CONSTRAINED WIKIGAME: BENCHMARKING DEDUCTIVE REASONING FOR MULTI-STEP PLANNING

Rafael Mosquera-Gómez, Juan Felipe Rodriguez, Martín Díaz-Vélez , Ivan Sousa & Juan Sebastian Jaramillo *
Factored AI
Mountain View, CA 94041, USA
{rafael.mosquera,juan.rodriguez,martin.diaz}@factored.ai
{ivan.alvarenga,juan.jaramillo}@factored.ai

ABSTRACT

Benchmarking LLMs on multi-step planning tasks typically relies on final answer accuracy. This results in evaluation that fails to distinguish correct reasoning from lucky outcomes. We introduce Constrained Wikigame, a benchmark that extends the classic Wikigame (navigating Wikipedia from a source to a target article via hyperlinks) by introducing category constraints. This addition transforms a task where memorization and shortest-path heuristics may drive success into a step-level deduction task, as each decision involves explicitly justifying consistency with the constraint. We benchmark a suite of frontier reasoning and thinking models using both outcome level (success rate, constraint violation and path efficiency) as well as reasoning validity, directly testing whether extended reasoning translates into reliable constrained planning.

1 INTRODUCTION

Evaluating whether large language models (LLMs) have deductive reasoning capabilities by using final-answer accuracy can result in different failure modes: a model may reach the correct outcome for incorrect reasons or provide a wrong answer despite reasoning correctly at intermediate steps, due to planning errors or incorrect assumptions. In multi step tasks, these situations get amplified by the large search space and the compounding of errors.

Graph navigation tasks require sequential decision making, but navigation alone is not enough when trying to determine deductive capabilities: priors (such as memorization) and shortest-path heuristics can yield high success rates even under the first failure mode described earlier. In contrast, enforcing a restriction that must hold at each intermediate step requires step-wise deduction, as each move must be justified as consistent with the constraint, minimizing heuristics and priors determining the outcome alone.

We propose **Constrained Wikigame**: a benchmark that stems as an extension to the well known Wikigame, where a player is meant to navigate Wikipedia articles from a source node to a target node using available hyperlinks at each step. Instead of allowing any hyperlink, this extension involves navigating while maintaining a global constraint on intermediate nodes. The benchmark is designed to test whether step-level reasons are sufficient for reliable constrained planning by comparing outcome success with process metrics derived from the model's step outputs.

Contributions.

- We define a constrained navigation task that isolates global constraint maintenance in multi-step planning on Wikipedia graphs.
- We describe a construction pipeline that produces n test samples with baseline shortest paths, together with programmatically checkable constraints.
- We propose success, process-level, and efficiency metrics, making it possible to thoroughly assess model performance.

*Correspondence to: rafael.mosquera@factored.ai

- We provide a standardized evaluation protocol intended for reproducibility (fixed snapshot, fixed constraints, structured model outputs, deterministic scoring).

2 Related Work

A growing body of research has investigated language model reasoning, especially in structured and unstructured contexts. For example, Transformers as Soft Reasoners over Language (see Clark et al. (2020), demonstrates that transformers can be trained as "soft theorem provers" to perform deductive reasoning on facts and rules expressed in natural language, achieving high accuracy and generalization to deeper reasoning chains than seen in training. This work suggests a paradigm where reasoning does not require formal symbolic systems but can emerge from language-based rule representations, which is conceptually related to our setup of reasoning over interconnected concepts in the WikiGame.

Another complementary approach to structured reasoning comes from Explaining Answers with Entailment Trees (see Neves Ribeiro et al. (2022), which re-frames explanation generation as the construction of multi-step entailment trees connecting facts to hypotheses through intermediate conclusions. The idea of breaking reasoning into sequential inferential steps parallels chain-of-thought (CoT) formats and suggests ways to inspect and debug reasoning processes, an idea we build upon when analyzing model paths.

Recent work has also highlighted limitations in reasoning consistency. Self-Contradictory Reasoning Evaluation and Detection defines a taxonomy of self-contradictory reasoning, where "generated reasoning either fails to justify the predicted answer (Type 1), produces correct outputs via incorrect justifications (Type 2), or contains internal logical contradictions (Type 3) (see Liu et al. (2024)), and evaluates these phenomena in large language models. This taxonomy provides useful definitions for characterizing reasoning failures and motivates evaluation beyond final-answer accuracy, especially in tasks that require stepwise deductions.

Finally, the Evaluating Large Language Models on Wikipedia Graph Navigation Margiotta et al. (2025) study closely relates to our task setting by systematically benchmarking LLM performance on the WikiGame. This work compares human and model strategies under different navigation conditions, highlighting characteristic structural and error patterns of LLM-directed navigation.

Together, these works contribute conceptual or methodological building blocks for our deductive reasoning benchmark.

3 Benchmark Construction

We construct instances from a Wikipedia hyperlink graph (see Wenger (2024)) and annotate nodes with DBpedia Level-2 ontology categories. Each instance consists of a source page s, a target page t, a banned category c_{ban}, and a solution path $p^\star$ produced by an iterative BFS-based procedure that successfully follows the constraint.

