Natural Language Edge Labelling (NLEL): Directing Structured LM Reasoning via Edge-Level Natural-Language Control

Abhinav Madahar

October 1, 2025

Abstract

We propose **Natural Language Edge Labelling (NLEL)**, a control layer for Tree-/Graph-of-Thoughts in which each edge carries a *natural-language* label that directs how the next step should proceed. A *tuner language model*—which may be a non-reasoning or a reasoning model—reads the tuple (P, L, C) (parent node P, natural-language edge label L, and context C) and maps it directly to a control vector Π that configures decoding, search, retrieval, and verification for the next expansion; the child is then expanded under Π. Under the hypothesis that natural language captures nuance that small formal symbolic taxonomies cannot, inserting an intermediate symbolic layer would be a lossy transformation; we therefore use natural language as the native control interface (see Choi, 2022).

Keywords: structured reasoning, tree-of-thoughts, graph-of-thoughts, natural-language control, process supervision, variance-aware search

1. Executive summary

We introduce **Natural Language Edge Labelling (NLEL)**: a control layer for structured LM reasoning in which each *edge* carries a *natural-language* label (e.g., "apply an anthropological lens; probe for defeaters"). A *tuner language model*—either *non-reasoning* or *reasoning* (e.g., CoT/ToT)—reads the tuple (P, L, C) and *maps it directly* to a *control vector* Π (decoding, search, retrieval, verification knobs). The child is then expanded under Π.

What can appear in C? Example elements include:

- The **full current tree/graph** (as text or structured form).
- **Sibling/frontier summaries**: median uncertainty σ across candidates; novelty statistics; best (μ, σ) among siblings; counts by edge label.
- **Budgets and quotas**: remaining compute, per-form quotas, depth caps.
- **Verifier configuration**: number/strictness of passes; available validators.
- **Task metadata** and **domain status**: e.g., argumentation/attack-support status, threatened-mass metrics.

Why natural language? Under the working hypothesis that natural language can encode approach, perspective, and risk posture with nuance exceeding compact symbolic taxonomies, a symbolic intermediary would be *lossy.* NLEL therefore treats edge language as the *native control API*, converting it directly to actuators (Choi, 2022).

2. Motivation: Directing the reasoning process

The overarching aim is **control**: to *direct how* a model reasons—its approach, perspective, and risk posture—rather than only shaping output format. CoT/ToT/GoT organize multi-step inference, but edges are commonly untyped dependencies or fixed modules. In **NLEL**, the edge becomes a *first-class, executable natural-language control object*:

- **Expressive direction**: labels such as "seek a counterexample", "work backward", "analogical mapping", "anthropological lens; probe for defeaters" specify *how to think next.*
- **Direct NL→control**: edge text maps *directly* to Π (sampler, search, retrieval, verification) with no symbolic intermediary (Choi, 2022).
- **Process attribution**: because each expansion is conditioned on an explicit label and a concrete Π, performance changes can be attributed to *edge-level decisions.*

3. Problem statement & formalization

Inputs: a tuple (P, L, C) where P is the parent node, L is *natural-language text* describing the desired relationship/approach for the next edge, and C is the remaining state (including the current tree/graph, sibling/frontier summaries, budgets, verifier config, etc.).

Output: a control vector Π with fields such as:

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

Let Ψ denote the *tuner mapping* from (P, L, C) to Π: $\quad \Pi = \Psi(P, L, C)$. Historical expansions are labeled *Pareto* or *dominated* with respect to a multi-objective outcome vector (e.g., ΔV_{root}, tokens used, verification events, success@compute) and presented in-prompt.

4. Approach

4.1 Three-step expansion

1. **Choose the edge label** L (natural-language relationship/approach).
2. **Find the control vector** $\Pi = \Psi(P, L, C)$.
3. **Expand the child** under Π.

4.2 Prompt-Only JSON Parameter Emitter (JPE)

A *tuner LM* reads (i) a **schema** specifying control fields and bounds, (ii) a **historical ledger** of $(P_i, L_i, C_i) \Rightarrow \Pi_i$ with outcomes and tags (*Pareto*/*dominated*), and (iii) the **current case** (P, L, C); it then *samples* a single JSON control vector Π that respects the schema and bounds.

