Can LLMs Reason Structurally? Benchmarking via the Lens of Data Structures

Ellen Vitercik

ICML'26

Yu He

Yingxi Li

Colin White

Stanford University

Stanford University

Isolating LLM algorithmic reasoning primitives

- LLMs increasingly handle **complex, multi-step** real-world decision tasks
  - Planning, analysis, and structured problem-solving
- These tasks require more than language generation:
  - Models must **execute operations, update state, compose steps**
- But current benchmarks rarely isolate **algorithmic reasoning primitives**

  [See, e.g., "Position: We Need An Algorithmic Understanding of Generative AI," Eberle et al., ICML'25]
- Can LLMs **reason structurally**, or just recognize familiar patterns/templates?

Structural reasoning

- **Structural reasoning:** Ability to
  - Understand relationships like order, hierarchy, connectivity, composition
  - Manipulate objects according to these relationships

- Multi-stage decision-making tasks depend on **reliable structural updates**
  - Construct, manipulate, and compose relationships

- Real-world systems use the same abstractions
  - Trip planning, scheduling, …

Benchmarks don’t isolate structural reasoning

- Existing benchmarks are domain-specific, atomic skills may be **entangled**
  - E.g., math/STEM [Liu et al., NeurIPS’23, EMNLP’23; Hendrycks et al., NeurIPS’21]
- Coding benchmarks measure **generation**, not **inherent reasoning**
  - Interpreters offload computation; online code risks contamination
  - **Our focus: inherent** reasoning without code
    - Tool use can obscure the model’s inherent reasoning ability
    - In line with recent initiatives like Google & OpenAI performance in IMO competitions
      - Prohibit code/proof assistance, emphasizing end-to-end reasoning
- Algorithmic benchmarks still diagnose too **coarsely** [Markeeva et al., DMLR’26]
  - Complex responses obscure algorithmic primitives
  - Failures difficult to localize precisely

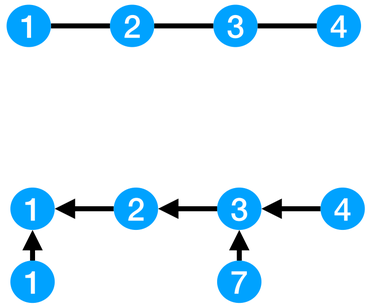
Data Structure Reasoning (DSR) Benchmark

TCS lens for evaluation: use primitives to study generalization, reliability

- We introduce **DSR-Bench** for structural reasoning evaluation
- **Data structures** are fundamental algorithmic building blocks
  - They support **composed, multi-step reasoning**
- Their operations are **interpretable** and **deterministic**
  - Outputs can be verified automatically
- Data structures **isolate structural relationships** cleanly

DSR-Bench: Main suite

Goal: cover distinct structural abstractions



- **Linear** and **temporal** capture ordered sequences
  - E.g., arrays, stacks, queues
- **Associative** structures model key-based retrieval
  - E.g., hash tables
- **Hierarchical** structures test multi-level state maintenance
  - E.g., BSTs, red-black trees
- **Network** and **hybrid** structures combine dependencies
  - E.g., graphs

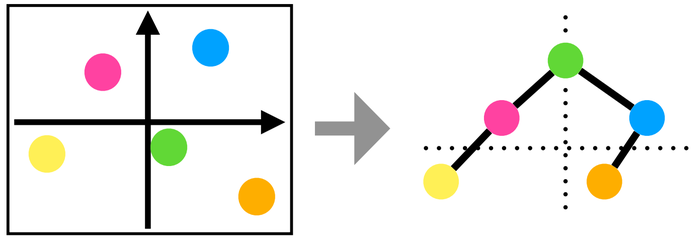
DSR-Bench

Benchmark overview

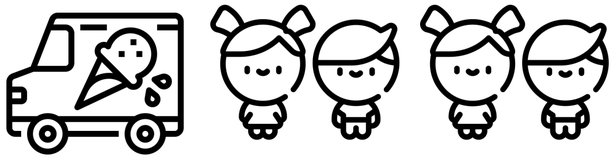
- We introduce **DSR-Bench** for structural reasoning evaluation
  - 20 data structures; 35 operations; 4,140 instances
- Organizes tasks **hierarchically** by reasoning complexity
  - Simple tasks support complex tasks
- Uses **synthetic generation** and **deterministic scoring**
  - Minimizes contamination; avoids subjective judging
- Produces interpretable **failure-mode diagnostics**
  - Pinpoints where structural reasoning breaks down

DSR-Bench: Auxiliary probes

DSR-Bench spatial



DSR-Bench realistic



An ice cream truck rolled in. Children began to form a line, each newcomer taking their place at the end… **Q**: What is the order of the kids in line?