3.1 Wikipedia Graph Setup

We extract a directed graph $G = (V, E)$ where V are pages and $(u, v) \in E$ if page u links to page v in the snapshot.

Source selection. We sample sources s subject to: (i) outdegree ≥ 50, and (ii) the page title contains no parentheses. The outdegree criterion reduces states with limited branching, while the title heuristic aims to reduce disambiguation-like pages.

Target selection. For each source, we sample targets t such that: (i) indegree ≥ 50, and (ii) $t \neq s$. The indegree criterion biases toward pages that are reachable by multiple routes, reducing the fraction of instances that are solvable only by a single path.

Baseline shortest path. We compute an unconstrained BFS shortest path p_0 from s to t and require its length to be at least 4 edges. This enforces multi-step planning rather than single hop solutions.

3.2 CONSTRAINT GENERATION

Given (s, t), we generate a constraint by selecting a banned DBpedia Level-2 category as follows:

1. Retrieve target incoming neighbors: $N^-(t) = \{u \in V : (u, t) \in E\}$. (The set of nodes u such that there exists a directed edge from u to t).
2. Select $u \in N^-(t)$ with the highest indegree (ties broken deterministically).
3. Classify u using the DBpedia Level-2 ontology (e.g., `Place`, `Event`, `Organization`, `Food`).
4. If classification is unavailable, fall back deterministically (e.g., choose the next-highest indegree neighbor with an available label; if none, resample (s, t)).
5. Randomly select one ontology class as the banned category from a fixed set, using a fixed seed.

Constraint. Let $\mathrm{cat}(v)$ denote the DBpedia Level-2 category assigned to node v (or $\perp$ if missing). The constraint is:

> *Intermediate nodes (excluding source and target) must not belong to the banned DBpedia category* c_{ban}.

For a path $p = (v_0 = s, v_1, \ldots, v_L = t)$, we require $\forall i \in \{1, \ldots, L-1\}$, $\mathrm{cat}(v_i) \neq c_{\mathrm{ban}}$.

3.3 CONSTRAINED PATH CONSTRUCTION

We construct a constrained solution path via an iterative pruning-and-search algorithm:

1. Randomly sample (s, t).
2. Compute a shortest path ignoring constraints.
3. Classify intermediate nodes on the candidate path.
4. If at least one node on the candidate path belongs to the banned category, we compute the next shortest path excluding the intermediate nodes previously used.
5. Repeat Step 4 up to 20 iterations.

If no valid path is found within the 20 iterations, we resample (s, t).

4 EVALUATION PROTOCOL

An *episode* begins at source s and ends when the model selects the target t, exceeds a step limit, or produces an invalid action (e.g., choosing a non-neighbor). At each step, the model outputs a structured object:{`chosen_node`,`justification restriction_justification`}. We evaluate traces using the stored graph and DBpedia labels.

4.1 SUCCESS RATE (SR)

Let $\mathcal{D}$ be the dataset, and let $\mathbf{1}[\cdot]$ be an indicator. Define $\mathrm{su}(x) = 1$ if the episode for instance $x \in \mathcal{D}$ reaches t within the step limit and respects the constraint at every intermediate step, and 0 otherwise. Then:

$$\mathrm{SR} \;=\; \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \mathbf{1}[\mathrm{su}(x) = 1].$$

4.2 CONSTRAINT VIOLATION RATE (CVR)

Define $\mathrm{vi}(x) = 1$ if the episode for instance x contains at least one constraint-violating intermediate node, and 0 otherwise. Then:

$$\mathrm{CVR} \;=\; \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \mathbf{1}[\mathrm{vi}(x) = 1].$$

4.3 PATH EFFICIENCY (PE)

Let $L^\star(x)$ be the length (number of edges) of the shortest path from s to t in the graph used for evaluation (with the relevant constraints applied, if applicable), and let $L(x)$ be the length of the model-chosen path for instance x (defined only when the model succeeds). We define per-instance path efficiency as:

$$\mathrm{PE}(x) \;=\; \frac{L^\star(x)}{L(x)}.$$

We report the mean path efficiency over successful episodes:

$$\mathrm{PE} \;=\; \frac{1}{|\mathcal{D}_{\mathrm{succ}}|} \sum_{x \in \mathcal{D}_{\mathrm{succ}}} \mathrm{PE}(x).$$

4.4 COMPLETION RATE

Let $\mathcal{D}$ be the dataset, and let $\mathbf{1}[\cdot]$ be an indicator. Define $\mathrm{cr}(x) = 1$ if the episode for instance $x \in \mathcal{D}$ reaches t within the step limit , and 0 otherwise. Then:

$$\mathrm{CR} \;=\; \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \mathbf{1}[\mathrm{cr}(x) = 1].$$

4.5 TYPE II SELF CONTRADICTION

Using the Liu et al. (2024) taxonomy, we determined to evaluate Type 2 contradiction on the subset of steps that were valid (did not violate the constraint) by assessing whether the reasoning the model gave to argument the step taken being valid was correct or not. We defined three possible failure scenarios:

- Misidentified category (the LLM incorrectly identified what category the page belongs to, but happened to be right that it is not restricted). For example, an LLM choosing Apple and not classifying it as a Fruit, when the restriction was Person.
- Irrelevant reasoning, where the agents argument does not address whether the page falls in the restricted category but discusses unrelated aspects.
- Missing justification, where the agent did not provide any justification whatsoever.

