

Hierarchical reasoning and abstraction

Abstraction maps and the information required for decisions.

Reasoning agents, feedback, and scientific evidence

A short summary can erase the needed distinction

Two experiments share the summary “data ready.” In one, the measurement units and calibration are valid. In the other, calibration has expired.

A planner that sees only the summary takes the same action in both. Yet one case permits analysis and the other requires checking.

The problem is not simply too little memory. The representation has erased a distinction needed for the decision.

Constructing an abstraction around a decision

Construct a summary around the action it must support. If two detailed states need different actions, merging them may remove necessary information.

Adding calibration status repairs this example. Another task may require units, uncertainty or a history-dependent belief rather than the same field.

An abstraction map sends detailed states to a coarser representation. Its purpose determines which distinctions it must preserve.

Choosing what a representation preserves

Candidate	Reason to choose it	Cost or limitation
Fixed summary	Known fields support a stable task	An omitted distinction can change the action
Learned representation	Repeated tasks provide a useful training signal	Prediction quality may miss decision failures
Belief or structured state	Uncertainty and prerequisites affect decisions	More state and update logic must be maintained

Choose the simplest representation that preserves the distinctions needed for the stated decision, then test where it merges incompatible cases.

Pedagogical reconstruction

[Sutton et al. \(1999\)](#)

[Kaelbling et al. \(1998\)](#)

A scientific hierarchy can organise decisions

A project objective constrains candidate mechanisms; mechanisms motivate discriminating experiments; experiments require protocols; protocols invoke tools.

This is a course modelling proposal, not an interpretation automatically established by a neural network with multiple timescales. The levels may overlap and influence each other.

Use explicit interfaces to state what each level receives, produces, and can revise when a lower-level result contradicts an assumption.

State aggregation can destroy the Markov property

Two detailed states may map to the same abstract state while requiring different actions or having different transition distributions.

A controller that sees only the abstract state then faces hidden information. It may need memory, a refined representation, or a probabilistic belief state.

Evaluate abstraction quality through downstream decisions and transition predictions, rather than treating compression alone as evidence of useful hierarchy.

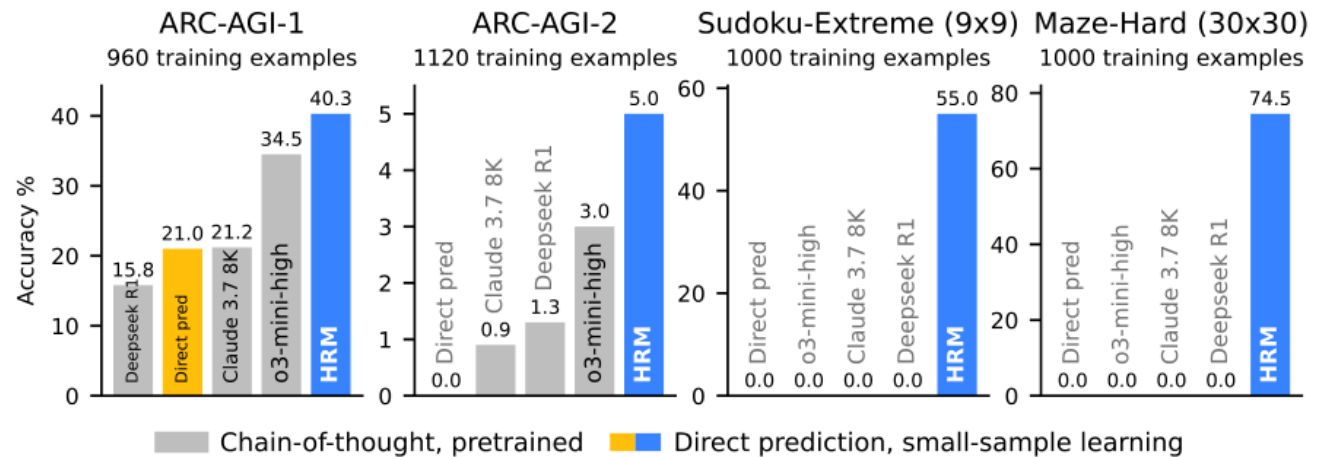
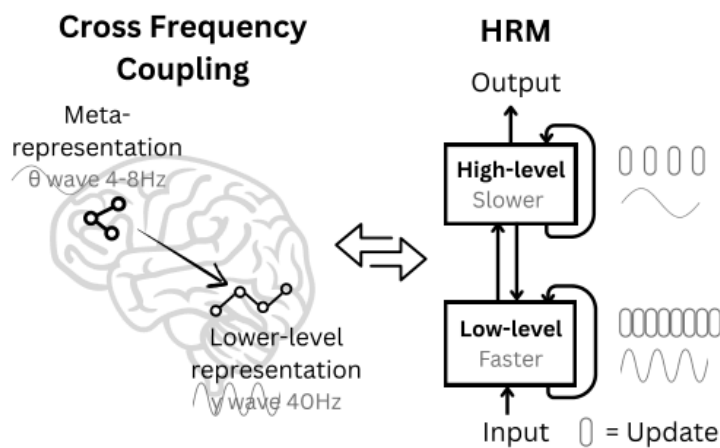
Hierarchical Reasoning Model: the method

HRM uses coupled recurrent components operating at different update rates, together with a training procedure designed for deep iterative computation.