DSR-Bench code

Given an array [1, 2, 3, 4].
Q: Write code to find the value stored in index 2.

- **Spatial:** tests high-dimensional data across dimensions and distributions
- **Realistic:** embeds operations in context-rich narrative scenarios
- **Code:** compares direct reasoning, self-generated code, external execution

Outline

1. Introduction and motivation
2. **Benchmark details**
3. Results and failure modes
4. Conclusions and future directions

Operations and difficulty levels

- Each structure includes multiple operation types:
  - **Inspection** asks models to read existing structure
    - Examples: Access, traversal, search, depth
  - **Manipulation** asks models to update state: insert, delete, remove
  - **Compound** tasks require sequential updates
    - Examples: insert, insert, delete
- Difficulty depends on task **length**: short, medium, and long inputs

Example prompt

- A queue maintains a first-in, first-out (FIFO) order
  - Items are added at one end and removed from the other
- There are two operations:
  - (`enqueue, k`) adds `k` to the back
  - (`dequeue`) removes the front
- You are given an empty queue initially
- What is the final queue after: (`enqueue, 49`), (`enqueue, 85`), (`dequeue`), …?

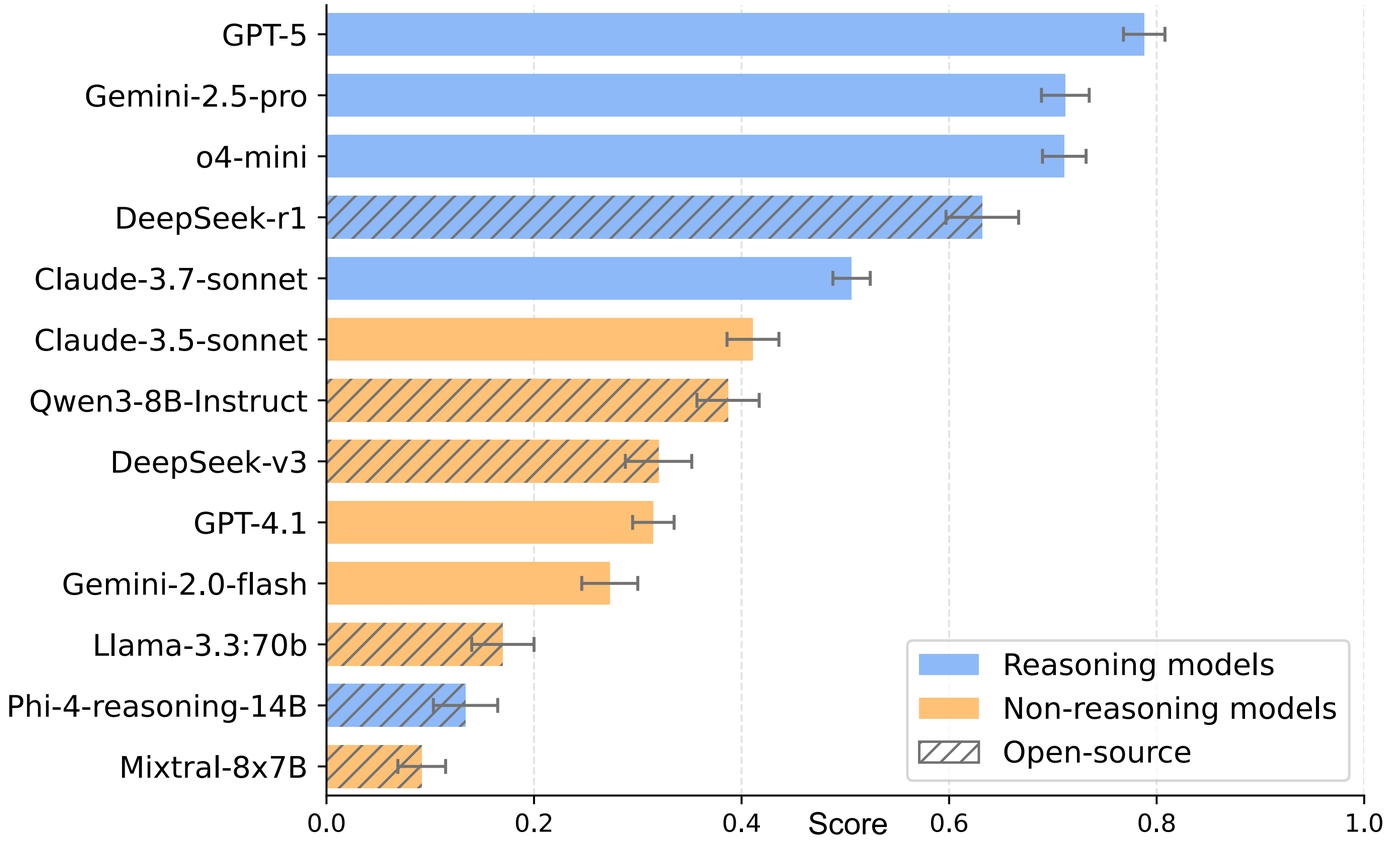
Prompt and evaluation design

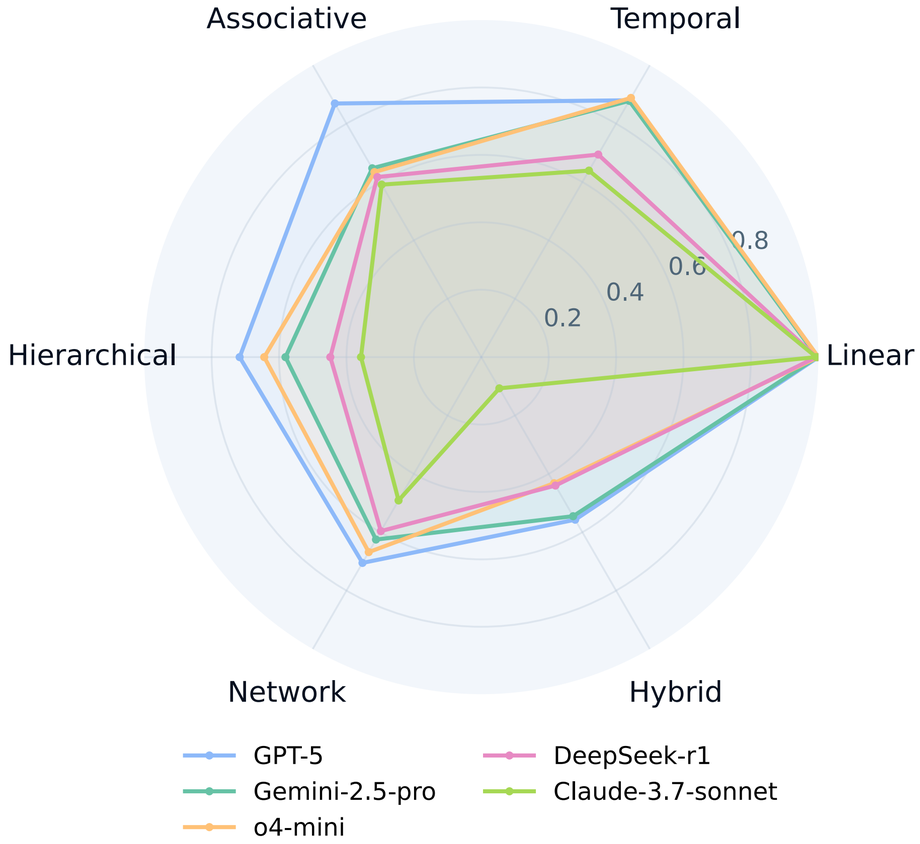
- Operations are specified to remove ambiguity:
  - Hashes, tie-breaking, and formats are explicit
- Synthetic data populates reusable prompt templates
- Programmatic implementations generate ground-truth outputs
- Structured outputs enforce machine-readable JSON answers
- Binary scoring compares against one deterministic solution
- Tests five prompting strategies: Stepwise, 0-CoT, CoT, 3-shot, None

Outline

1. Introduction and motivation
2. Benchmark details
3. **Results and failure modes**
4. Conclusions and future directions