Tuner type (hyperparameter of the architecture):

- **Non-reasoning LM** (direct decoding),
- **CoT-LM** (internal chain-of-thought while tuning; outputs JSON only),
- **ToT-LM** (internal search-style deliberation while tuning; outputs JSON only).

4.3 Context features (concise, measurable)

- **Frontier uncertainty**: median σ across candidate downstream values (ensembles/bootstraps/dropout).
- **Novelty deficit**: low median nearest-neighbor distance among frontier candidates (embedding + lexical).
- **Depth**: distance from root (for exploration annealing and quotas).

4.4 Downstream selection (agnostic to NLEL)

Given Π, one can use a variance-aware score such as

$$S \;=\; \mu + \beta\,\sigma + c_{\mathrm{uct}}\sqrt{\frac{\log N(\text{parent})}{N(\text{edge})+1}},$$

with β and c_{uct} provided by Π.

5. Historical ledger & labels (in-prompt supervision)

For each logged expansion we store (P, L, C), the chosen Π, and outcomes (e.g., ΔV_{root}, tokens, verification events, success). Rows are tagged as *Pareto* or *dominated* and presented with *Pareto* rows first, followed by *dominated* contrasts matched on context. This yields contrastive, weight-free signals about efficient trade-offs directly in the prompt.

6. Distinguishing contributions

1. **Edge-level natural-language control**: edges carry executable *natural-language* labels; label text directly controls decoding/search/retrieval/verification.
2. **Direct NL$\rightarrow$control mapping**: *no* intermediary symbolic layer (avoids lossy transformation; preserves nuance/compositionality).
3. **Prompt-only controller**: the tuner learns from an in-prompt history; no separate training or retrieval infrastructure required.

4. **Tuner ablations as a first-class scientific question**: non-reasoning vs CoT vs ToT tuner (controller only), holding the child reasoner fixed.

5. **Process-level attribution**: outcomes attributable to the edge label and Π, enabling clean analysis of *how* control affects reasoning.

7. Relation to prior work (brief)

CoT/ToT/GoT structure multi-step inference but generally treat edges as untyped or predeclared modules; NLEL makes edges *natural-language control points* with direct actuator effects. ReAct/Reflexion interleave reasoning with actions/feedback; NLEL targets *edge-time control* of generation and search. Typed reasoning/meta-prompting choose reasoning modes; NLEL uses *edge phrasing* as the unified control interface, converted *directly* to knobs.

8. Method details (ready to implement)

8.1 Input tuple & context separation

- **Parent** P: the node being expanded.
- **Label** L: *natural-language text* describing the desired relationship/approach for the edge.
- **Context** C: remaining state (including the full current tree/graph, sibling/frontier summaries, budgets, verifier config, etc.).
- **Historical ledger**: contrastive rows with *Pareto*/*dominated* tags and outcome metrics.

8.2 Control schema (example; adjustable)

```
{
  "decode":   { "temperature": [0.1, 1.2], "top_p": [0.5, 1.0], "max_tokens": [16, 256] },
  "search":   { "branch_quota": [1, 6], "risk_beta": [0.0, 1.5], "uct_c": [0.0, 2.0] },
  "verify":   { "level": ["normal","strict"], "passes": [0, 3] },
  "retrieval":{ "anthropology": [0.0,1.0], "general": [0.0,1.0], "other": [0.0,1.0] }
}
```

Continuous fields are clamped to bounds; mixtures are normalized to sum to 1.