The architecture is evaluated on structured reasoning tasks such as puzzles. Its state-update schedule and supervision are concrete computational choices.

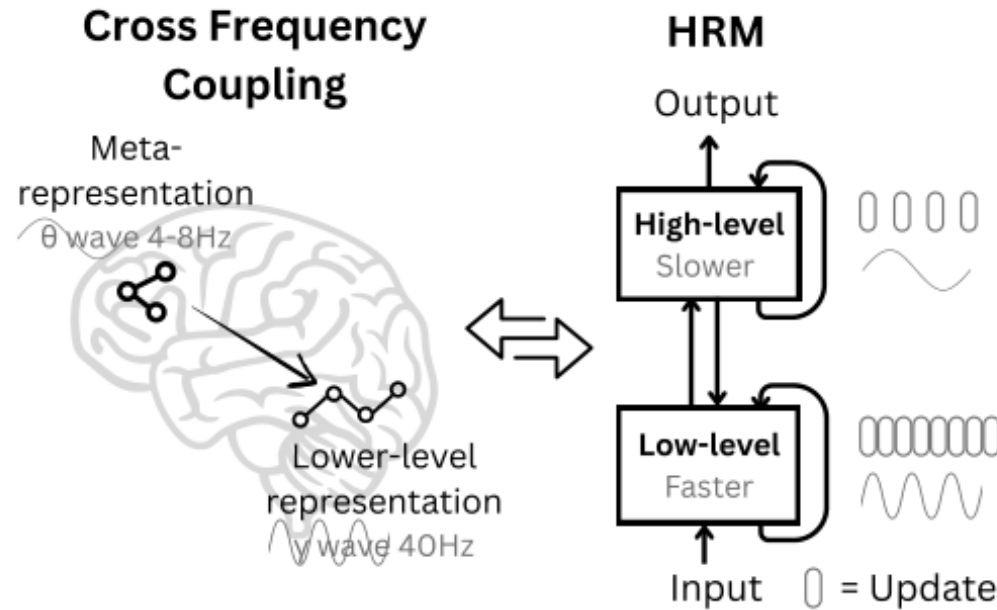
Calling the states high-level and low-level describes architectural roles; interpreting them as scientific strategy and procedure would require additional causal and behavioural evidence.

HRM: coupled recurrent components



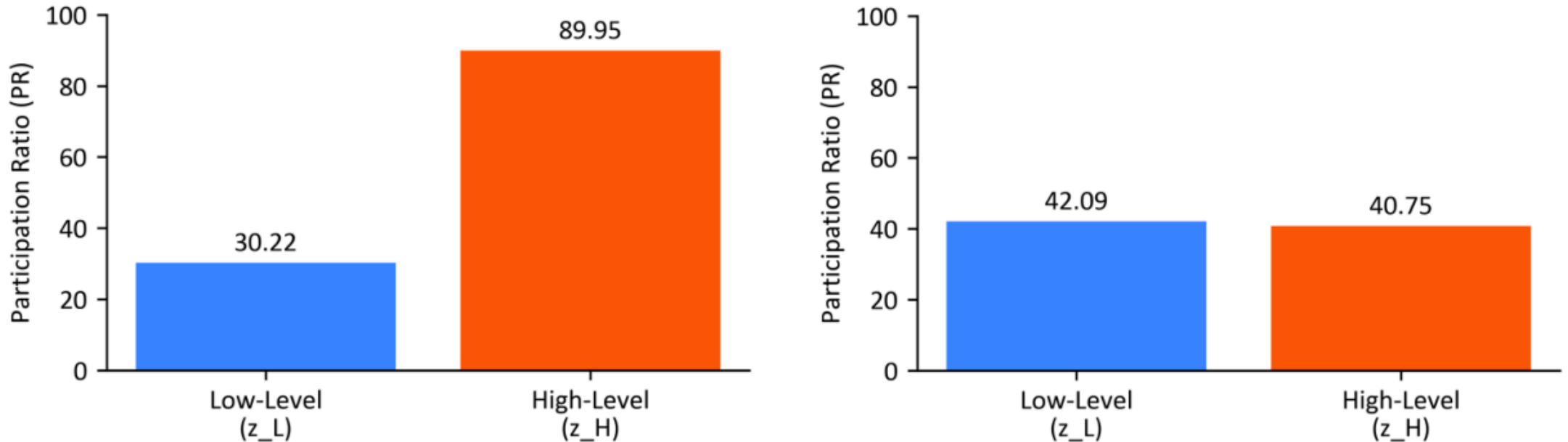
Two recurrent components operate on different timescales through feedback connections. The implemented transitions define the model. Biological inspiration supplies motivation rather than an additional performance guarantee.

