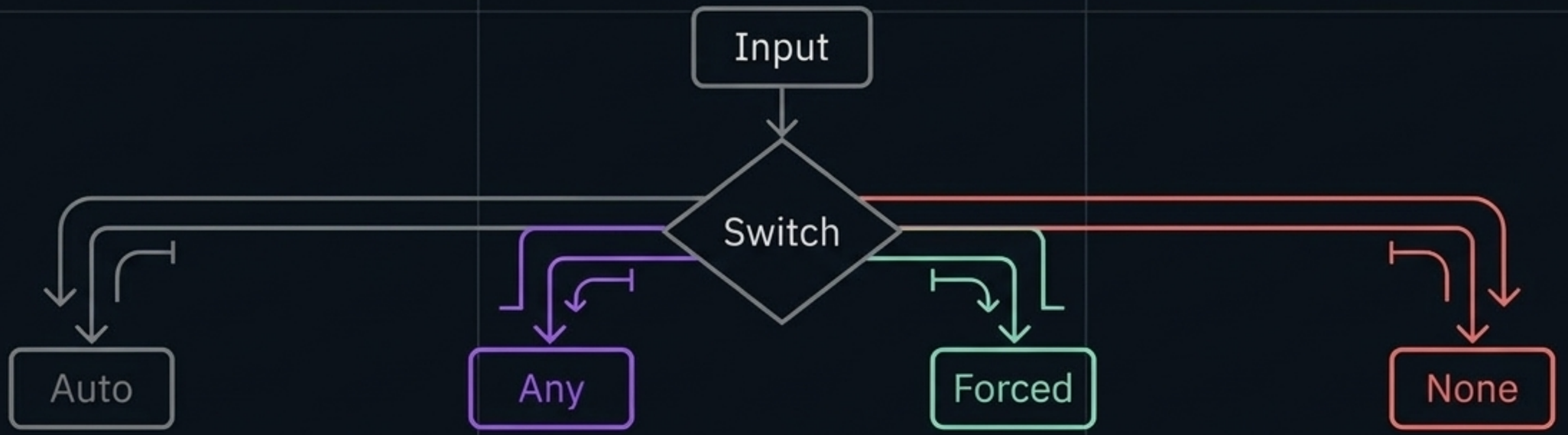


Syntax Guaranteed?: ✓ Yes.

Best Fit: Multi-schema routing, mandatory first-step extraction.

Note: Hitting max_tokens is a hard cutoff, producing truncated JSON regardless of schema guarantees. Size token limits with generous headroom.

API Enforcement: Tool Choice Mechanics



Loops freely in agentic chat until `end_turn`.

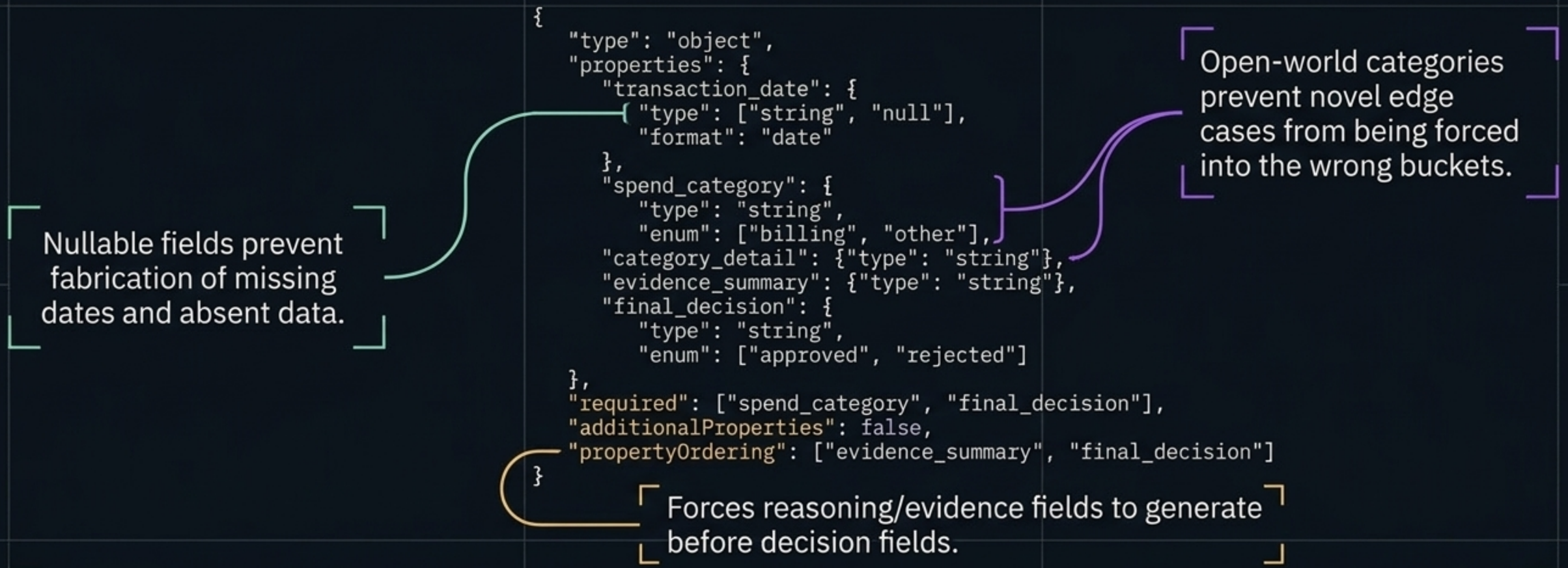
Forces routing. Model must pick from `extract_invoice`, `extract_receipt`, etc.