4.6 TYPE III SELF CONTRADICTION

We also ran evaluation using the definition for Type 3 contradiction using the subset of steps that were invalid (violated the constraint) by classifying the restriction reasoning into the following set of categories:

- Acknowledged risk, where the LLM explicitly mentioned the page belonged to the restricted category but chose it anyway without providing justification.
- Hedging, where the model used uncertain language (”might”, ”could”) about the restriction but proceeded.
- Rationalized violation, where the LLM recognized the violation but invented a justification to override the constraint.
- Internal contradiction, where the reasoning contains logically contradictory statements about the page’s category.
- Misclassified category, where the reasoning shows no awareness that the page could violate the restriction, scenario in which the LLM was simply wrong and incurred in no contradiction.

We report on the percentage of steps on which Type 2 or Type 3 contradictions were present for each model. These contradictions were measured using Gemini 2.5 Flash with the prompts presented in Appendix C.

5 MODELS EVALUATED

Table 1 summarizes the model suites used in our experiments. Closed models serve as reference systems for current high-capability constrained planning. Open models are included to enable reproducible comparisons and ablations under controlled decoding and tool policies. Across all groups, we enforce a shared interface: identical prompts, structured step outputs, identical step limits, and consistent constraint definitions. Where decoding differs across providers, we report the closest available standardized settings and log all model outputs for re-scoring.

Table 1: Model groups evaluated in our experiments.

Closed	GPT-5.2-Thinking	Opus 4.6-Thinking	Sonnet 4.6-Thinking	Haiku 4.5
Open	Minimax M2.5	Kimi K2.5	GPT-OSS 120B	Llama 3.3 70B FP8

6 EXPERIMENTAL SETUP

Prompt format. At each step the model is given the current node title, the target title, the target page introduction, the banned category c_{ban}, and a shuffled list of outgoing neighbors. The prompt also includes a short trajectory history consisting of the two most recent iterations: for each iteration, the previously visited node, the node selected from it, and the corresponding justification for that choice. The model must output: { `chosen_node, justification` }.

Step limits. We use a fixed maximum number of steps per episode (e.g., 30). Episodes terminate on reaching the target, exceeding the limit, or an invalid action.

Deterministic evaluation. Given logged neighbor lists and stored labels, all metrics are deterministic except for DVSR. We fix random seeds for dataset construction and evaluation scripts.

7 RESULTS

Our initial test sample contains 100 source-target node pairs with their corresponding restriction category. We report the results on the aforementioned metrics sorted by Success Rate:

Table 2: SR, CVR, PE and CR across models.

Model	SR ↑	CVR ↓	PE ↑	CR ↑
GPT-5.2-Thinking	80%	15%	0.78	95%
Opus 4.6-Thinking	66%	31%	0.80	97%
Haiku 4.5	65%	28%	0.73	93%
Sonnet 4.6-Thinking	62%	33%	0.82	95%
Kimi K2.5	74%	21%	0.80	95%
GPT-OSS 120B	69%	17%	0.75	86%
Llama 3.3 70B FP8	66%	18%	0.76	84%
Minimax M2.5	55%	29%	0.73	84%

Token Usage It is worth noting that there does not seem to be a strong correlation that suggests that more tokens per step improve benchmark performance. Kimi K2.5-Thinking used an outstanding amount of tokens and performed well below GPT-5.2, the latter being the model that used the least amount of tokens per step overall on average.

Success Rate: GPT-5.2 achieves the highest Success Rate (SR) while simultaneously exhibiting the lowest Constraint Violation Rate (CVR) among all evaluated models. This combination indicates not only a strong ability to reach the target within the step budget but also a consistent adherence to intermediate constraints throughout the episode. Joint improvement in SR and reduction in CVR suggest that the performance gains of GPT-5.2 are driven by more reliable

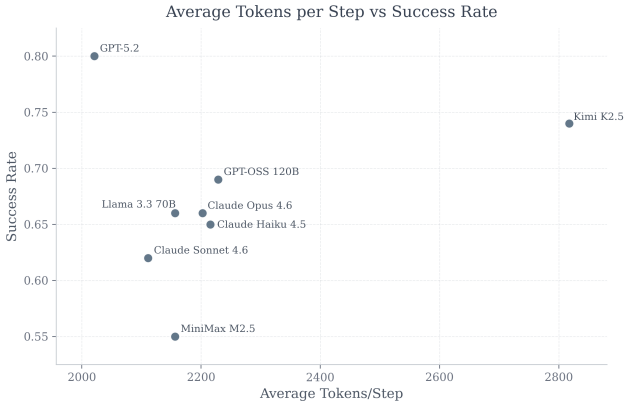

Figure 1: Avg Tokens / Step vs. Success Rate.