Hierarchical Reasoning Model: mechanism detail



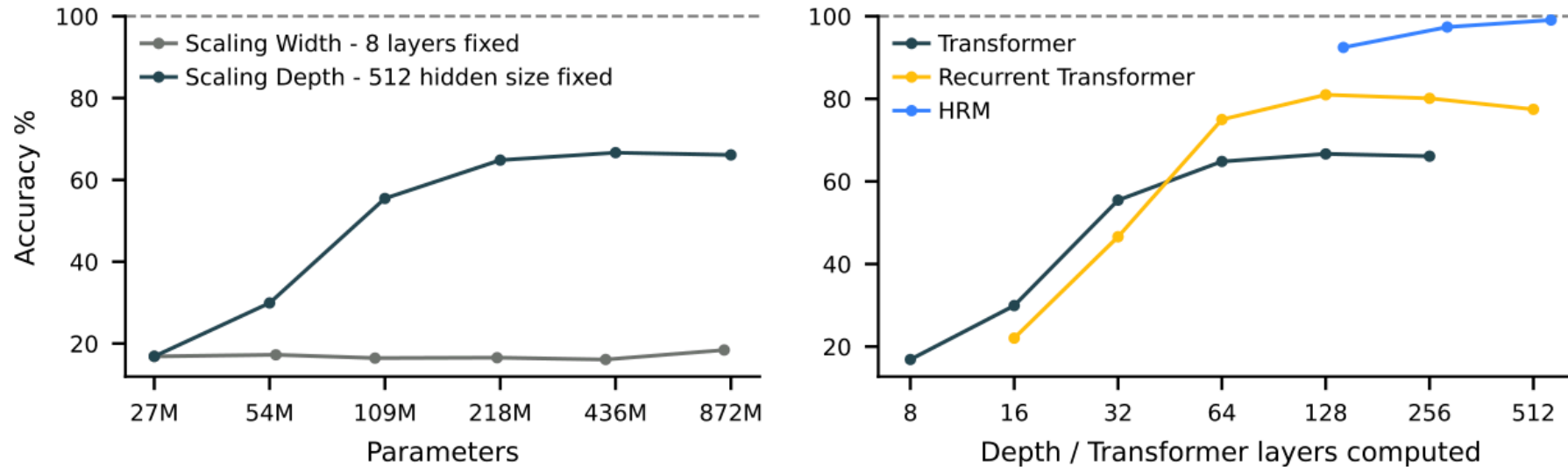
The two recurrent states receive different information and update on different schedules. This documented coupling describes the mechanism; its interpretation as scientific levels is a separate, untested claim.

Evidence: dimensionality does not identify an algorithm



In the trained model, high- and low-level participation ratios are 89.95 and 30.22; in the untrained control, they are 40.75 and 42.09. Participation ratio measures how covariance spreads across directions. This representational statistic alone does not establish that a module causally implements abstract planning.

Hierarchical Reasoning Model: interpretation and limits



Small structured datasets, augmentation, supervision choices, and test-time computation affect comparisons. Puzzle success and a two-timescale architecture do not establish a general scientific hierarchy or a biological explanation of reasoning.

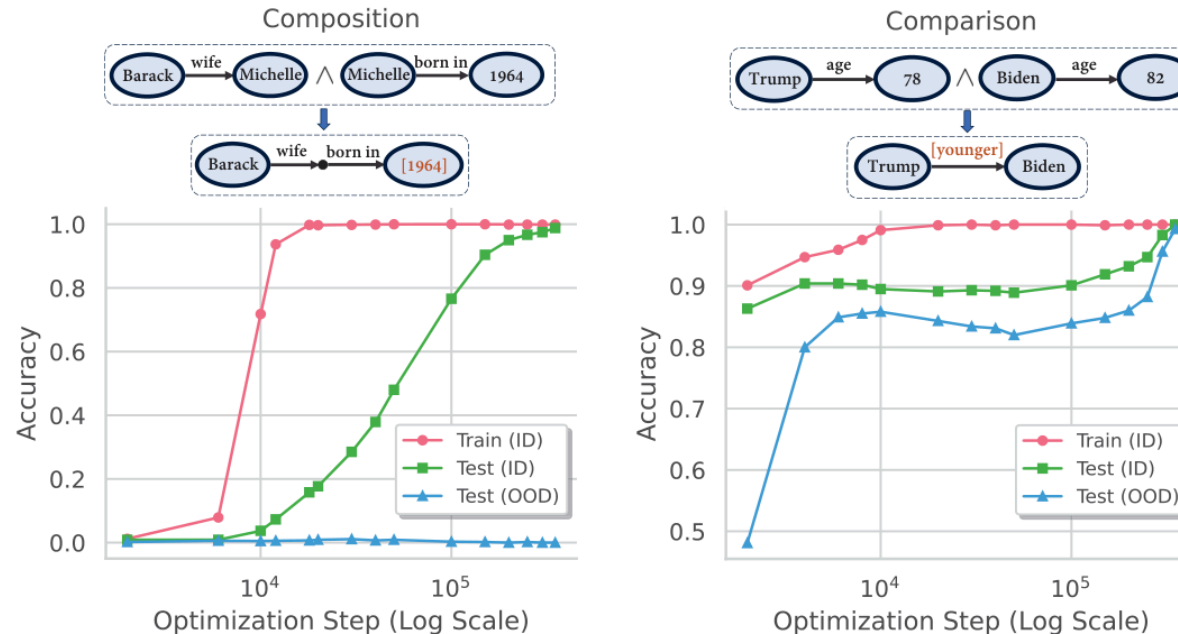
Grokked transformers: controlled study design

