Survey on Implicit Reasoning in LLMs: Hidden Computation, Latent Thought, and Test-Time Reasoning Beyond Text

Mohammad Ebrahimian
Sharif University of Technology

Armin Geramirad
Sharif University of Technology

Taha Izadi
Sharif University of Technology

Mahshid Dehghani
Sharif University of Technology

Sahand Akramipour
Sharif University of Technology

Shaygan Adim
Sharif University of Technology

Reviewed on OpenReview:

Abstract

Explicit chain-of-thought (CoT) is a central interface for improving large-language-model reasoning, but visible rationales are only one part of how models compute. Recent work shows that reasoning-relevant information can appear in hidden states, logits, activation trajectories, recurrent states, continuous thoughts, discrete latent codes, diffusion latents, abstract tokens, guidance vectors, and learned reasoning modules before or instead of being verbalized. This survey synthesizes this literature through a multi-level taxonomy. At the top level, we distinguish work that studies latent reasoning as an object to be understood, work that builds latent reasoning systems, and work that formalizes the computational role of latent thought. These classes are further divided into methodological subclasses covering observability, dynamics, compression, latent languages, guidance and steering, iterative or switch-based computation, and distributional or formal reasoning. Beyond taxonomy, the survey contributes a terminology layer, a narrative account of how diagnostic evidence led to constructive systems and theory, benchmark-specific comparison guidance, and a structured set of research gaps and future directions. This survey explores an emerging hybrid perspective: the hypothesis that text may support memory, supervision, communication, and stochastic sampling, while latent computation could potentially offer compactness, semantic density, adaptive compute, and non-verbal search.

1 Introduction

Large language models often improve on difficult tasks when they write intermediate steps before answering. This behavior, usually called Chain-of-Thought reasoning, gives the model a serial scratchpad and exposes part of the solution process in natural language (Wei et al., 2022). Yet a written rationale is not the same as the full computation that produced it. Hidden representations can contain bridge entities, future-answer

signals, correctness cues, and local planning information before the corresponding text is emitted (Yang et al., 2025; Xu et al., 2026). Generated rationales are therefore an interface to reasoning rather than a complete account of the underlying mechanism.

This broader substrate is the focus of *implicit reasoning*. We use the term for reasoning-relevant computation that is encoded, transformed, or controlled in hidden states, logits, recurrent iterations, latent vectors, learned code tokens, guidance vectors, or auxiliary reasoning modules, whether or not that computation is fully verbalized. The literature now includes both diagnostic work on what ordinary models already encode and constructive work that trains or controls models to reason through continuous thoughts, abstract tokens, recurrent depth, diffusion latents, switch-based inference, or cross-model latent guidance (Hao et al., 2025; Ramji et al., 2026; Shi et al., 2026; Chen et al., 2026).

The field has developed in a sequence of connected problems. Diagnostic studies first asked whether internal activations reveal multi-hop variables, trajectory success, self-verification, or planning horizons. Dynamic and geometric studies then reframed reasoning as motion through representation space. Constructive studies converted these observations into systems that compress CoT, preserve semantic alignment, transfer latent guidance, optimize hidden states, refine latent embeddings, or alternate between explicit and latent modes. Formal studies ask when latent thought has a computational advantage over textual thought, and when text remains preferable because it supports stochastic sampling, memory, or communication (Zhou et al., 2026; He et al., 2025; Zhu et al., 2025).

A strictly paper-by-paper presentation might obscure an emerging conceptual shift. The object of study appears to be broadening beyond simply the written rationale. Researchers are increasingly exploring a potential family of reasoning media: natural-language tokens, hidden activations, continuous thought vectors, discrete abstract codewords, recurrent states, diffusion latents, guidance vectors, and distributions over latent trajectories. Current literature suggests these media may present different trade-offs across dimensions such as accuracy, cost, interpretability, semantic fidelity, stability, and controllability.

We organize the survey through a multi-level classification. At the top level, we distinguish three roles for latent reasoning research. *Understanding latent reasoning* asks what hidden reasoning information exists and how it evolves. *Building latent reasoning systems* studies how latent states can replace, compress, guide, steer, control, or scale reasoning computation. *Formalizing latent reasoning* asks what latent states can represent in principle and how they differ from explicit textual thought. Within these classes, we identify seven methodological subclasses: latent-state observability and probing; reasoning dynamics and geometry; explicit-to-implicit CoT compression; learned latent reasoning languages; modular guidance, steering, and controllers; iterative or switch-based latent computation; and distributional and formal reasoning.