Aggregated scores





Where composition breaks

Multi-attribute and multi-hop failure analysis

- **Composition** fails when state has multiple attributes
  - E.g., priority queues add priorities and insertion order
- **Multi-hop** structures stress memory across levels
  - E.g., red-black trees require ancestral balance updates

Structure	GPT-5	Gemini-2.5-Pro	o4-mini	DeepSeek-R1	Claude-3.7-Sonnet	Take-away
Queue	1.00	1.00	1.00	0.98	1.00	FIFO state is mostly solved
Priority queue	0.52	0.51	0.55	0.65	0.28	Add priority/tie-breaking: accuracy drops
BST	1.00	0.97	0.86	0.73	0.64	Basic tree structure is easier
RB tree	0.76	0.49	0.65	0.62	0.30	Balance constraints are harder

Length generalization is still weak

Take-away: Many models degrade sharply with input length

Short = 5–10; Medium = 11–20; Long = 21–30 inputs/operations

Structure / operation	GPT-5	Gemini-2.5-Pro	DeepSeek-R1	Claude-3.7-Sonnet
Queue compound	1.00→1.00→1.00	1.00→1.00→1.00	1.00→1.00→0.97	1.00→0.93→0.98
Priority Queue compound	0.84→0.44→0.28	0.89→0.41→0.23	0.92→0.54→0.48	0.70→0.11→0.04
RB Tree compound	0.91→0.67→0.49	0.91→0.41→0.13	0.91→0.37→0.03	0.57→0.03→0.00
Graph DFS	0.93→1.00→0.96	1.00→0.81→0.14	0.80→0.58→0.22	0.50→0.11→0.00

Challenge subset

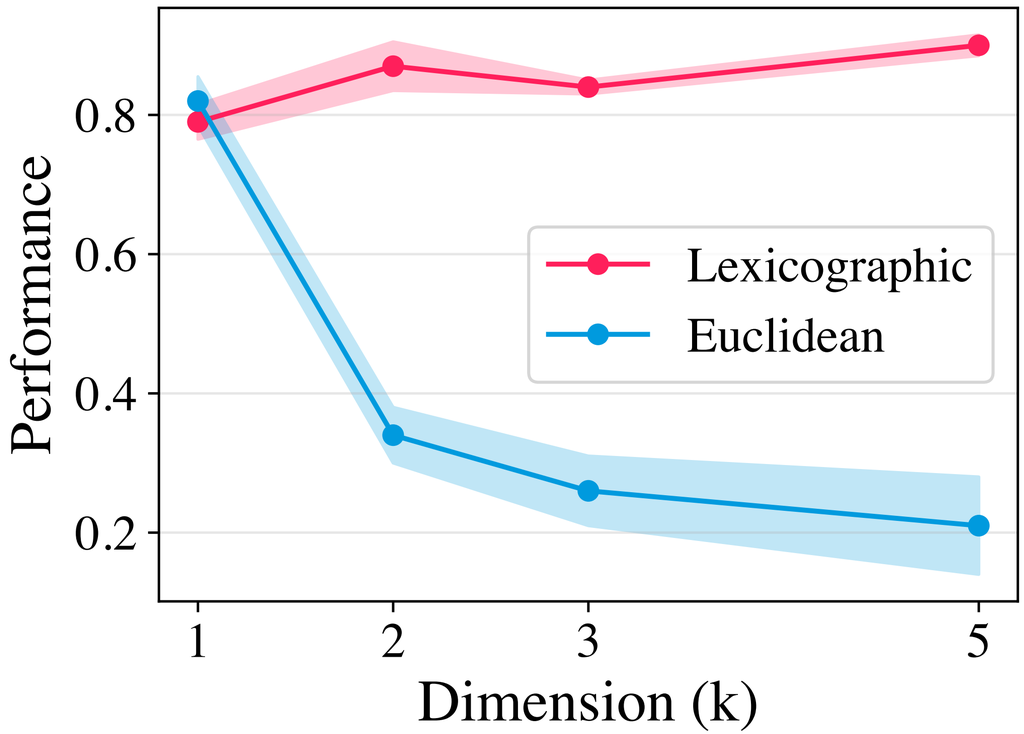
Stress-tests complex structural reasoning