8.3 Tuner prompt (expanded template)

System:

> *You are a control-strategy tuner. Read the historical examples and their outcomes. For the CURRENT CASE, output* ***only*** *a JSON control object that satisfies the schema and* ***maximizes*** *the objective below. Do not include any text outside JSON.*

Objective (example):

Maximize success@compute and ΔV_{root}. *Penalties:* $\lambda_{compute} = 0.3$ *per 100 tokens;* $\lambda_{risk} = 0.2$ *per verification failure.*

Schema (with bounds): Insert the JSON schema (above), adapted per task.
Historical ledger (contrastive): Provide rows in the format:

```
# Example k  (PARETO or DOMINATED)
PARENT: <text of P>
LABEL: "<natural-language edge label>"
CONTEXT-HEADER: depth=…; budgets=…; frontier: median  =…; novelty=…; siblings: …
CONTROL: { ... JSON within bounds ... }
OUTCOMES: ΔV_root=…; success=…; tokens=…; verify_fail=…
```

List *Pareto* rows first, then *dominated* contrast pairs matched on context regime.
Current case:

```
PARENT: <text of P>
LABEL: "<natural-language edge label>"
CONTEXT-HEADER: depth=…; budgets=…; frontier: median  =…; novelty=…; siblings: …
CONTEXT-FULL: <full current tree/graph and state, as budget allows>
```

Assistant (tuner LM): Outputs one JSON control object Π respecting schema and bounds. The tuner may be a non-reasoning LM, CoT-LM, or ToT-LM.

8.4 Stability & safety (non-intrusive)

- **Schema/bounds validation** for emitted JSON.
- **Trust-region projection** around safe defaults to prevent pathological jumps.
- **Depth-annealed exploration** to keep late-depth expansions conservative.

9. Evaluation plan

9.1 Benchmarks

We target the following suite:

- **GSM8K (full)** — multi-step arithmetic word problems.
- **ARC-Challenge** — adversarial multiple-choice science questions.
- **HotpotQA (distractor)** — two-hop open-domain QA with supporting facts.
- **EntailmentBank (S/M)** — multi-hop textual entailment with explanations.
- **ProofWriter (depth ≤ 3–4)** — rule-based reasoning with short proof chains.
- **MBPP (standard)** — Python function synthesis with unit tests.

We will *limit sample counts as needed to remain within budget*, while maintaining equal-compute comparisons and process metrics.

9.2 Baselines

- **Unlabeled edges** (vanilla ToT/GoT control).
- **Label-as-node-text only** (edge label does not control actuators).
- **Heuristic control rules** (n-gram triggers for Π).
- **Entropy/diversity-guided branching** (no NL control).
- **Risk-aware UCT without NL edges**.

9.3 Ablations

- **Tuner type**: non-reasoning LM vs CoT-LM vs ToT-LM (controller only; child fixed).
- **Ledger composition**: with vs without *dominated* contrasts; with vs without *Pareto* tags.
- **History size**: include larger in-prompt histories where feasible to test data-efficiency of in-context tuning within budget.
- **Symbolification ablation**: NL→symbolic tag→control vs **direct NL→control** (NLEL).

9.4 Metrics

- **Success@compute** (equal-compute curves).
- $\Delta V_{\mathbf{root}}$ uplift per expansion.
- **Defeater-discovery latency** (where applicable).
- **Pareto-coverage** of outcomes relative to historical fronts.
- **Control validity rate** (JSON correctness, projection frequency).
- **Cost per improvement** (tokens or $ per % gain).

10. Budget (compute-rich; tuner LM = child reasoner)

Assumption: the *same frontier-scale LM* is used as both *tuner* and *child reasoner*.

- **Compute (GPU/Cloud):** $2,200–$3,200
  Approx. 350–500 GPU-hours on A100/L40S-class hardware across experiments (tuner=child); includes ablations and repeated runs.
- **Model/API & tooling:** $600–$1,500
  ChatGPT Pro (research assistant role): $200/month × 3–6 months = $600–$1,200.
  Optional ancillary API usage or storage/egress: $0–$300.

- **Publication & dissemination (ICML 2026):** \$2,000–\$3,200
  Registration (single author): \$1,000–\$1,600.
  Travel & lodging (economy airfare + 3–4 nights): \$900–\$1,500.
  Poster/misc.: \$100.

Total indicative budget: \$4,800–\$7,700.