The study trains transformers on controlled reasoning relations and analyses generalisation after extended training.

It investigates composition and comparison tasks, connecting behavioural changes with circuit-level analyses.

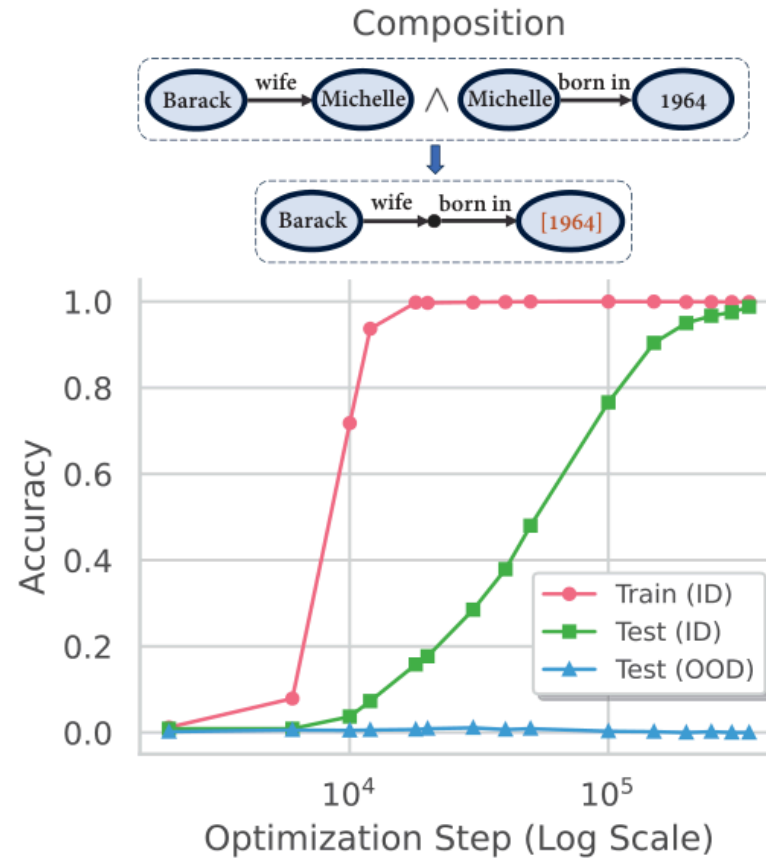
The controlled setup permits a closer test of implicit reasoning than a broad language benchmark, while its synthetic structure also limits direct conclusions about open scientific reasoning or universally meaningful latent abstractions.

Grokking transformers: generalisation curves



Training, in-distribution test and out-of-distribution test curves show different stages of learning. Generalisation may change after training accuracy is high. The relational examples define the particular reasoning operations under evaluation.

Grokked transformers: composition task



The composition example specifies a relation to be inferred and information withheld from the test input. This task construction determines whether successful prediction is evidence of the intended reasoning operation.