Mandatory extraction step. Locked to a specific named function before downstream enrichment.

Forbids execution but keeps schema definitions in context.

Tool choice is an API lever, not a prose suggestion. Use 'any' to handle unknown document types securely.

Bulletproof Schema Design Rules



Never describe a JSON shape in prose with different key names from the schema. Set additionalProperties: false to enforce strict compilation.

The Validation Chasm: Syntax vs. Semantics

Syntax / Structural

```
{
  "user": "Alice"
  "id": "A123",
  "active": "true",
  "role": 1
}
```



API Constrained Decoding

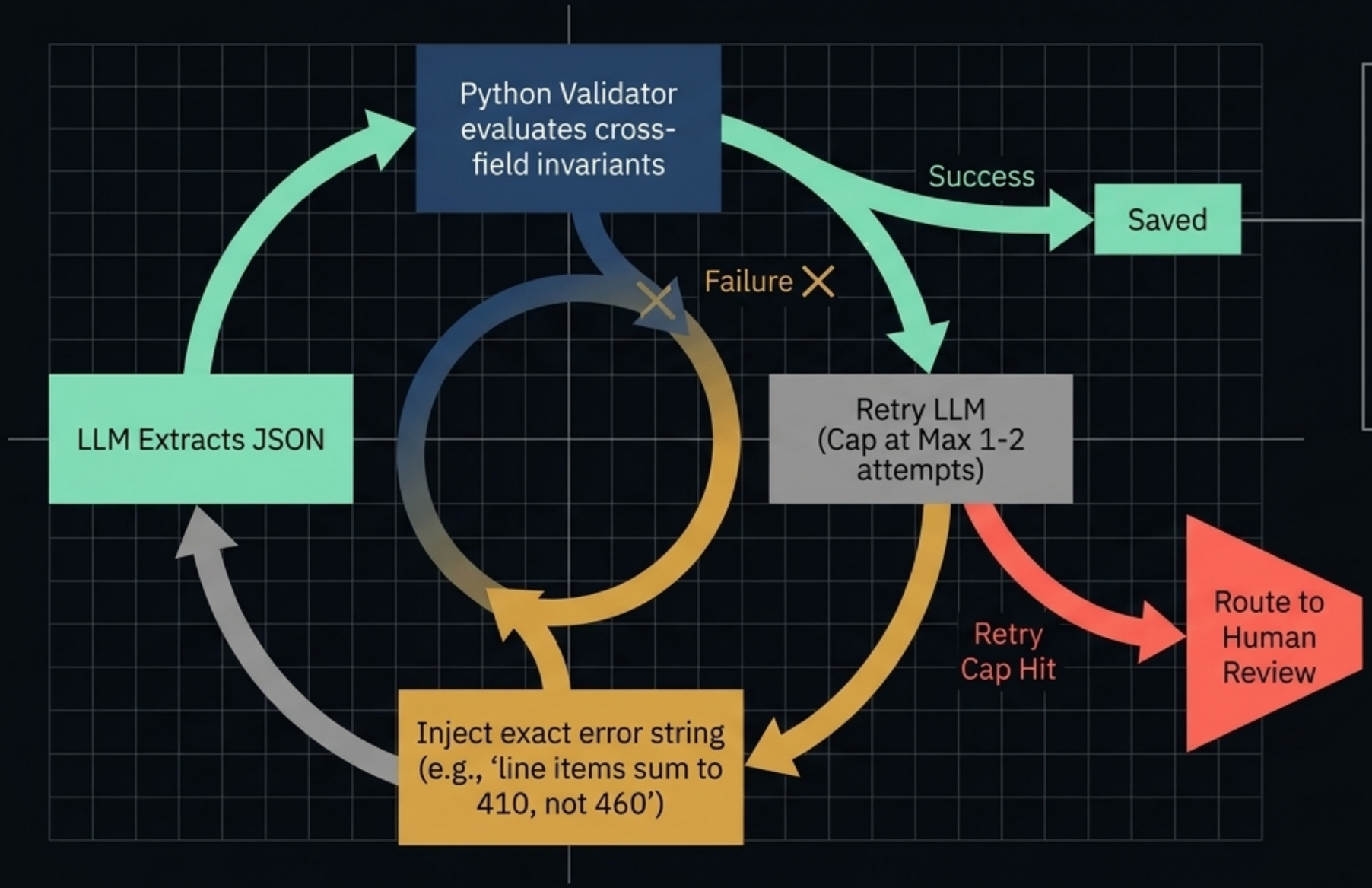
Eliminated by the API natively.