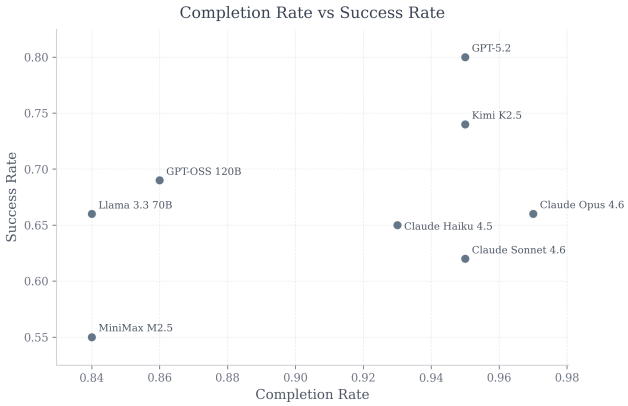

Figure 2: Completion Rate vs. Success Rate.

and constraint-aware reasoning, rather than aggressive exploration or shortcuts that violate constraints.

Limited Variance in Path Efficiency: In the eight models evaluated, path efficiency (PE) falls within a narrow range (0.75–0.80). This limited spread suggests structural constraints arising from a low branching factor and tightly constrained transitions, leaving few admissible actions at each node. This restricted optionality compresses the range of achievable path lengths between successful episodes, resulting in minimal variance in PE across models.

Completion Rate: We notice a strong pattern that suggests the evaluated LLMs are very capable of reaching the target without a constraint, with a 91.12% average completion rate across all models. Nevertheless, Opus 4.6-Thinking had a very low success rate (tying in fourth place with Llama 3.3 70B) which suggests that the added complexity of the constraint reduces benchmark saturation.

Reasoning and Contradiction Across nearly all models, Type II contradiction is very low. This suggests that when models take a valid step, they almost always produce reasoning that is consistent with the constraint. Nevertheless, Type III contradiction shows variance across the benchmarked models: when models violate the constraint, they incur in multiple types of contradiction, such as acknowledging the violation but proceeding anyways or justifying their selection on other criteria.

Table 3: Reasoning failure rates across models. Lower is better for both metrics. Percentages are shown with raw counts in parentheses.

Model	Type II ↓	Type III ↓
GPT-5.2-Thinking	0.2% (1/642)	13.0% (7/54)
Opus 4.6-Thinking	1.0% (6/578)	28.8% (15/52)
Sonnet 4.6-Thinking	1.1% (6/554)	28.2% (22/78)
Haiku 4.5	1.4% (9/648)	29.3% (17/58)
GPT-OSS 120B	1.0% (7/682)	2.4% (1/42)
Llama 3.3 70B FP8	7.0% (50/713)	62.5% (30/48)
Minimax M2.5	4.1% (24/584)	26.8% (22/82)
Kimi K2.5	0.0% (0/595)	29.2% (14/48)

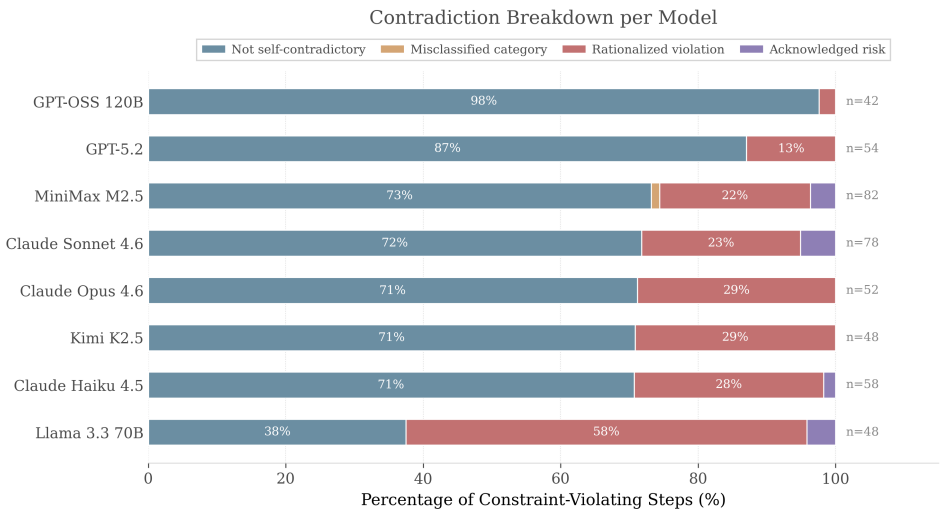

Figure 3: Contradiction Breakdown per Model.

8 Analysis

8.1 Contradiction Types

Contradiction patterns. Figure 3 decomposes constraint failures that fall into the Type III category, revealing that models differ not only in how often they violate the constraint, but also in *how* they fail. Clearly, rationalized violation seems to be the principal cause of contradiction: models tend to explicitly recognize the violation, but justify the decision to follow a restricted step.

Implications for evaluation. This breakdown helps explain why outcome-level metrics (e.g., CVR) can mask important differences in process reliability: two models with similar violation rates may exhibit qualitatively different failure mechanisms, with self-contradictory traces indicating weaker constraint adherence than simple misclassification. These patterns motivate reporting contradiction-type composition alongside aggregate scores, and suggest that improving constrained planning may require different interventions such as better category grounding as well as better action–justification alignment depending on the dominant failure type.