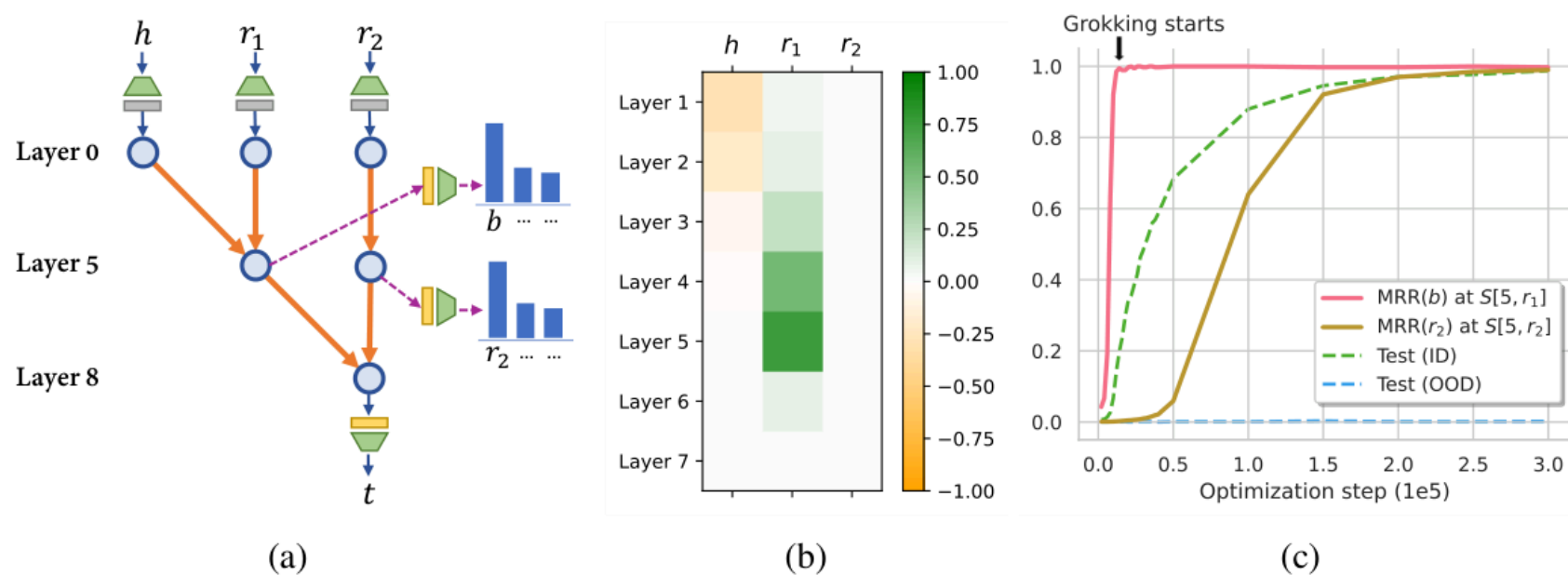
Evaluation scope

The source reports delayed generalisation and studies internal circuits associated with composition and comparison.

The extracted circuit figure links architectural locations, interventions or attribution measures, and behavioural learning curves.

An observed association between a representation and generalisation is a starting point for causal analysis. Check the intervention procedure before concluding that a particular circuit implements the claimed computation.

Grokking transformers as implicit reasoners: interpretation and limits



Grokking depends on the training setup and task. An interpretable circuit in a controlled relation task is not automatically a scientific mechanism representation or evidence that every pretrained model can reason similarly.

Options formalise temporally extended actions

An option specifies an initiation set, an internal policy, and a termination condition. It compresses a sequence of primitive actions into an action usable by a higher-level policy.

Duration and accumulated reward matter because options can last different amounts of time. A failed precondition should prevent initiation.

In a research workflow, “validate the measurement” is a candidate option only when its inputs, completion test, and failure states are defined.

Timescale separation is an empirical hypothesis

A slow state can coordinate several updates of a fast state. This can stabilise or structure computation, but its semantic role must be established rather than named into existence.

Matched one-state and two-state models, altered update schedules and equivalent total depth isolate different explanations for a hierarchical model's performance.

Interventions and task-specific probes distinguish a state that carries strategic information from another representation of local computation.

Abstraction contracts expose failures

A contract states which variables are retained, which effects are abstracted away, and which predictions should remain approximately invariant.

When an abstract plan fails, identify whether the cause is a broken contract, a poor plan, or a failed executor. Each diagnosis suggests a different repair.

A hierarchy becomes useful for scientific reasoning when it makes these distinctions easier to test and revise.

Three meanings of a higher-level step

Object	Internal state	Termination or reuse
Option in reinforcement learning	Policy plus an initiation set	A termination rule ends a temporally extended action
HRM recurrent computation	Learned high/low hidden states	An architecture-specific halting mechanism
Voyager procedural skill	Executable code with a description	Retrieve and invoke a reusable behavior