Semantic / Logic

```
{
  "line_items": [
    {"id": "L1", "amount": 400},
    {"id": "L2", "amount": 540}
  ],
  "sum(line_items)": 940,
  "stated_total": 1040
}
```

Invisible to the API; Constrained decoding cannot prevent a shipping date from being copied into an invoice date. Requires deterministic Python checks.

Designing the Validation-Retry Loop

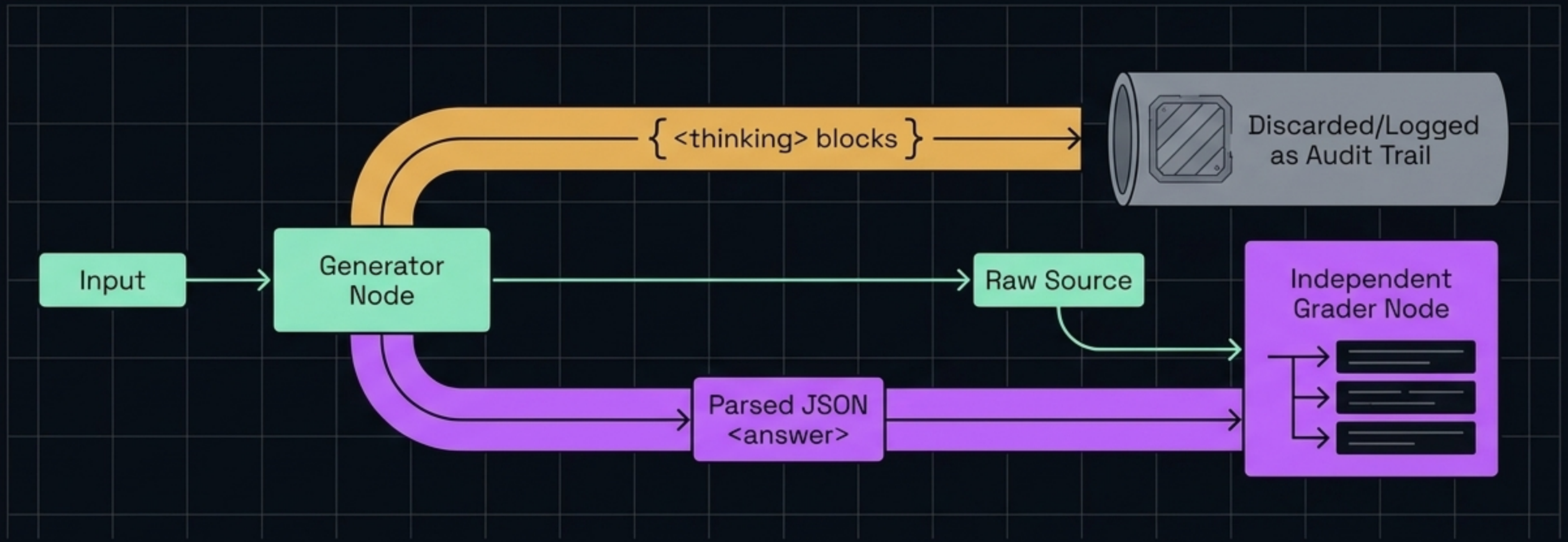


Never retry indefinitely.

Retrying on data that is genuinely missing causes the model to fabricate.

Supply the `original source`, `failed JSON`, and exact `validator message` on the retry turn.