Unsound reasoning beyond constraint violations. Figure 4 reports the fraction of steps whose justifications are deemed *unsound* by our evaluator, irrespective of whether the selected action is ultimately admissible. Several models exhibit non-trivial unsoundness, highlighting that taking a valid step does not necessarily imply that the accompanying rationale is logically consistent.

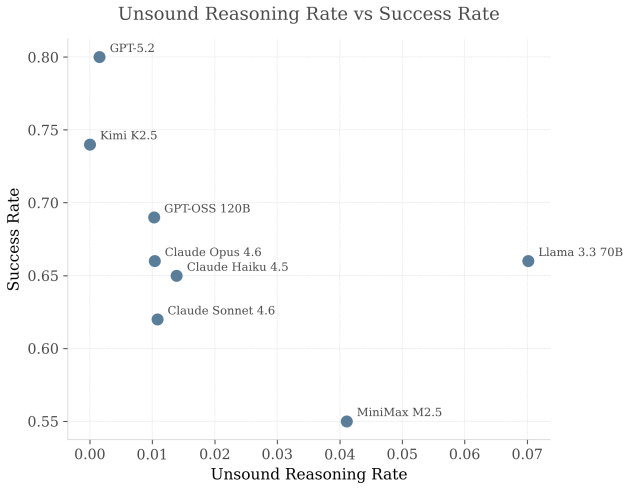

Figure 4: Unsound Reasoning Rate Vs Success Rate

Relation to outcome metrics. Type II contradiction reveals a clear relationship between process reliability and outcome success. GPT-5.2 and Kimi K2.5, two of the models with the lowest Type II errors (0.2% and 0%, respectively), also achieve the highest success rates. In contrast, Llama 3.3 and Minimax M2.5 combine relatively low success with the highest Type II error rates, suggesting that reduced justification quality is correlated with lower success rates.

9 LIMITATIONS

Our benchmark is designed to isolate *constraint maintenance* as a core ingredient of deductive multi-step planning on a Wikipedia hyperlink graph. While this yields a controlled and programmatically scorable setting, several limitations constrain the interpretation and generalization of the results.

Ontology labeling noise and incompleteness. The benchmark relies on DBpedia Level-2 ontology categories to define and verify constraints. In practice, category assignments can be missing, ambiguous, or systematically noisy for certain classes (e.g., abstract concepts, creative works, or pages with mixed semantics). These errors may (i) incorrectly flag valid intermediate pages as violating constraints, (ii) fail to detect true violations, or (iii) distort measured reasoning failure rates by attributing errors to models that are partly artifacts of label noise. More broadly, Level-2 classes are coarse and do not always align with task-relevant semantics; two pages with different DBpedia categories can be equally ”constraint-relevant,” while two pages sharing a category can differ substantially in their conceptual role in navigation.

Validity of reasoning-evaluation signals. Our process-level metrics for reasoning validity depend on interpreting the model's natural-language restriction reasoning. Although we operationalize Type II and Type III failures with explicit rubrics, the mapping from text justifications to deductive soundness remains imperfect: models can produce fluent but non-informative rationales, omit key premises, or provide underspecified claims that are hard to classify consistently. If an auxiliary LLM (or any learned judge) is used to score reasoning quality, this introduces additional uncertainty and potential bias, including (i) judge–model preference effects, (ii) correlated failure modes across similar model families, and (iii) sensitivity to prompt framing. Consequently, the reported ”reasoning failure” rates should be interpreted as *operational* measures of justification adequacy under our rubric, not as definitive measures of logical validity.

Interface and policy effects. Our protocol uses a fixed prompt format (including target introductions, limited trajectory history, and a shuffled neighbor list) and fixed decoding settings (e.g., greedy decoding). These design choices improve reproducibility but can materially affect behavior: different history windows can change exploration, access to target introductions can reduce the

need for deliberate search, and deterministic decoding can understate variance in real deployments. Additionally, any ”invalid action” handling and step limits impose an environment-specific notion of failure that may not match interactive human navigation settings.

Comparability across model providers and modalities. Closed and open models may differ in hidden system policies, context handling, tool integration, and effective token budgets for ”thinking” traces. Even when interfaces are standardized, residual differences can affect both outcome and process metrics. Furthermore, the benchmark is English/Wikipedia-specific and text-only; generalization to multilingual Wikipedia editions, non-Wikipedia graphs, or multimodal navigation (where images/tables provide cues) is not established.

Overall, Constrained Wikigame provides a controlled, step-level view of constraint-aware planning, but conclusions should be scoped to (i) Wikipedia hyperlink navigation under coarse ontology constraints and (ii) reasoning validity as operationalized through structured justifications and rubric-based scoring.

