¡Full Paper Title¿

Anonymous Authors[1]

Abstract

1. Introduction

2. Related Work

3. Preliminaries and Problem Setup

Reasoning structure. We model inference as expansion of a directed tree (or a DAG with tie-breaking) $G = (V, E)$. Each node $v \in V$ is a *reasoning step* with textual content x_v; the root v_0 holds the task statement. Each edge $e = (u \to v) \in E$ carries a natural-language label L_e and induces a control vector Π_e used to expand the child v. We distinguish two roles: a *labeller* LM Λ that proposes edge labels, and a *tuner* LM Ψ that emits control, with mappings

$$L = \Lambda(P, C), \qquad \Pi = \Psi(P, L, C).$$

Here P denotes the parent node text (and any exposed metadata), and C denotes a compact context.

Context C. We keep C compact and measurable. In our setting, C may include:

- **Frontier uncertainty:** summaries such as the median σ across candidate values;
- **Novelty:** nearest-neighbor distances among frontier candidates (embedding or lexical);
- **Depth:** distance from the root;
- **Sibling/frontier summaries:** best (μ, σ) among siblings;
- **Raw label history:** the most recent edge labels as *strings* (from siblings and, optionally, a short frontier window);
- **Budgets:** token usage, retrieval calls, and verification outcomes.

Control schema Π. The tuner controls a task-agnostic set of fields:

- **Decoding:** temperature, top-p, maximum tokens, repetition penalty;
- **Generation:** `gen_count` $\in \mathbb{N}^+$ (bundle size under this label);
- **Search:** branch quota and an exploration coefficient β;
- **Retrieval:** mixture weights over indices or corpora;
- **Verification:** number and strictness of checks;
- **(Optional) Selection hint:** `keep_k` $\in \mathbb{N}^+$ (if set, passed to the child-selection module).

Given Π, a downstream selector (agnostic to NLEL) can use scores such as $S = \mu + \beta\,\sigma$ or a standard ToT culling operator.

Edge labels. Labels are produced by Λ from (P, C).

Problem instances. An instance consists of a task T, root v_0 text, and an evaluation function producing (μ, σ) for partial answers. Unless noted, we treat G as a tree; extension to DAGs is straightforward by merging isomorphic textual states.

Notation summary.

Symbol	Meaning
P	parent node content (text + exposed metadata)
L	natural-language edge label
C	compact context features (bulleted above)
Λ	labeller LM mapping $(P, C) \to L$
Ψ	tuner LM mapping $(P, L, C) \to \Pi$
Π	control vector (decoding, search, retrieval, verification)
μ, σ	value / uncertainty estimates used by the selector
w	retrieval mixture weights over indices/corpora
β	exploration coefficient in selection
c_e, C_t	per-edge and cumulative compute cost
`gen_count`	generation bundle size (per edge label)

[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.

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 Π 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 Π 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 β, 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 **??**), Ψ 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 at a parent p in two phases: label emission and bundle generation, followed by a single selection step.

1. **Emit labels.** Use the labeller to obtain a set of edge labels for p: $\mathcal{L}_p = \{L_1, \dots, L_m\}$, where each $L_i = \Lambda(P, C)$. The number of labels may be governed by a search quota or policy.
2. **Generate bundles under each label.** For each $L \in \mathcal{L}_p$, obtain control $\Pi = \Psi(P, L, C)$ and generate a bundle of `gen_count` candidate children under L using Π.
3. **Select children (ToT).** Let $\mathcal{B}(L)$ denote the bundle generated under label L. Form the union of all candidates for the parent, $\mathcal{C}_p = \bigcup_{L \in \mathcal{L}_p} \mathcal{B}(L)$, and apply the standard ToT child-selection operator to $\mathcal{C}_p$. We inherit ToT's selector as-is.
4. **Update state.** Add survivors to the frontier and update C (budgets, summaries, raw label history strings).

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 Π 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 σ across candidate 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 (μ, σ) among siblings; raw label history (strings); budget usage.

4.6. Downstream Selection (Agnostic to NLEL)

We inherit the standard ToT child-selection operator and apply it once to the union of all candidates produced for a parent (across labels).

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.

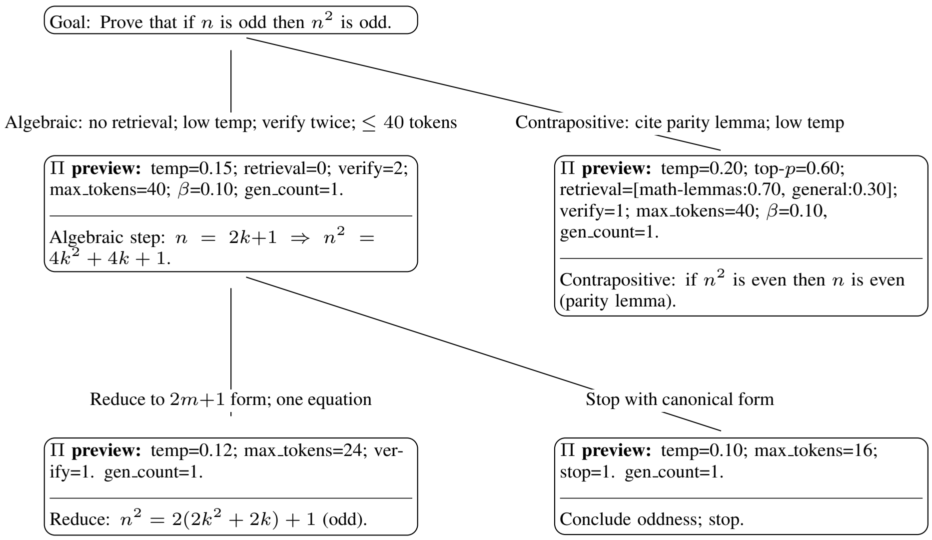

Figure 1. A synthetic example of NLEL being used in a ToT setting. For simplicity, gen_count is set to one for all Π_i.

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 Π.

5. Theory (Optional)

6. Experiments

7. Limitations

8. Conclusion

Impact Statement

References

A. Additional Experimental Details

B. Proofs

C. Extra Results