Multi-Pass Review & CoT Isolation



Keep reasoning out of the final answer. Exposing internal reasoning ahead of JSON breaks parsers. Grade the parsed answer against labeled data, not the fluency of the explanation.

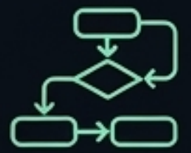
Execution Modalities: Sync vs. Batch SLAs

Sync API

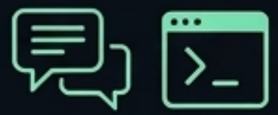


Speed:
Interactive (seconds)

Cost:
Standard expensive rates



Capabilities:
Supports multi-turn tool execution



Use Case:
Live chat, interactive IDE assistants

Batch API

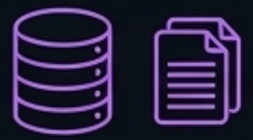


Speed:
Up to 24h SLA (out-of-order execution)

Cost:
50% cheaper



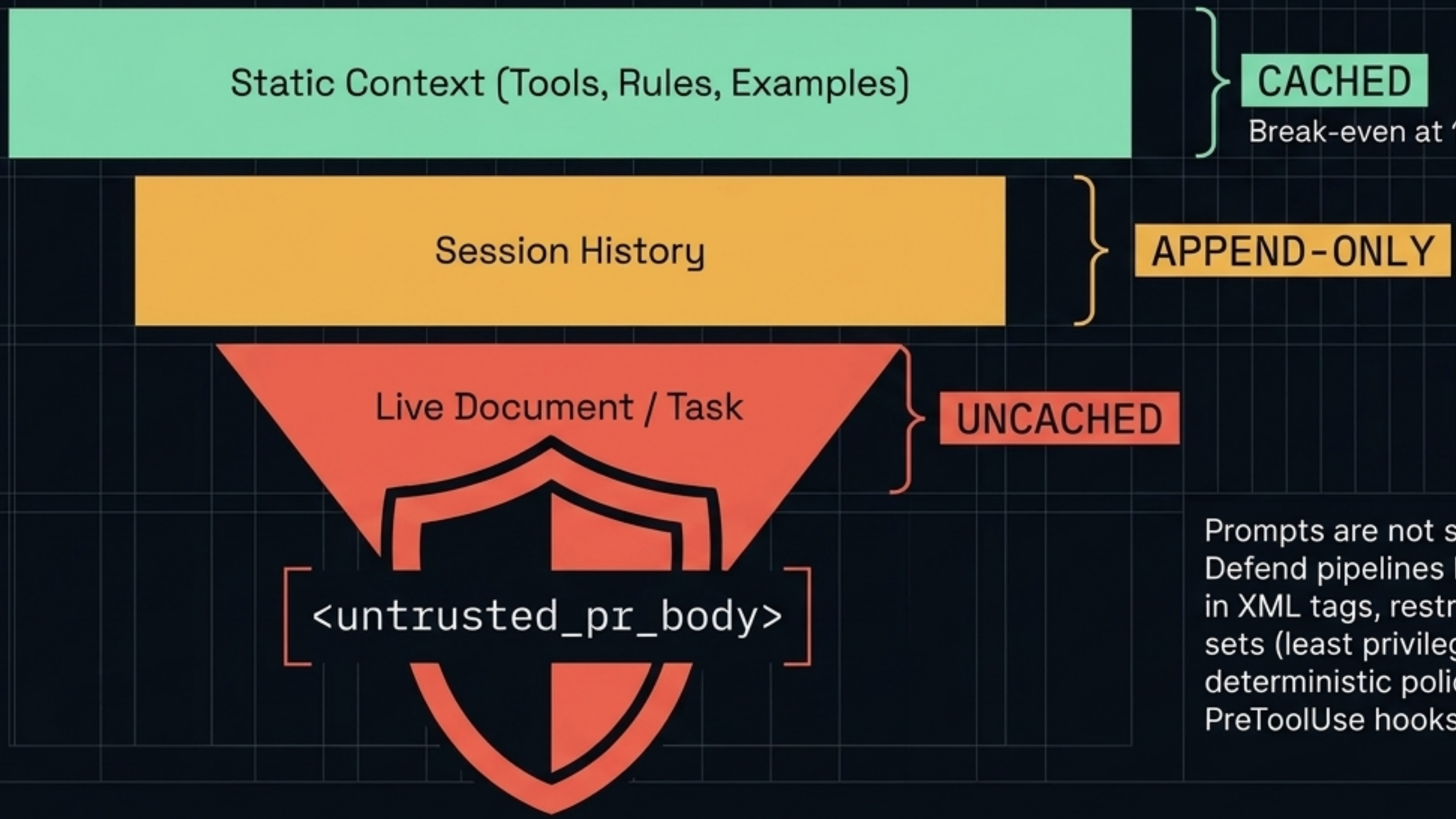
Capabilities:
Single-turn only. Requires custom_id for correlation