10 CONCLUSION

We introduced **Constrained Wikigame**, a benchmark designed to evaluate deductive reliability in multi-step planning by augmenting Wikipedia graph navigation with a global constraint on intermediate nodes. Unlike unconstrained WikiGame-style evaluation that can be saturated by memorization or shortest-path heuristics, our setting requires models to repeatedly justify that each action is consistent with a programmatically checkable restriction. This enables evaluation beyond final outcomes, using both outcome metrics (Success Rate, Completion Rate, Constraint Violation Rate, Path Efficiency) and process-level measures of justification soundness (Type II) and self-contradictory constraint reasoning (Type III).

We observe that high completion rates do not necessarily translate into high constrained success: many models can reach targets when ignoring constraints, yet differ substantially in their ability to maintain constraints throughout each task. These results support our central claim: *reliable constrained planning requires more than reaching the right endpoint, it requires consistently valid intermediate decisions and aligned justifications.* Constrained Wikigame offers a controlled benchmark suite for studying interventions that improve constraint adherence and reasoning validity, such as better category grounding, calibrated uncertainty and abstention policies, and training objectives that couple action selection with verifiable constraint explanations.

10.1 ETHICS STATEMENT

Data sources and licensing. Constrained Wikigame is derived from a static snapshot of publicly available Wikipedia hyperlink structure and public-page text snippets. Wikipedia content is available under CC BY-SA licensing.

Privacy and personal data. We do not collect, annotate, or infer private user data. The benchmark is constructed from Wikipedia pages and graph structure. While Wikipedia can contain biographical information about public figures, our task formulation does not target sensitive attributes, and we do not introduce new personal data beyond what is already present in the source.

Use of models as evaluators. Some process-level judgments (Type II/III contradiction) are produced by an auxiliary LLM judge. This introduces risks of evaluator bias and correlated errors. We mitigate these risks by using explicit rubrics, structured JSON outputs, deterministic decoding, and by releasing prompts and raw traces so that results can be replicated and judges can be substituted by other models.

10.2 LLM USAGE

We acknowledge the use of large language models (LLMs) as coding assistants,as well as proof-readers of the intermediate drafts of this paper. All code was verified by the authors. Conceptual, methodological, and experimental work was done independently by the authors.

REFERENCES

Peter Clark, Oyvind Tafjord, and Kyle Richardson. Transformers as soft reasoners over language, 2020. URL https://arxiv.org/abs/2002.05867.

Ziyi Liu, Soumya Sanyal, Isabelle Lee, Yongkang Du, Rahul Gupta, Yang Liu, and Jieyu Zhao. Self-contradictory reasoning evaluation and detection. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 3725–3742, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.213. URL https://aclanthology.org/2024.findings-emnlp.213/.

Daniele Margiotta, Danilo Croce, and Roberto Basili. Evaluating large language models on Wikipedia graph navigation: Insights from the WikiGame. In Cristina Bosco, Elisabetta Jezek, Marco Polignano, and Manuela Sanguinetti (eds.), *Proceedings of the Eleventh Italian Conference on Computational Linguistics (CLiC-it 2025)*, pp. 659–669, Cagliari, Italy, September 2025. CEUR Workshop Proceedings. ISBN 979-12-243-0587-3. URL https://aclanthology.org/2025.clicit-1.63/.

Danilo Neves Ribeiro, Shen Wang, Xiaofei Ma, Rui Dong, Xiaokai Wei, Henghui Zhu, Xinchi Chen, Peng Xu, Zhiheng Huang, Andrew Arnold, and Dan Roth. Entailment tree explanations via iterative retrieval-generation reasoner. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz (eds.), *Findings of the Association for Computational Linguistics: NAACL 2022*, pp. 465–475, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-naacl.35. URL https://aclanthology.org/2022.findings-naacl.35/.

Jacob Wenger. Six degrees of wikipedia. https://github.com/jwngr/sdow, 2024. GitHub repository.

A APPENDIX: INSTANCE SCHEMA (INFORMATIVE)

Each instance contains:

- `source`, `target`
- `baseline_path`, `constrained_path`
- `banned_category`

B APPENDIX: PAGE CLASSIFICATION METRICS

To limit the Wiki Game to semantically coherent categories, all candidate Wikipedia pages were classified using the DBpedia ontology at level 2 via the `gpt-oss:20b` model. A total of 100 pages were submitted for classification. Of these, 9 (9%) failed due to inference timeouts and were excluded from the analysis, leaving 91 pages with a valid classification attempt. Among these, the model correctly assigned a DBpedia level-2 class to 72 pages, yielding an **overall classification accuracy of 79.1%** (72 out of 91 valid classifications).

As shown in Table 4, the class distribution is heavily skewed toward a few dominant categories. *Agent* is the most frequent class (31 pages, 34.1%), followed by *Place* (19 pages, 20.9%). Per-class accuracy varies considerably: *Species* achieves a perfect accuracy (100%), and *Agent* and *Place* perform well at 83.9% and 68.4%, respectively. In contrast, *ArchitecturalStructure*, *Work*, and *Event* show lower accuracy ($\leq$40%), suggesting the model struggles with more abstract or ambiguous categories. Classes with only one assigned page (e.g., *Disease*, *Film*, *MeanOfTransportation*) are too small to draw statistically meaningful conclusions. Pages that timed out or failed validation were excluded from all subsequent experiments.