Challenge task	GPT-5	Gemini-2.5-Pro	DeepSeek-R1	Claude-3.7-Sonnet	Take-away
Overall score	0.46	0.30	0.21	0.10	No model reaches 0.5 overall
B+ Tree	0.98	0.94	0.21	0.13	Some complex structures partly solved
K-D Tree	0.67	0.16	0.01	0.00	Spatial hierarchy remains brittle
Geom Graph	0.19	0.02	0.01	0.00	Spatial network construction collapses
DAWG	0.00	0.01	0.00	0.00	Hybrid trie-graph composition collapses

Specifications can lose to learned priors

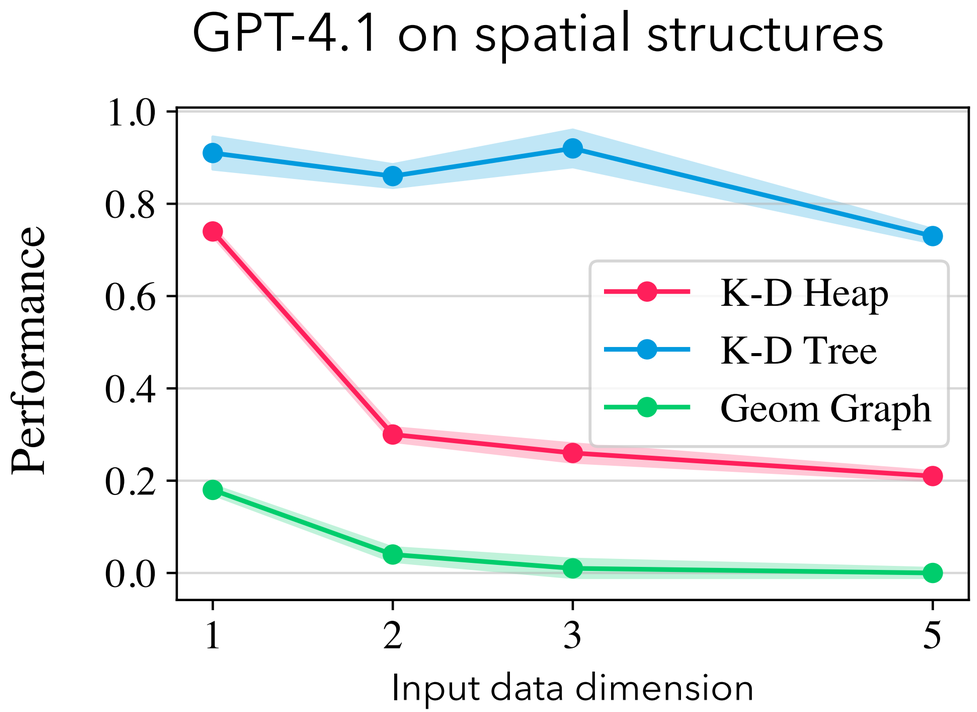
Take-away: Explicit constraints don’t override learned defaults

o4-mini on K-D Heap with different tie-breaking rules



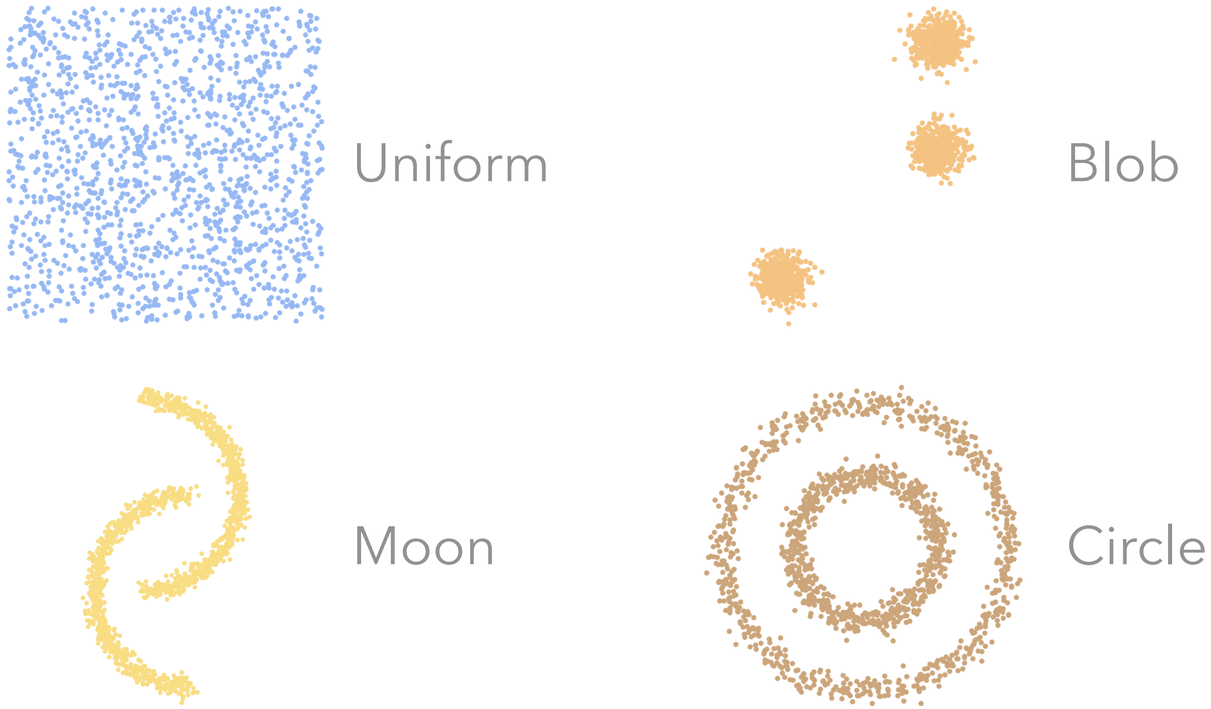
Distribution shift ➡ non-algorithmic behavior