Use Case:
Overnight log summarization, massive offline evaluation sweeps

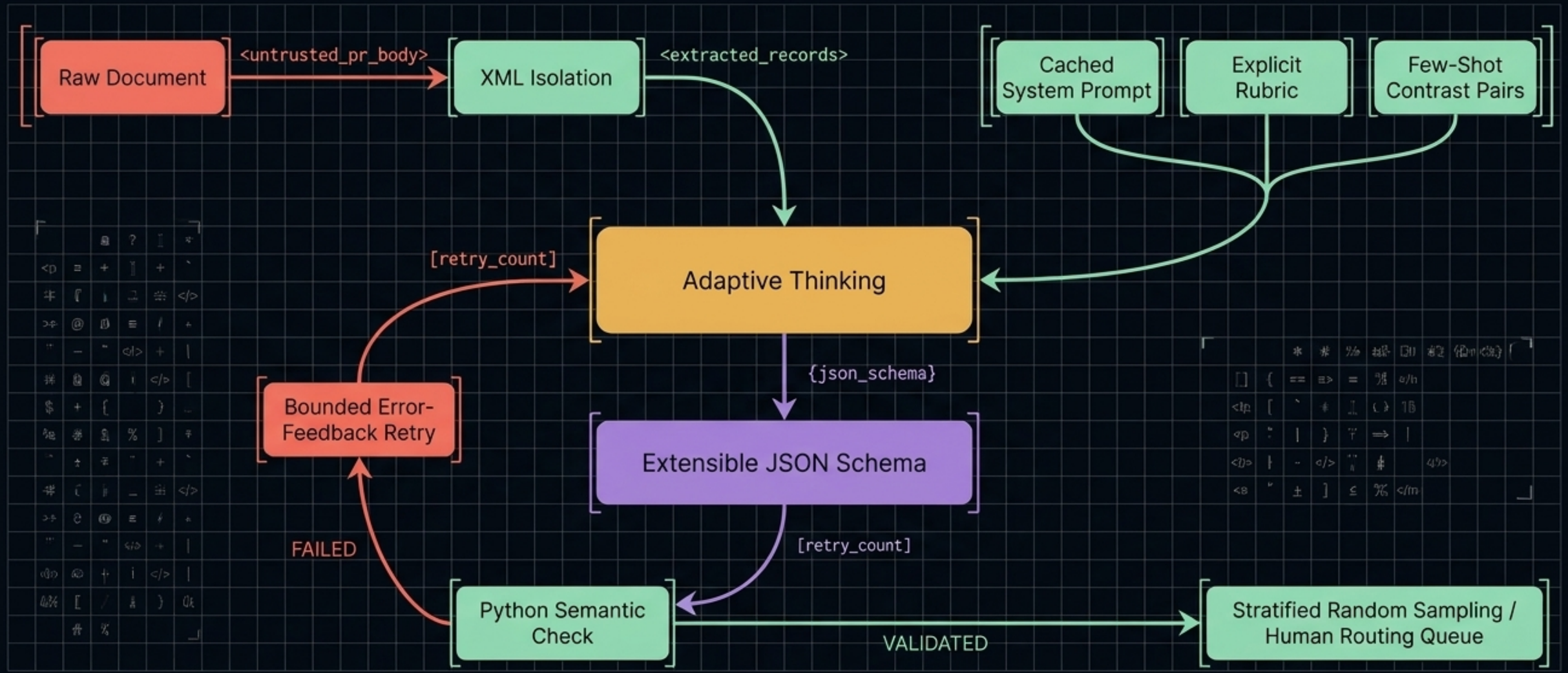
Prompt chaining in batch requires passing validated structured data in `<extracted_records>` tags to the next stage. Do not re-send raw text, which invites contradictions.

Caching Architecture & Untrusted Inputs



Prompts are not security boundaries. Defend pipelines by isolating inputs in XML tags, restricting tool sets (least privilege), and enforcing deterministic policy checks via PreToolUse hooks in application code.

Synthesis: End-to-End Extraction Blueprint



This blueprint unites authoring levers (prose, isolation, multishot) with API enforcement levers (schemas, execution modes, retries) to transform unpredictable natural language into a deterministic software component.