The survey contributes four elements. First, it defines terminology that separates hidden states, latent states, latent thoughts, latent tokens, trajectories, and controllers, reducing ambiguity in a literature where these terms are sometimes conflated. Second, it provides a multi-level taxonomy that is not merely chronological or architectural, but organized by the role played by latent computation. Third, it develops a narrative stack from hidden-state evidence to trajectory analysis, constructive latent systems, and formal limits. Fourth, it identifies benchmark-compatible comparisons, research gaps, and future directions that follow from the synthesis rather than from any single method.

An emerging perspective in the surveyed literature leans toward a hybrid approach rather than a strict replacement of existing methods. Explicit CoT is frequently highlighted for its readability, ease of supervision, and natural support for sequential memory and stochastic exploration. Concurrently, latent computation shows potential for offering compactness, semantic density, recurrence, optimizability, and reduced constraints from surface language. Consequently, an ongoing design challenge is not necessarily whether to eliminate text, but rather exploring how computation might be allocated across visible and hidden media: investigating when a model should speak, think silently, recur, compress, request guidance, or stop(Wei et al., 2022; Hao et al., 2025; Shi et al., 2026).

The survey is organized as follows. Section 2 defines terminology and scope. Section 3 introduces background concepts. Section 4 presents the taxonomy, and Section 5 summarizes the comparative trajectory of the field. Sections 6 and 7 cover the understanding class. Sections 8–11 cover constructive systems. Sections 12 and

13 cover distributional and formal accounts. Sections 14–17 discuss evaluation, research gaps, and future directions.

2 Terminology and Scope

2.1 Terminology

The phrase "latent reasoning" is used inconsistently across the literature, so we fix the terminology used throughout this survey. A *hidden state* is a concrete internal vector produced by a model layer at a token position. A *latent state* refers to a broader category of internal mathematical representations that carry reasoning information. Rather than being limited to a standard hidden state produced by a model's layer, it encompasses various forms of 'silent computation'—such as an optimized vector in a latent space, a state maintained across recurrent iterations, a variable refined during a diffusion process, or the output of a specialized auxiliary reasoning module. A *latent thought* is a latent state explicitly used as an intermediate reasoning step, for example by feeding it back, refining it, or conditioning answer generation on it (Hao et al., 2025; Kong et al., 2026).

A *latent token* is a discrete or vector-valued placeholder that substitutes for a segment of textual reasoning. Codebook-based methods, reserved abstract symbols, and implicit CoT tokens all fall under this term, even though their training objectives differ (Su et al., 2025; Ramji et al., 2026; He et al., 2025). A *latent trajectory* is an ordered sequence of latent states produced by recurrence, diffusion, flow matching, residual refinement, or explicit-latent switching. A *latent controller* is any mechanism that chooses, steers, refines, or routes latent computation, including learned networks, optimization objectives, training-free switching policies, and large-to-small guidance vectors (Jiang et al., 2026; Shi et al., 2026; Chen et al., 2026).

When we say that latent reasoning enables *parallelism*, we do not automatically mean GPU-level parallel execution or lower wall-clock time. We usually mean *representation-level* or *algorithmic* parallelism: a single vector, distribution, or recurrent state can maintain several hypotheses, search frontiers, or partial trajectories before committing to a surface token. Hardware efficiency is a separate empirical question that depends on latent updates, forward passes, cache behavior, auxiliary modules, and decoding overhead (Zhu et al., 2025; Xu & Sato, 2026).

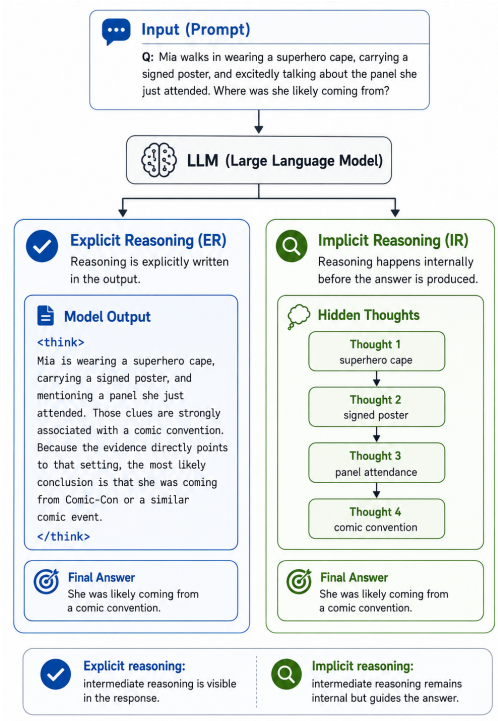

Figure 1: Conceptual distinction between explicit and implicit reasoning.