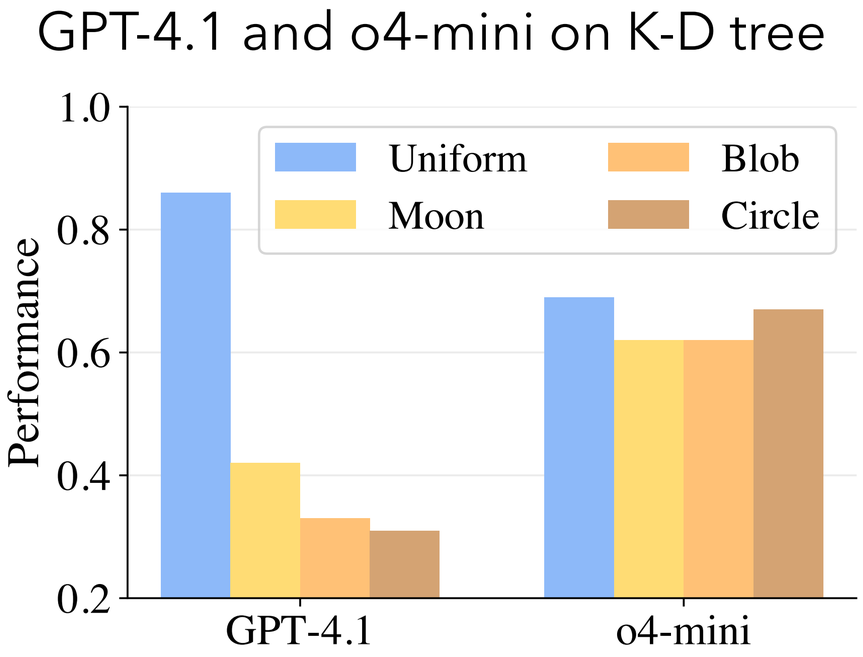
Auxiliary spatial probe varies dimensionality and input distribution