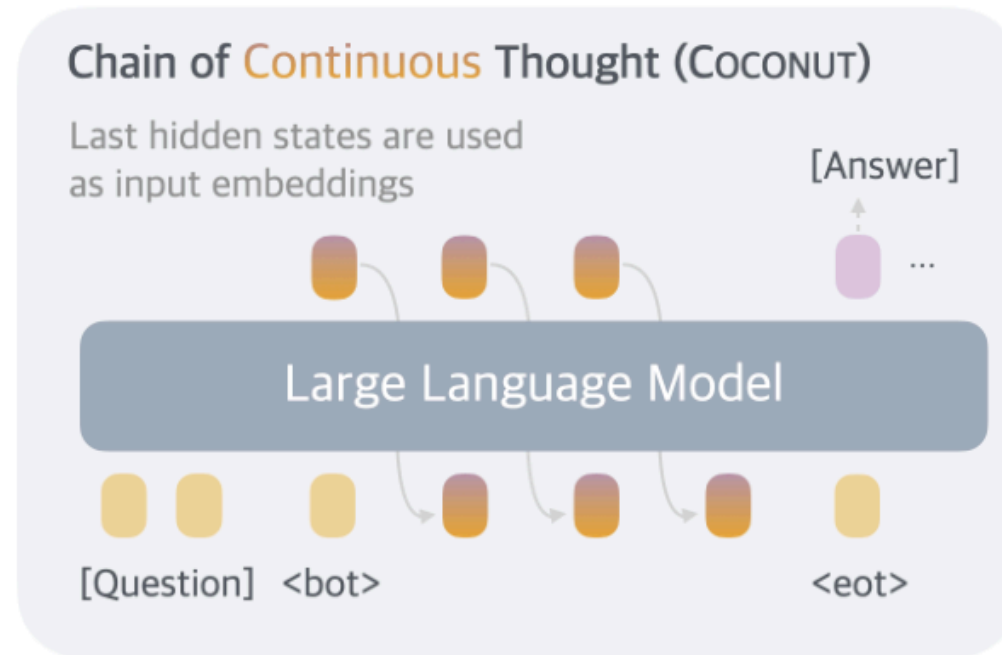
An option, a recurrent state and a stored program are different abstractions. The shared word “hierarchy” does not make them equivalent.

An abstraction can hide the decisive variable

Underlying state	Coarse summary	Best action in the example
Sensor warmed; reference stable	Measurement disagreement	Recalibrate the sensor
Sensor stable; reference drifting	Measurement disagreement	Repair or replace the reference
Both mapped to one state	“Disagreement” omits its cause	One policy cannot choose both correct actions

Synthetic state-aliasing example. A useful abstraction must preserve distinctions needed for the decisions it supports.

Coconut continuous thoughts: mechanism detail



Replacing a token embedding with a hidden state changes the input representation. The training objective and curriculum determine how the model encounters that representation and its possible distributional mismatch.

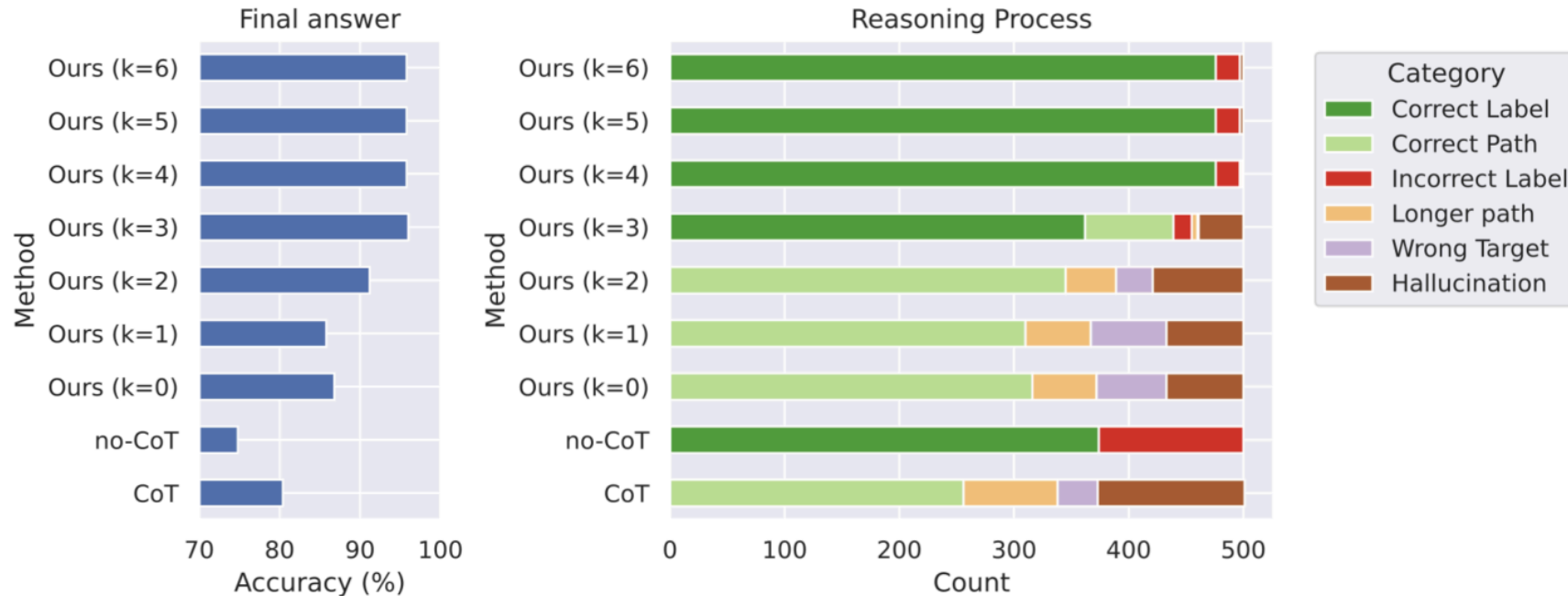
Evaluation scope

The source evaluates reasoning tasks and analyses final-answer accuracy alongside properties of the generated reasoning process.

