# ¡Full Paper Title¿

**Anonymous Authors**[1]

## Abstract

# 1. Introduction

# 2. Related Work

# 3. Preliminaries and Problem Setup

# 4. Method

## 4.1. Overview

We propose *Natural Language Edge Labelling* (NLEL), a control layer for structured language-model (LM) reasoning in which each edge carries a natural-language label that specifies *how* the next step should proceed (e.g., "seek a counterexample", "work backward", "apply an anthropological lens; probe for defeaters"). A dedicated *tuner* LM reads a tuple $(P, L, C)$—the parent node $P$, the edge label $L$, and the current context $C$—and maps it directly to a control vector $\Pi$ that configures decoding, search, retrieval, and verification for the next expansion.

## 4.2. Inputs, Outputs, and Mapping

**Inputs.** $P$ is the current parent state (text and optional structure). $L$ is a free-form natural-language directive for the edge. $C$ denotes the remaining state, which can include the partial tree/graph, concise summaries of the frontier and siblings, budget trackers, and verifier configuration.

**Output.** A control vector $\Pi$ whose fields actuate the reasoning stack. A task-agnostic schema can include:

- **Decoding:** temperature, top-$p$, max tokens, repetition penalty;
- **Search:** branch quota, variance/risk coefficient $\beta$, and a UCT/exploration constant;
- **Retrieval:** mixture weights over indices or corpora;
- **Verification:** number and strictness of checks.

**Mapping.** Let $\Psi : (P, L, C) \mapsto \Pi$ denote the tuner mapping. In our prompt-only instantiation (Section 4.4), $\Psi$ is realized by a JSON parameter emitter that respects a schema with bounds and learns from a compact in-prompt ledger of historical expansions.

## 4.3. Expansion Procedure

We expand the structure in three steps:

1. **Select an edge label $L$.** Labels are natural-language imperatives specifying *how* to think next (e.g., generate a counterexample, analogize, or recurse on a subgoal).
2. **Emit control $\Pi = \Psi(P, L, C)$.** The tuner LM consumes $(P, L, C)$ and produces a single control vector adhering to a schema with bounds.
3. **Expand under $\Pi$.** Generate or select the child using the actuated settings; update the frontier summaries and budgets in $C$.

## 4.4. Prompt-Only JSON Parameter Emitter (JPE)

The tuner LM receives three ingredients in the prompt: (i) a concise *schema* that specifies control fields and bounds; (ii) a *historical ledger* of $(P_i, L_i, C_i) \mapsto \Pi_i$ with outcomes, where rows are tagged as *Pareto* or *dominated* to provide contrastive signals about efficient trade-offs; and (iii) the *current case* $(P, L, C)$. It emits a single JSON object $\Pi$ that must validate against the schema. The ledger can be curated with a lightweight objective that balances task success against compute usage and verification reliability (e.g., success@compute with penalties for excessive tokens or failed checks).

## 4.5. Context Features

To keep $C$ compact and measurable, we surface a small set of features that capture the state of search:

- **Frontier uncertainty:** median $\sigma$ across candidate

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

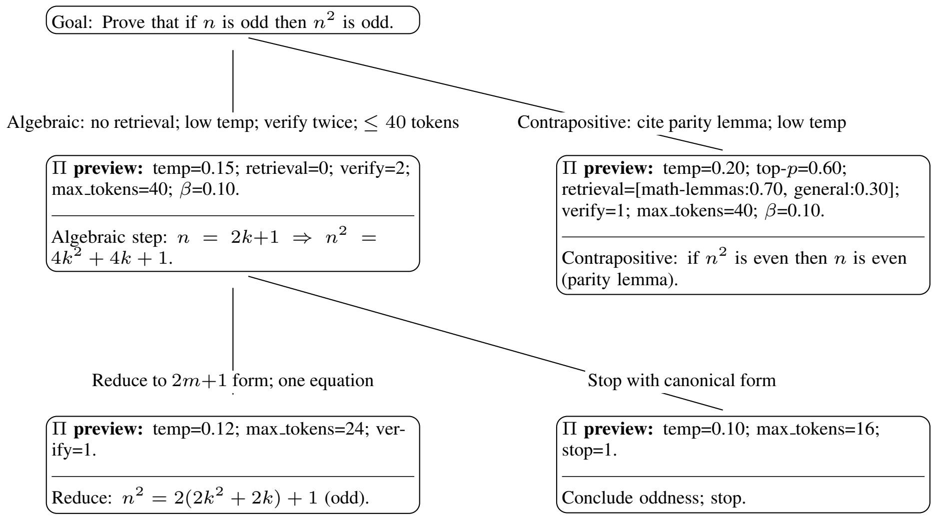


*Figure 1.* Rows of width $[1, 2, 2]$ with edges drawn explicitly from the bottom of each parent node to the top of the child node. Natural-language edge labels sit on the edges near the child (never adjoining each other). Each child node starts with an upright $\Pi$ *preview*, then a horizontal separator, then the node's reasoning content.

downstream values (from ensembles, bootstraps, or dropout estimates);

- **Novelty deficit:** median nearest-neighbor distance among frontier candidates (embedding or lexical);
- **Depth:** distance from root (enables exploration annealing and quota schedules);
- **Sibling/frontier summaries:** best $(\mu, \sigma)$ among siblings; counts by edge label; budget usage.

### 4.6. Downstream Selection (Agnostic to NLEL)

Given $\Pi$, any downstream planner can be used. A simple variance-aware score combines an estimate of value and uncertainty, for example $S = \mu + \beta\,\sigma$, optionally augmented with a UCT-style exploration term. The specific selector is orthogonal to NLEL; $\Pi$ only sets the knobs.

### 4.7. Stability and Safety

We employ non-intrusive guards: (i) strict schema/bounds validation for emitted JSON; (ii) projection into a trust region around safe defaults to prevent pathological jumps; and (iii) depth-annealed exploration so late-depth expansions remain conservative.

### 4.8. Design Notes

NLEL is compatible with a non-reasoning tuner or a reasoning tuner (e.g., CoT/ToT) used *only* as a controller. The child reasoner can be held fixed to cleanly attribute outcomes to the edge label and the control vector $\Pi$.

## 5. Theory (Optional)

## 6. Experiments

## 7. Limitations

## 8. Conclusion

## Impact Statement

## References

## A. Additional Experimental Details

## B. Proofs

## C. Extra Results