2.2 Compact mathematical notation

A useful way to separate explicit, implicit, and hybrid reasoning is to distinguish the *medium* of the intermediate computation. Given a query q, explicit CoT introduces a visible textual trace $r_{1:T}$ before producing an answer $y_{1:U}$:

$$p_\theta(r_{1:T}, y_{1:U} \mid q) = \prod_{t=1}^{T} p_\theta(r_t \mid q, r_{<t}) \prod_{u=1}^{U} p_\theta(y_u \mid q, r_{1:T}, y_{<u}). \tag{1}$$

Implicit reasoning replaces some or all of the textual trace with latent states $z_{1:K}$, where each z_k may be a hidden state, a learned latent token, a recurrent state, a guidance vector, or an optimized representation:

$$z_{1:K} = G_\phi(q), \qquad p_\theta(y_{1:U} \mid q, z_{1:K}) = \prod_{u=1}^{U} p_\theta(y_u \mid q, z_{1:K}, y_{<u}). \tag{2}$$

Hybrid methods can be viewed as choosing a reasoning mode $m_k \in \{\text{text}, \text{latent}\}$ at each intermediate step. This notation makes clear why two systems with similar answer accuracy may still differ substantially: one may spend computation in T visible tokens, another in K latent updates, and a third in controller or teacher-model calls. Thus, the comparison target is not only $p(y \mid q)$, but also the cost and faithfulness of the intermediate medium.

2.3 Scope and boundaries

This survey focuses on implicit and latent reasoning in language models. We include hidden-state probing, logit- or activation-level diagnostics, reasoning trajectories, continuous and discrete latent thoughts, CoT compression, latent guidance, steering and refinement, recurrent or switch-based inference, distributional latent search, and formal comparisons between textual and latent thought.

We do not attempt a general survey of prompt engineering, tool use, agents, retrieval, multimodal reasoning, or cognitive-science theories of thought. Such topics are discussed only when they directly alter, diagnose, or formalize latent reasoning in LLMs. We also avoid treating all efficiency results as directly comparable: a method that reduces generated tokens, a method that adds recurrent hidden computation, and a method that invokes a teacher model may occupy different points in the accuracy-cost tradeoff.

3 Background: Reasoning Beyond Generated Text

Explicit CoT as a serial scratchpad. Chain-of-Thought prompting and training improve reasoning by letting a model emit intermediate tokens before the final answer (Wei et al., 2022). This increases effective computation depth and provides a memory trace that future tokens can attend to. It also has costs. It consumes many decoding steps, can include linguistically useful but computationally redundant tokens, may be unfaithful to the model's internal decision process, and forces every intermediate state through a discrete natural-language bottleneck (He et al., 2025; Shi et al., 2026).

Implicit reasoning as hidden computation. Implicit reasoning refers to reasoning-relevant information or transformations that occur without being fully written. In a factual multi-hop query, a hidden state may encode the bridge entity; in mathematical reasoning, a state may encode success likelihood; in self-verification, a state may encode correctness; in planning, a model may have only a short latent horizon even while producing a long chain (Yang et al., 2025; Afzal et al., 2025; Zhang et al., 2025; Xu et al., 2026). Decoding such information is useful, but decodability alone is not causal evidence. A variable can be linearly recoverable from hidden states while playing little role in the model's next-token policy.

Latent thought as a computational medium. Constructive methods use latent states not only as objects of analysis, but as actual reasoning media. Continuous-thought methods feed hidden states back as inputs (Hao et al., 2025). Discrete latent methods compress textual spans into learned or reserved tokens (Su et al., 2025; Ramji et al., 2026). Self-distillation and semantic-alignment methods train implicit tokens to preserve the reasoning information of explicit traces (Shen et al., 2025; He et al., 2025). Recurrent-depth methods spend extra compute by applying shared blocks repeatedly (Geiping et al., 2025; Kohli et al., 2026; Saunshi et al., 2025). Diffusion, flow, switching, and rethinking methods treat latent thought as an object to refine, sample, route, or optimize (Kang et al., 2026; Shi et al., 2026).

Hidden states, probes, and logit lenses. A common diagnostic tool maps intermediate representations into interpretable targets: bridge entities, future tokens, final answers, correctness labels, uncertainty, or reasoning length. Logit-lens methods project hidden states through a language-model head or an adapted head (nostalgebraist, 2020). These tools are most informative when paired with interventions, cross-task comparisons, or downstream controllers, because they distinguish "information is present" from "information is used."

Trajectories, steering, and geometry. Reasoning evolves over layers, tokens, recurrent iterations, and latent steps. This motivates studying trajectories: where the hidden state is, how quickly it moves, which direction it moves in, and whether its path curves toward a decision (Zhou et al., 2026; Kumar et al., 2026). Recent control-oriented papers push this idea further: latent guidance distills a large model's plan into vectors for a smaller executor, contrastive feedback searches for better update directions, and gradient-based representation optimization steers a base model toward CoT-like hidden states while regularizing against language-prior drift (Chen et al., 2026; Wang et al., 2026a;b).