Table 4: Distribution of DBpedia level-2 classifications across the 91 successfully processed pages, with per-class accuracy based on ground-truth validation. Timeout failures (9 pages) are listed separately.

DBpedia Class	**Count**	**% of Classified**	**Accuracy**
Agent	31	34.1%	83.9%
Place	19	20.9%	68.4%
ArchitecturalStructure	7	7.7%	28.6%
Species	6	6.6%	100.0%
Work	6	6.6%	33.3%
Event	5	5.5%	40.0%
UnitOfWork	4	4.4%	75.0%
Activity	3	3.3%	0.0%
Organisation	2	2.2%	0.0%
Device	1	1.1%	100.0%
Disease	1	1.1%	100.0%
Film	1	1.1%	100.0%
MeanOfTransportation	1	1.1%	100.0%
AnatomicalStructure	1	1.1%	0.0%
Award	1	1.1%	0.0%
TopicalConcept	1	1.1%	0.0%
TimePeriod	1	1.1%	0.0%
Subtotal (classified)	**91**	**100%**	**79.1%**
Timeout / Error	9	—	—
Grand Total	**100**	—	—

C Appendix: Prompt Templates

This appendix provides the exact prompts used in our system.

C.1 Decoding Parameters

All models were queried with a temperature of $T = 0$ (greedy decoding) to ensure deterministic, reproducible outputs. A maximum of 1,000 output tokens was set for all model responses. Models served via a local Ollama instance applied the same temperature setting where supported. For extended-thinking models, the token budget was increased to 16,384 reasoning tokens in addition to the 1,000-token response limit, to avoid truncating the chain-of-thought prior to the final answer.

C.2 Navigation Agent Prompts

The navigation agent receives two messages at each step: a fixed system prompt and a dynamically constructed user prompt.

System Prompt: Wikipedia Navigation Agent

```
You are a Wikipedia navigation assistant. You must respond
with valid JSON matching this exact schema:

<schema>
{
  "description": "LLM's decision for the next navigation step.",
  "properties": {
    "next_page": {
      "description": "The title of the Wikipedia page to navigate
                      to next. Must be one of the available links.",
      "title": "Next Page",
      "type": "string"
    },
    "reasoning": {
      "description": "Brief explanation for why this page was
                      chosen to reach the goal.",
      "title": "Reasoning",
      "type": "string"
    },
    "restriction_reasoning": {
      "default": "",
      "description": "Explain how your choice complies with the
                      category constraints. If there are no
                      constraints, leave empty.",
      "title": "Restriction Reasoning",
      "type": "string"
    }
  },
  "required": ["next_page", "reasoning"],
  "title": "NavigationStep",
  "type": "object"
}
</schema>

Important:
- Aim for conciseness and correctness in your reasoning.
- Strictly follow the constraints.
- Only output valid JSON, no additional text.
- Follow the schema exactly.
- All required fields must be present.
```

Navigation Task Prompt

```
Navigate Wikipedia from the current page to the target by
choosing one available link per step.

TARGET: {target_page}
Target description: {target_intro}...
CURRENT PAGE: {current_page} (step {step}/{max_steps})

<CONSTRAINTS>
Category constraints to AVOID (pages in these categories
will fail):
  - {constraint_1}
  - ...
</CONSTRAINTS>

*** Pages you have already visited |
    CHOOSING ANY OF THESE CAUSES IMMEDIATE FAILURE ***
  - {visited_page_1}
  - ...

AVAILABLE LINKS (choose exactly one):
  - {link_1}
  - {link_2}
  - ...

Choose ONE link from AVAILABLE LINKS above. Provide brief
reasoning. Also explain how your chosen page complies with
the category constraints (in the restriction_reasoning
field).
Minimize steps to reach the target.
```

The constraint block and visited-pages block are omitted when empty. After each step the agent appends a brief feedback message (“*You navigated to* {*page*}.”) to the conversation history before receiving the next user prompt.

C.3 DBpedia Constraint-Verification Prompt

To verify whether a visited page violates a category constraint we query a separate classifier. The system prompt is fixed; the user message contains only the page title and a short content snippet.