The original bars distinguish correct labels, correct paths, and several failure categories as the number of continuous steps changes.

These diagnostics support a task-specific investigation of latent computation. Transfer to scientific mechanism search requires testing predictions, calibration, and experiment choice in that setting.

Coconut continuous thoughts: interpretation and limits



Benefits vary across tasks and training conditions. A decoded trace or probe may not faithfully represent internal computation, and a reduction in visible tokens is not necessarily a proportional reduction in FLOPs or latency.

An abstraction preserves selected information

An abstraction compresses states for a particular prediction or decision. Two states that look similar can require different actions.

State the preserved property before claiming transfer. A representation useful for retrieval may be insufficient for planning or scientific inference.

Three reusable objects

Object	What is reused	Validity question
State abstraction	Compressed representation	Does relevant information survive?
Skill	A policy over actions	Do initiation and termination apply?
Plan template	A structured action sequence	Do preconditions and effects match?

Reuse saves effort only if the object remains applicable in the new decision context.

A transfer contract is explicit

Record the conditions under which a reusable procedure starts, the observations it requires and the condition that ends it. Identify failure signals that return control to the planner.

Scientific reuse also requires compatible units, identifiers and measurement conventions. A familiar procedure name is weak evidence of compatibility.

A compressed state loses a decision variable

Full state	Compressed label	Valid next action
Data calibrated; units known	Data ready	Estimate parameter
Data uncalibrated; units known	Data ready	Calibrate
Data calibrated; units missing	Data ready	Recover units

Synthetic workflow. The common label hides information needed to select a valid next action.

A skill has its own time scale

Operation	Calculation or state	Consequence
Skill duration	Two primitive steps	Discount across both steps
Rewards	First: 1; second: 2	Discounted reward $1+0.9\times 2$
Continuation	Next-state value 5	Discounted by 0.9^2
Backup target	$1+1.8+4.05=6.85$	A one-step shortcut is wrong

Synthetic two-step option with discount factor 0.9. The state abstraction and termination rule determine which continuation value is appropriate.

Worked abstraction: identical summaries, different actions



The abstraction maps two decision-relevant states to the same input. A deterministic memoryless abstract policy must choose the same action in both. Adding a calibration flag or an evidence-based belief restores the missing distinction; randomisation alone cannot recover the discarded information.

Show why a memoryless abstract policy fails

A deterministic policy using only the abstract state must choose the same action in s_1 and s_2 . It cannot reproduce the correct state-dependent choice in both.

Randomisation does not restore the missing information; it merely mixes errors and unnecessary work.

The example demonstrates an information limitation of this abstraction. It does not imply that all compact or learned representations are unsuitable for scientific planning.

Repair the abstraction with a sufficient distinction

Add a calibration-status variable or a belief about measurement validity. The policy can now distinguish a ready-to-report state from one requiring validation.

If calibration status is itself uncertain, store the relevant evidence and update a belief rather than inventing a precise flag.

The right repair depends on what observations are available. More detail is useful only when it supports a better decision or a more reliable transition model.

Abstraction transfer across scientific contexts

Tasks involving calibration, batch effects or context-dependent mechanisms expose differences between a value-only summary, a typed state and a learned representation.

A fixed high-level planner isolates the representation's contribution. A fixed representation isolates changes in planning.

Decision regret and failure diagnoses on held-out contexts reveal transfer limitations. A representation that works on one simulator can omit a variable that becomes decisive elsewhere.

Options change the Bellman backup

$$Q(s, o) = \mathbb{E} \left[\sum_{j=0}^{k-1} \gamma^j r_{t+j} + \gamma^k \max_{o'} Q(s_{t+k}, o') \right]$$

An option o runs for random duration k before terminating. Rewards accumulate during its execution, and the continuation value is discounted by γ^k .

The option has an initiation set, internal policy, and termination rule. These specify when the abstraction is usable and when control returns.

Discounting every option by only γ would incorrectly treat a long procedure as a one-step action.

Decision preservation under abstraction

Construct two histories mapped to the same abstract state but with different sensor-calibration status. If the optimal next test differs, the abstraction omits decision-relevant information.

Add the missing status variable or maintain a belief over it, then repeat the comparison.

Test value and transition predictions under the abstraction. A readable hierarchy is useful only to the extent that its compressed state supports the intended decisions.