Why theory matters. Latent reasoning systems raise a formal question: what can latent thought compute more efficiently than text, and what does text still do better? Recent work suggests that continuous or looped latent states can parallelize some deterministic computations, such as DAG evaluation or breadth-first graph search (Zhu et al., 2025; Xu & Sato, 2026). At the same time, explicit CoT benefits from stochastic token generation and can support approximate counting and sampling in ways deterministic latent loops cannot. This motivates hybrid rather than exclusive reasoning designs.

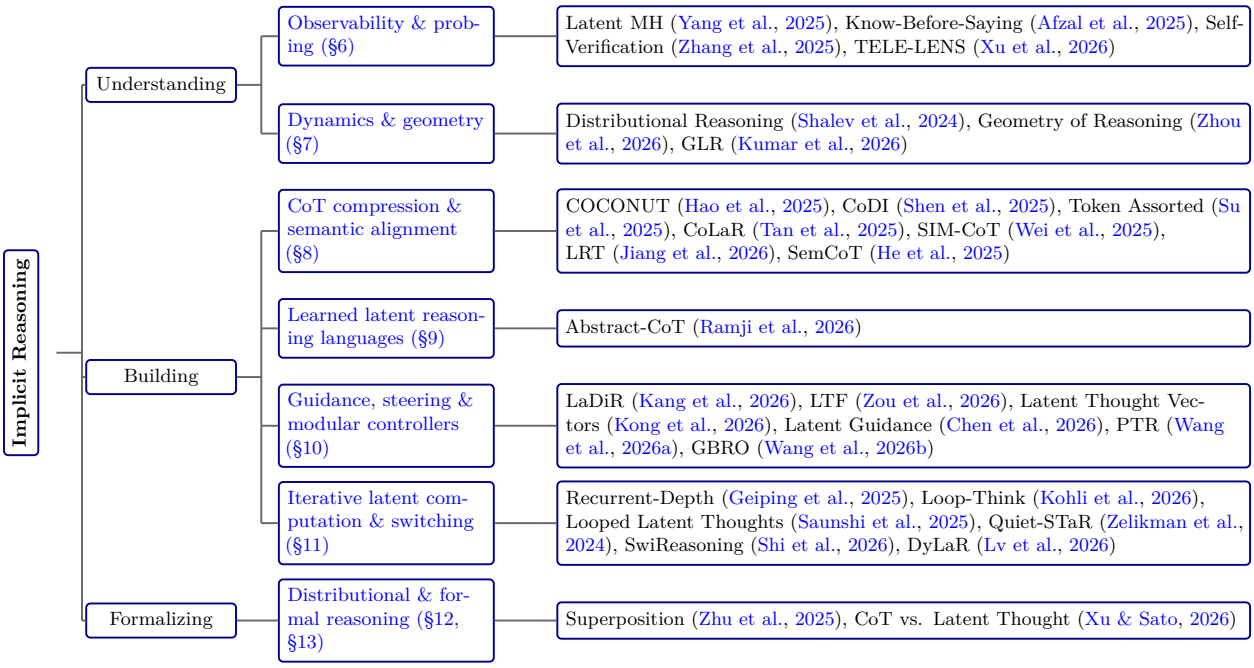

Figure 2: Taxonomy of implicit reasoning.

From background concepts to taxonomy. These background ideas motivate the hierarchical taxonomy used in the rest of the survey. Observability, probes, and trajectory analysis naturally support the *understanding* class; latent tokens, recurrent depth, diffusion latents, semantic alignment, large-to-small guidance, and auxiliary controllers motivate the *building* class; and computational separations between textual and latent thought motivate the *formalizing* class (Yang et al., 2025; Hao et al., 2025; Zhu et al., 2025). The survey therefore groups papers first by their top-level research objective and only then by the more specific mechanism they introduce or analyze.

4 A Taxonomy of Implicit Reasoning

We organize the literature with a *multi-level* taxonomy. At the highest level, the field asks three questions: how latent reasoning can be *understood*, how it can be *built and improved*, and how it can be *formalized*. Each top-level class then splits into methodological subclasses. This structure is more informative than a flat continuous-versus-discrete split, because methods using similar representations can pursue different scientific goals, while methods using different representations can solve the same systems problem (Hao et al., 2025; Ramji et al., 2026; Shi et al., 2026).

Figure 2 summarizes the multi-level taxonomy with representative method labels. Table 1 lists the primary placement of each surveyed paper. The placements are not meant to imply disjoint intellectual boundaries. Several papers intentionally cross categories: LRT compresses explicit trajectories but also uses a latent reasoning network; LaDiR is a modular latent reasoner with distributional sampling behavior; SemCoT is an implicit-token method whose main novelty is semantic alignment; and SwiReasoning and DyLaR are iterative methods that also act as controllers for explicit-latent mode selection. We assign papers by their central contribution and discuss secondary roles in the corresponding thematic sections.