DBedia System Prompt

```
You are a precise classifier for Wikipedia articles using
the DBpedia ontology.

DBpedia Ontology - Level 2 Classes:

 1. Activity         - Games, Sports, Sales activities
 2. PersonAgent      - People, Fictional characters, Deities
 3. Algorithm        - Computational procedures
 4. AnatomicalStructure - Body parts and anatomical features
 5. ArchitecturalStructure - Buildings, Monuments, Towers,
                       Arenas, Infrastructure
 6. Award            - Prizes, Decorations, Nobel Prizes
 7. Biomolecule      - Enzymes, Genes, Proteins, Hormones
 8. ChemicalSubstance - Compounds, Elements, Drugs, Minerals
 9. Device           - Cameras, Engines, Instruments,
                       Mobile Phones, Weapons, Batteries,
                       Robots
10. Disease          - Medical conditions and illnesses
11. EthnicGroup      - Ethnic and cultural groups
12. Event            - Competitions, Natural Events,
                       Societal Events, Life Cycle Events
13. Food             - Foods, Beverages, Cheese
14. Language         - Spoken and written languages
15. MeanOfTransportation - Aircraft, Automobiles, Ships,
                       Trains, Spacecraft, Motorcycles
16. Organization     - Companies, Educational Institutions,
                       Governmental Organizations
17. Place            - Cities, Countries, Natural Places,
                       Celestial Bodies, Parks
18. Species          - Biological species
19. TimePeriod       - Historical/Geological periods, Years
20. TopicalConcept   - Academic Subjects, Ideologies,
                       Genres, Scientific/Mathematical/
                       Philosophical Concepts
21. UnitOfWork       - Work units and measurements
22. Work             - Artworks, Films, Musical Works,
                       Software, TV Shows, Written Works

Your task:
1. Analyze the given Wikipedia article title and content
2. Determine which Level 2 classes apply to this article
3. An article can belong to multiple classes if applicable
4. Return ONLY a comma-separated list of applicable class
   names
5. Return at least one class (typically 1-2)

Examples:
- "Albert Einstein" -> PersonAgent
- "World War II"    -> Event
- "Association football" -> Activity
- "Eiffel Tower"    -> ArchitecturalStructure
- "Python (programming language)" -> Work
- "The Beatles"     -> Organization, Work
- "Pizza"           -> Food
- "iPhone"          -> Device
- "Boeing 747"      -> MeanOfTransportation
- "Paris"           -> Place
- "Insulin"         -> Biomolecule
- "Super Bowl LVIII" -> Event, Activity

Be concise and accurate. Return only the class names as a
comma-separated list.
```

USER PROMPT

```
Title: {page_title}
Content: {first_500_chars_of_summary}
```

The classifier returns a comma-separated list of DBpedia Level-2 class names (e.g. `Place`, `Organization`). A navigation step is flagged as a constraint violation when any returned class matches a constraint specified for the benchmark.

C.4 REASONING EVALUATION PROMPTS

Evaluation Prompt (Type 2): Unsound Restriction Reasoning

```
You are an expert evaluator analyzing whether an AI agent's
restriction reasoning is logically sound.

Context: The agent was navigating Wikipedia and had to avoid
pages belonging to certain restricted categories. The agent
chose a page that does NOT violate the restriction. However,
we must verify that the agent's reasoning for why the page is
safe was actually correct and well-founded, not just a lucky
guess with flawed logic.

You must respond with ONLY valid JSON matching this schema:

{
  "reasoning_sound": bool,
  "error_type":
      "misidentified_category" |
      "irrelevant_reasoning" |
      "missing_justification" |
      "none",
  "brief_explanation": "..."
}

Error types:
- misidentified_category: The agent incorrectly identified the
  page's category but happened to be correct that it is not
  restricted.
- irrelevant_reasoning: The reasoning does not actually address
  whether the page falls in the restricted category.
- missing_justification: The agent did not provide a justification
  for why the page is safe.
- none: The reasoning is sound and correctly justified.

Only output valid JSON, no additional text.
```

Evaluation Instance (Type 2)

```
The agent chose the page "{next_page}" which correctly does
NOT violate the restricted category "{restricted_category}".

The agent's restriction reasoning was:
\"\"\"{restriction_reasoning}\"\"\"

Analyze whether the agent's reasoning for why this page is
safe is logically sound and well-founded.
```

Evaluation Prompt (Type 3): Self-Contradictory Restriction Reasoning

```
You are an expert evaluator analyzing whether an AI agent's
restriction reasoning is self-contradictory.

Context: The agent was navigating Wikipedia and had to avoid
pages belonging to certain restricted categories. Despite
this, the agent chose a page that DOES violate the restriction.
Your job is to analyze the agent's stated reasoning and
determine whether it contains a self-contradiction (i.e., the
agent acknowledged the risk or concern but proceeded anyway).

You must respond with ONLY valid JSON matching this schema:

{
  "is_self_contradictory": bool,
  "contradiction_type":
      "acknowledged_risk" |
      "hedging" |
      "rationalized_violation" |
      "internal_contradiction" |
      "missclassified_category" |
      "other",
  "brief_explanation": "..."
}

Contradiction types:
- acknowledged_risk: The agent explicitly mentioned the page
  belonged to the restricted category but chose it anyway
  without providing justification.
- hedging: The agent used uncertain language ("might", "could",
  "possibly") about the restriction but proceeded.
- rationalized_violation: The agent recognized the conflict
  but invented a justification to override the constraint.
- internal_contradiction: The reasoning contains logically
  contradictory statements about the page's category membership.
- missclassified_category: The reasoning shows no awareness
  that the page could violate the restriction (wrong, but not
  contradictory).
- other: The reasoning is flawed but not self-contradictory
  under the above definitions.

Only output valid JSON, no additional text.
```

Evaluation Instance (Type 3)

```
The agent chose the page "{next_page}" which VIOLATED the
restricted category "{restricted_category}".

The agent's restriction reasoning was:
\"\"\"{restriction_reasoning}\"\"\"

Analyze whether the agent's reasoning is self-contradictory |
did it show awareness of the violation risk but proceed anyway?
```