Distribution shift ➡ non-algorithmic behavior

Auxiliary spatial probe varies dimensionality and input distribution



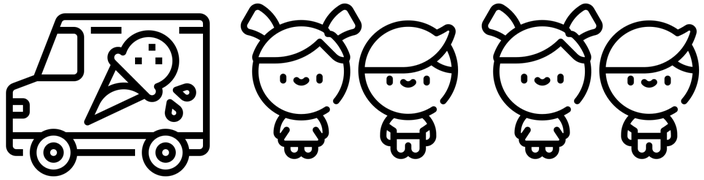


Take-away: Robust computation shouldn’t depend on familiar patterns

Auxiliary probe: Natural language

Realistic probe wraps structures in narrative scenarios

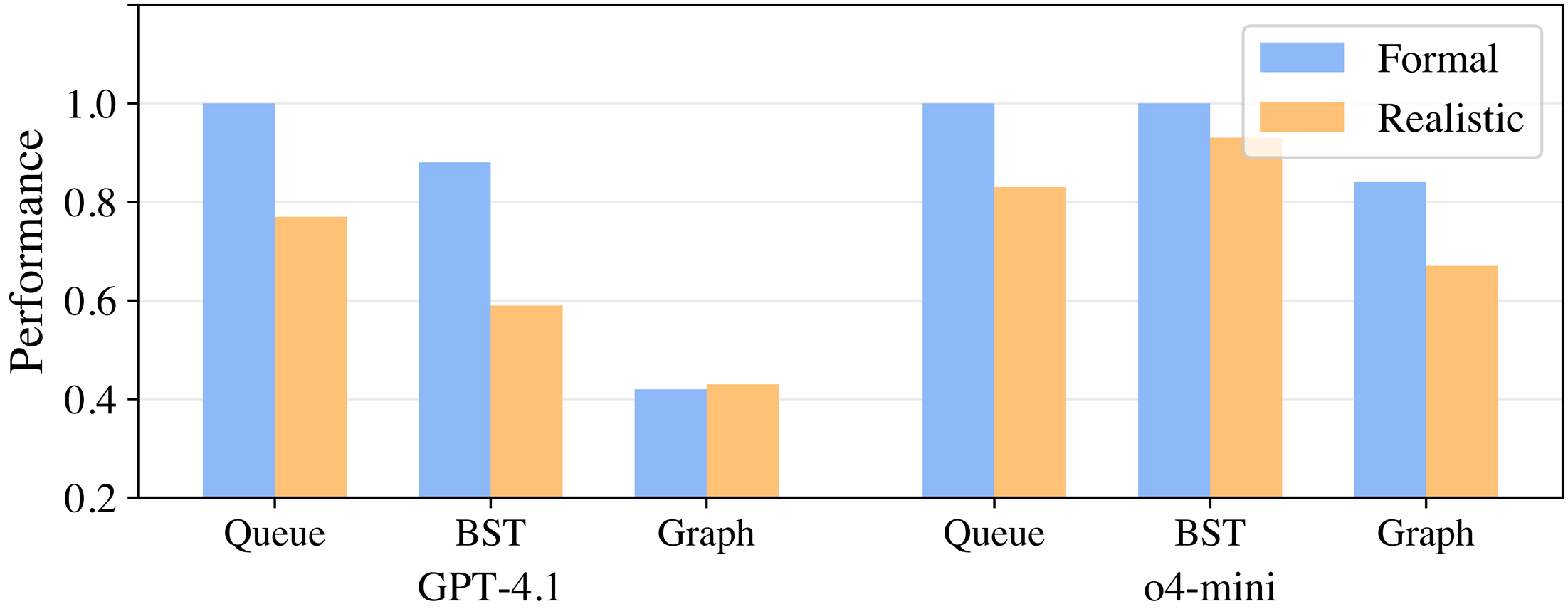
- On a sunny afternoon in the park, an ice cream truck rolled in…
- Children began to form a line, each newcomer taking their place at the end while the vendor served from the front...
  - Isabella Miller ran over and joined the ice cream line.
  - The next kid in line was served promptly.
- **Q:** What is the order of the remaining kids in line?



- Joining line means enqueue; served means dequeue

Auxiliary probe: Natural language

Take-away: **Performance drops versus formal descriptors**



Outline

1. Introduction and motivation
2. Benchmark details
3. Results and failure modes
4. **Conclusions and future directions**

Conclusions

TCS lens for evaluation: use primitives to study generalization, reliability

- DSR-Bench turns reasoning into **testable algorithmic primitives**
  - Order, mappings, hierarchies, networks, compositions
- Frontier models are strong, but **challenge** accuracy remains below 0.5
- Breakdowns appear under **composition** and **length**
- **Specifications** and **distribution shift** reveal non-algorithmic behavior
  - Priors override rules; familiar patterns matter
- **Natural language** adds a semantics bottleneck

Ongoing and future directions

- Use **verifiable tasks** for targeted RL
  - Reward exact intermediate states, not just answers
- Apply **interpretability** methods to structural failure modes
  - Compare activations on correct versus failing traces
  - Identify circuits for updates, memory, traversal

Can LLMs Reason Structurally? Benchmarking via the Lens of Data Structures

Ellen Vitercik

ICML'26

Yu He

Yingxi Li

Colin White

Stanford University

Stanford University