The three top-level classes provide the organizing logic for the rest of the survey. The *understanding* class concerns evidence and dynamics: what information is present internally, where it appears, and how it moves. The *building* class concerns interventions and systems: how latent computation can be compressed, aligned, guided, steered, switched, or iteratively refined. The *formalizing* class concerns representational and

Table 1: Multi-level taxonomy of implicit reasoning research. Papers are placed according to their primary contribution; several have secondary roles discussed in the text.

Top-level class	Subclass	Main question	Papers
Understanding latent reasoning	Latent-state observability & probing	What reasoning variables are already encoded before or during generation?	Yang et al. (2025); Afzal et al. (2025); Zhang et al. (2025); Xu et al. (2026).
Understanding latent reasoning	Reasoning dynamics & geometry	How does reasoning move through representation space, and what geometry does it induce?	Shalev et al. (2024); Zhou et al. (2026); Kumar et al. (2026).
Building latent reasoning systems	CoT compression & semantic alignment	Can verbal rationales be shortened, internalized, or replaced while preserving answer quality and semantic fidelity?	Hao et al. (2025); Shen et al. (2025); Su et al. (2025); Tan et al. (2025); Wei et al. (2025); Jiang et al. (2026); He et al. (2025).
Building latent reasoning systems	Learned latent reasoning languages	Can models learn a compact non-verbal language or codebook for reasoning?	Ramji et al. (2026).
Building latent reasoning systems	Guidance, steering & modular controllers	Can auxiliary modules, other models, or optimization objectives generate, refine, or steer latent reasoning states?	Kang et al. (2026); Zou et al. (2026); Kong et al. (2026); Chen et al. (2026); Wang et al. (2026a); Wang et al. (2026b).
Building latent reasoning systems	Iterative latent computation & switching	Can extra recurrent, residual, or switch-based latent computation replace longer textual thinking?	Geiping et al. (2025); Kohli et al. (2026); Saunshi et al. (2025); Zelikman et al. (2024); Shi et al. (2026); Lv et al. (2026).
Formalizing latent reasoning	Distributional & formal reasoning	When can a latent state represent many reasoning paths, and what are the formal limits of latent thought?	Zhu et al. (2025); Xu & Sato (2026).

computational limits: when latent states can carry multiple paths or simulate computations more efficiently than textual chains.

This organization also highlights a field-wide progression. Early work decoded hidden evidence of reasoning; later work engineered latent computation as a controllable interface; recent work adds controllers, alignment objectives, and formal accounts of when latent reasoning has a genuine computational advantage. The taxonomy should therefore be interpreted as a map of primary roles rather than as a claim that each paper belongs to only one conceptual neighborhood.

5 Comparative Synthesis: Method Families and Evaluation Axes

The taxonomy groups papers by the primary role of the latent state. This section explains how the groups should be compared. Benchmark scores should not be treated as interchangeable unless the task family, model family, decoding budget, and latency accounting are comparable. A fair comparison should specify what object is changed, what failure mode is addressed, and which benchmark family supports the claim.

The first part of the literature studies what hidden computation already contains. Observability papers compare probe targets, causal interventions, and localization of reasoning variables. Geometry and trajectory papers compare how states move, whether pivots can be identified, and whether continuous transitions can replace parts of a textual trace (Yang et al., 2025; Zhou et al., 2026). The next part builds systems. Compression and semantic-alignment methods are most comparable when they share math, symbolic, QA, or NLP reasoning benchmarks and report both accuracy and cost. Latent-language methods add a different axis: whether the learned code is reusable, controllable, or interpretable.

Controller-oriented methods should be compared by whether they improve the direction, stability, or allocation of latent computation. This includes latent diffusion, flow-based trajectory sampling, latent-vector refinement, large-to-small guidance, gradient-based hidden-state optimization, and explicit-latent switching. Iterative and switch-based methods should be evaluated under explicit token, wall-clock, recurrence, latent-step, or switch-count budgets, because their central claim is not only higher accuracy but better use of test-time

compute (Shi et al., 2026; Lv et al., 2026). Distributional and theoretical papers are best compared on synthetic graph, DAG, or circuit-style tasks that isolate whether latent states can represent multiple paths or parallel algorithmic structure more efficiently than text.