11. Risks & mitigations

- **Edge-label overfitting to phrasing.** Include *dominated* contrasts matched by context; vary label paraphrases; test robustness.
- **Prompt budget pressure (larger histories).** Compress ledger headers; standardize row format; scale history size to the token budget; amortize across runs.
- **Control instability.** Enforce schema validation, bounds, trust-region projection, and depth-annealed exploration.
- **Attribution confounds.** Hold the child reasoner fixed when varying tuner type; report equal-compute curves and process metrics.

12. Deliverable

A single deliverable: **Submission to ICML 2026** (main conference) with **open-source code** and **open data** (prompts, logs, and evaluation scripts) sufficient to reproduce all tables and figures.

Appendix

A. Tuner prompt (expanded template)

System:

> *You are a control-strategy tuner. Read the historical examples and their outcomes. For the CURRENT CASE, output **only** a JSON control object that satisfies the schema and **maximizes** the objective below. Do not include any text outside JSON.*

Objective (example):

> *Maximize success@compute and* ΔV_{root}*. Penalties:* $\lambda_{compute} = 0.3$ *per 100 tokens;* $\lambda_{risk} = 0.2$ *per verification failure.*

Schema (with bounds): Include the JSON schema and numeric bounds as in §8.2.
Historical ledger (contrastive): Rows in the format:

```
# Example k  (PARETO or DOMINATED)
PARENT: <text of P>
LABEL: "<natural-language edge label>"
CONTEXT-HEADER: depth=…; budgets=…; frontier: median =…; novelty=…; siblings: …
CONTROL: { ... JSON within bounds ... }
OUTCOMES: ΔV_root=…; success=…; tokens=…; verify_fail=…
```

List *Pareto* rows first, then *dominated* contrast pairs matched on context.
Current case:

```
PARENT: <text of P>
LABEL: "<natural-language edge label>"
CONTEXT-HEADER: depth=…; budgets=…; frontier: median =…; novelty=…; siblings: …
CONTEXT-FULL: <full current tree/graph and state, as budget allows>
```

Assistant (tuner LM): Outputs one JSON control object Π; tuner type may be non-reasoning, CoT, or ToT.

B. Minimal runtime pseudocode

```
def nlel_tune(P, L, C, ledger, schema, objective):
    prompt = render_prompt(schema=schema,
                           objective=objective,
                           history_block=ledger,
                           parent=P, label=L, context=C)
    json_out = tuner_lm.sample(prompt)           # tuner = non-reasoning or reasoning LM
    Pi = parse_json(json_out)
    Pi = clamp_and_project(Pi, schema, trust_region)  # bounds + stability
    return Pi

def expand_with_Pi(P, L, C, Pi):
    child = child_reasoner.generate(P, L, C, controls=Pi)
    return child
```

C. Notation

(P, L, C): parent node, natural-language edge label, context. Π: control vector (actuators). Ψ: tuner mapping $(P, L, C) \mapsto \Pi$. μ, σ: estimated mean and uncertainty of candidate downstream values.

One-paragraph summary

Natural Language Edge Labelling (NLEL) treats each edge in structured LM reasoning as a *natural-language control primitive*. A tuner LM—non-reasoning or reasoning—reads (P, L, C) and *directly* emits a control vector Π (decoding, search, retrieval, verification) with *no* intermediary symbolic layer, avoiding a lossy transformation under the hypothesis that natural language conveys nuance better than small tag sets (Choi, 2022). We present a prompt-only instantiation that learns from an in-prompt historical ledger and evaluate on GSM8K, ARC-Challenge, HotpotQA (distractor), EntailmentBank (S/M), ProofWriter (depth $\leq$ 3–4), and MBPP, with ablations over non-reasoning/CoT/ToT tuner types while holding the child reasoner fixed. The deliverable is an ICML 2026 submission with open code and data demonstrating improved compute-efficiency and directed exploration relative to unlabeled or symbolically-typed edges.

References

Choi, Y. (2022). The curious case of commonsense intelligence. *Daedalus, 151*(2), 139–155. https://doi.org/10.1162/daed_a_01906.