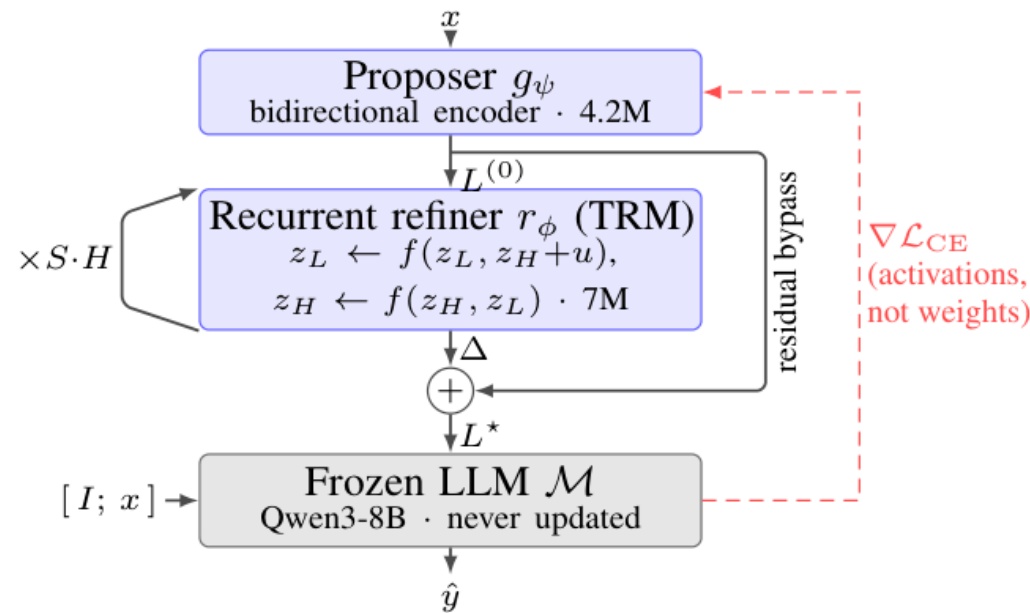
Recent research: Latent Recurrent Thoughts

LRT uses a task-specific proposer to construct continuous inputs, a recurrent refiner to improve them through bounded residual corrections, and a frozen language model to decode an answer.

The reported controlled comparison holds the decoder, prompts, training data, and budget fixed while varying the latent module.

The literature snapshot is dated 11 September 2026. Findings remain conditional on the source version and evaluation conditions. Recent preprints contribute reported results with distinct validation requirements.

LRT: latent refinement and decoding architecture



Base latents enter the recurrent refiner and then a frozen decoder. The trained and frozen modules define the adaptation boundary. This comparison addresses a different question from training the entire model end to end.

Recent methods in context

Recursive Language Models (preprint; revised 2026). Treats long context as an external environment that can be inspected programmatically and processed through recursive model calls. Nested calls can increase cost unpredictably; long-context benchmark gains do not establish universal task efficiency.

Agent Primitives (ICML 2026). Composes reusable agent operations using KV-cache communication and an organizer that chooses a configuration for each query. Attribution to collaboration or latent communication depends on matched backbones and resource budgets.

Hilbert (ICLR 2026). Combines informal decomposition, a formal prover, retrieval and verifier feedback to construct proofs recursively. Formal verification validates a formal statement under its assumptions; correspondence to the intended mathematical question remains a separate issue.

References (1/2)

[Guan Wang, Jin Li, Yuhao Sun et al. \(2025\). Hierarchical Reasoning Model](#)

[Boshi Wang, Xiang Yue, Yu Su et al. \(2024\). Grokked Transformers are Implicit Reasoners: A Mechanistic Journey to the Edge of Generalization](#)

[Shibo Hao, Sainbayar Sukhbaatar, DiJia Su et al. \(2024\). Training Large Language Models to Reason in a Continuous Latent Space](#)

[Alexia Jolicoeur-Martineau \(2025\). Less is More: Recursive Reasoning with Tiny Networks](#)

[Richard S. Sutton, Doina Precup, Satinder Singh \(1999\). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning](#)

[Leslie Pack Kaelbling, Michael L. Littman, Anthony R. Cassandra \(1998\). Planning and acting in partially observable stochastic domains](#)

References (2/2)

[Zhaoliang Chen, Jie Fu \(2026\). Latent Recurrent Thoughts: Recurrent Refinement of Proposed Latents for Reasoning with Frozen LLMs](#)

[Alex L. Zhang, Tim Kraska, Omar Khattab \(2025\). Recursive Language Models](#)

[Haibo Jin, Peng Kuang, Ye Yu et al. \(2026\). Agent Primitives: Reusable Latent Building Blocks for Multi-Agent Systems](#)

[Sumanth Varambally, Thomas Voice, Yanchao Sun et al. \(2025\). Hilbert: Recursively Building Formal Proofs with Informal Reasoning](#)

[Guanzhi Wang, Yuqi Xie, Yunfan Jiang et al. \(2023\). Voyager: An Open-Ended Embodied Agent with Large Language Models](#)