Benchmark overlap is strongest inside these lines rather than across them. Compression and switching methods often use math, STEM, coding, commonsense, and general-reasoning tasks. Recurrent-depth and looped-transformer papers are most directly comparable on controlled graph or knowledge-composition tasks, where hop count and depth extrapolation can be varied. Diagnostic papers should be compared through probe accuracy, intervention effects, and localization rather than final-answer accuracy alone. Formal papers use synthetic settings because their goal is to isolate computational resources rather than to maximize benchmark scores.

The narrative that follows is therefore not chronological. Hidden-state diagnostics establish that reasoning information can be internal; trajectory work studies how such information changes; constructive systems turn latent computation into a controllable interface; and formal work asks which advantages are fundamental. Each thematic section below uses this progression to compare methods within the appropriate benchmark and methodological context.

6 Understanding Latent Reasoning I: Latent-State Diagnostics and Reasoning Observability

The first top-level class in our taxonomy is *understanding latent reasoning*. Its first subclass is observational: what can be read from internal states before a model has finished speaking? Diagnostic papers do not necessarily change the inference algorithm. Instead, they provide evidence that generated text is only a partial trace of the underlying computation (Yang et al., 2025; Afzal et al., 2025; Zhang et al., 2025; Xu et al., 2026).

Latent multi-hop variables. Yang et al. (2025) ask whether LLMs internally perform multi-hop factual reasoning. In two-hop queries, a correct answer often requires a bridge entity. Their work probes whether this bridge is represented in hidden states and whether that representation is related to the final answer. The important contribution is methodological as much as empirical: latent reasoning can be studied by selecting a hidden variable that should exist if the model is composing facts, and then asking where and when that variable appears.

Knowing before saying. Afzal et al. (2025) shift the target from an intermediate entity to a success signal. Rather than asking what entity is encoded, they ask whether model states predict whether a CoT trajectory will eventually succeed. Such a success signal can support early stopping, re-sampling, routing, or verification before the model spends a long token budget. It also reframes reasoning as a process whose future reliability may be partially encoded before completion.

Self-verification in hidden states. Zhang et al. (2025) study whether reasoning models know when they are right. This line of work is closely related to uncertainty estimation, but with a stronger semantic target: correctness rather than only likelihood. The resulting view is that hidden activations can contain meta-reasoning information, such as whether the current answer or chain is likely reliable.

Latent planning horizon. Xu et al. (2026) refine earlier claims about hidden planning by introducing TELE-LENS, a probing method that predicts future tokens, final answers, and reasoning length from CoT hidden states. Their central finding is myopic rather than globally planned reasoning: hidden states often support local transitions and sometimes contain early coarse answer signals, but precise final-answer information on explicit compositional tasks tends to emerge near the end of reasoning. The result cautions against overinterpreting early decodability as evidence of a complete latent plan.

From states to dynamics. Diagnostic studies establish that reasoning information can be present internally, but they also reveal an important limitation. A static hidden state may encode a bridge entity, success

likelihood, or answer gist without explaining how the model transforms one reasoning state into the next. This motivates the dynamic and geometric perspective discussed next.

Comparison across diagnostic objectives. These papers form a progression from *what is encoded* to *when it becomes usable*. Latent MH focuses on an intermediate semantic variable, the bridge entity; Know-Before-Saying moves to a trajectory-level success signal; Self-Verification asks whether correctness is represented; and TELE-LENS tests how far future tokens, answers, and reasoning lengths can be decoded from the current CoT state (Yang et al., 2025; Afzal et al., 2025; Zhang et al., 2025; Xu et al., 2026). Their benchmarks do not always match exactly, so the fair comparison is not final-answer accuracy. The comparable quantities are probe target, timing of the signal, and whether the signal can support interventions such as early stopping, rerouting, resampling, or switching to latent computation.

7 Understanding Latent Reasoning II: Reasoning Dynamics and Geometry

The second subclass within the *understanding* class studies reasoning as motion. Instead of only asking whether a hidden state contains a variable, dynamic work asks how representations evolve across layers, generated tokens, recurrent iterations, or latent steps (Shalev et al., 2024; Zhou et al., 2026; Kumar et al., 2026; Xu et al., 2026).

Distributional paths in multi-hop reasoning. Shalev et al. (2024) model multi-hop reasoning as a distributional process. Rather than assuming that a model commits to a single bridge, they analyze how category distributions over possible entities or answers transform across the reasoning process. This anticipates multi-path latent reasoning: the model's internal state may encode a distribution over possibilities rather than a single textual path.

Flowing logics in representation space. Zhou et al. (2026) treat reasoning trajectories as geometric objects. Their central idea is that the direction and curvature of representation-space motion can capture logical progress. This complements probing: a probe may tell us what is available at a point, while geometry tells us how the model moves between points.

Geometric latent reasoning. Kumar et al. (2026) make the geometric view constructive. GLR treats a textual CoT as a discrete trajectory through the pretrained token-embedding space, then learns a transition head that predicts continuous displacement vectors. At inference, an initial segment of explicit reasoning is replaced by a fixed number of continuous embedding-space steps, after which normal token decoding resumes. The method shows that continuous trajectories can induce much shorter generations without an explicit length objective.

Planning dynamics and pivots. The latent-horizon results of Xu et al. (2026) also belong here because they show that uncertainty is concentrated in sparse pivot positions. Their "wooden barrel" principle suggests that global averages over all CoT tokens can dilute the few local transitions that determine whether a chain succeeds. This dynamic view connects hidden-state analysis to practical controllers: instead of scoring every token equally, a system can focus on the most informative reasoning pivots.

Why geometry matters for systems. Geometry and dynamics provide design constraints for latent reasoning systems. If latent states drift away from the embedding manifold, accuracy can collapse. If multiple frontiers are encoded in superposition, forcing a discrete token at every step may be inefficient. If only a few pivots determine uncertainty, token-level averages are poor controllers. New systems increasingly incorporate such structure through transition heads, auxiliary decoders, diffusion denoisers, priors, or flow objectives.

Comparison: from static probes to trajectory models. The dynamic papers ask what comes after probing. Distributional Reasoning shows that internal states may retain distributions over possible entities or answers; Geometry of Reasoning interprets logical progress as representation-space motion; GLR turns this geometry into a constructive transition mechanism that can shorten explicit generations; and TELE-LENS

adds a pivot view, showing that some positions matter more than others for future uncertainty (Shalev et al., 2024; Zhou et al., 2026; Kumar et al., 2026; Xu et al., 2026). The matching benchmark question is whether trajectory information predicts or improves multi-step reasoning beyond a static state probe. When benchmarks differ, the common comparison is trajectory fidelity: whether the method identifies a meaningful path, a useful transition direction, or a reliable pivot for control.

8 Building Latent Reasoning Systems I: Explicit-to-Implicit CoT Compression

The first major subclass within the *building latent reasoning systems* class is compression and semantic alignment. Its goal is to reduce the cost of textual reasoning without losing the benefits of intermediate computation. Compression methods differ in the representation they use, but they share a premise: many CoT tokens are useful as computational scaffolding, not because they must be communicated word by word (Hao et al., 2025; Shen et al., 2025; He et al., 2025).

Continuous hidden-state thoughts. COCONUT replaces some textual reasoning tokens with continuous hidden states that are fed back as subsequent inputs (Hao et al., 2025). This provides a direct way to perform reasoning in latent space and motivates later work on continuous CoT. However, it also illustrates a central training problem: latent states must be introduced gradually or with sufficient supervision, otherwise the model may not know how to use them as stable inputs.

Self-distillation into continuous thoughts. CODI addresses the performance gap between explicit CoT and implicit CoT by jointly training a teacher behavior that uses explicit CoT and a student behavior that generates a small number of continuous thoughts (Shen et al., 2025). Hidden-state alignment transfers reasoning information from the teacher into the student, shifting the objective from reconstructing each missing word to aligning the reasoning-relevant state at the point where the answer is generated.

Discrete latent-token compression. Token Assorted uses VQ-VAE latent tokens to replace an initial segment of CoT while leaving later textual reasoning visible (Su et al., 2025). The method randomly mixes latent and text tokens during training so that the LLM adapts to an extended vocabulary of previously unseen latent tokens. This hybrid design exposes an efficiency-interpretability tradeoff: early reasoning can be compressed, while later reasoning remains human-readable.

Dynamic compression and reinforcement learning. COLAR compresses consecutive reasoning tokens into dense latent variables and trains a probabilistic latent head to predict compressed embeddings (Tan et al., 2025). Its reinforcement-learning stage rewards both correctness and compactness, allowing the model to explore diverse latent reasoning paths and exploit shorter successful ones. The method marks a shift from fixed compression toward dynamic-speed reasoning.

Stabilizing implicit tokens. SIM-COT identifies a failure mode of implicit CoT: as the number of latent tokens grows, representations can homogenize and lose semantic diversity (Wei et al., 2025). Its auxiliary decoder provides step-level supervision, aligning each implicit token with an explicit reasoning step during training while removing the decoder at inference. The auxiliary decoder provides an important stabilization pattern for compressed reasoning systems.

Semantic alignment and token-generation speed. SEMCOT adds a second efficiency axis to implicit CoT: not only how many implicit tokens are produced, but also how expensive each implicit token is to generate (He et al., 2025). It uses a contrastively trained sentence transformer to measure semantic alignment between implicit reasoning and ground-truth explicit reasoning, then trains a lightweight implicit reasoning generator through knowledge distillation. The contribution treats hidden reasoning as a semantic object that should preserve the meaning of a valid rationale, rather than merely a short vector sequence optimized for the final answer.

Latent reasoning tuning. LRT challenges the need to generate full reasoning trajectories by showing that fragmented reasoning can still support correct answers, then replacing explicit token-by-token reasoning

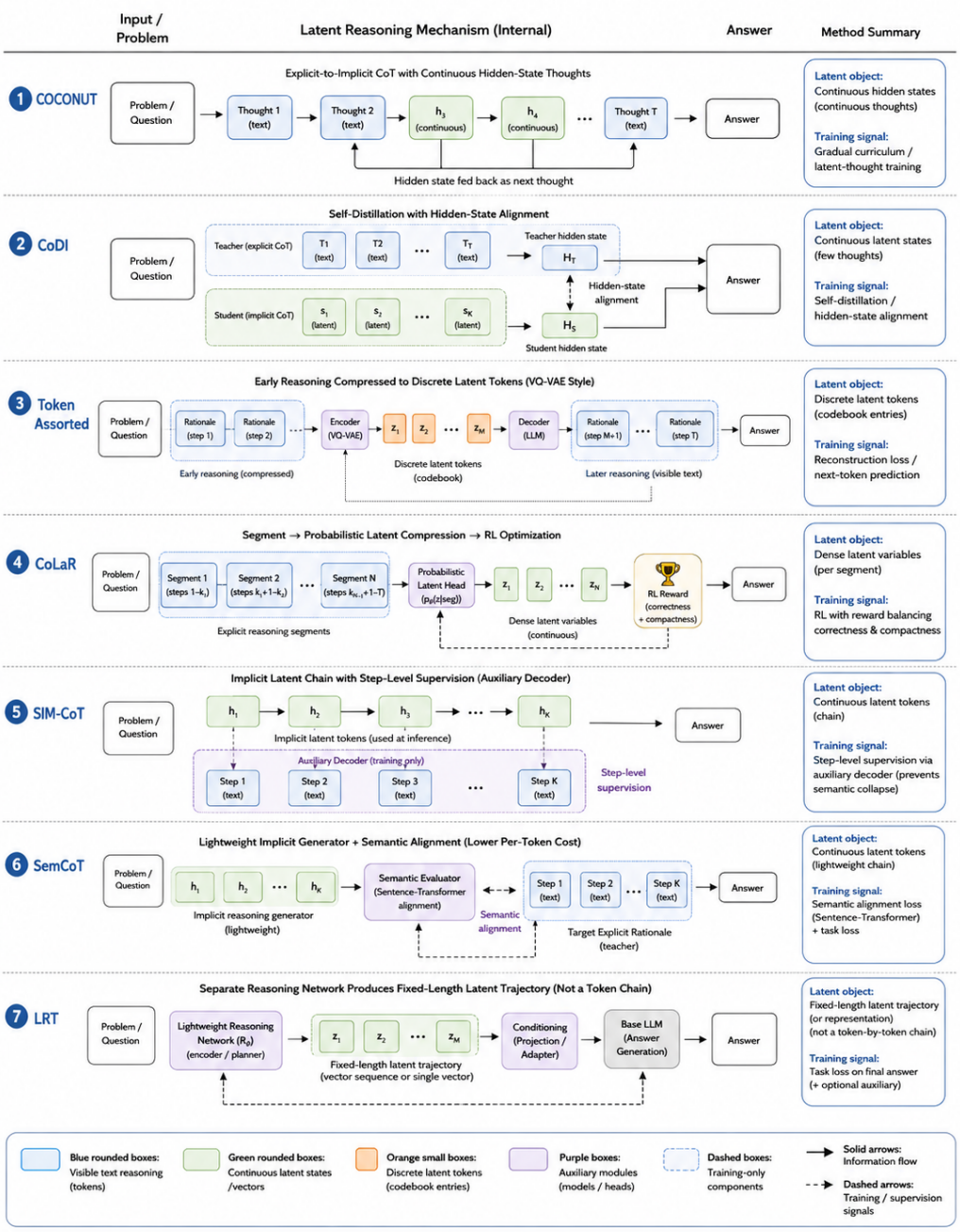

Figure 3: Architectural comparison of representative methods for explicit-to-implicit CoT compression. The figure highlights the internal latent object used by each method (continuous states, discrete latent tokens, segment-level latent variables, or fixed latent trajectories), together with the training mechanism that stabilizes or aligns latent reasoning.

with a compact latent trajectory produced by a lightweight reasoning network (Jiang et al., 2026). Unlike methods that sample latent tokens autoregressively, the reasoning network directly computes a fixed-length latent representation that conditions the base LLM's answer generation.

Compression as structured substitution. The progression from COCONUT to SEMCOT and LRT shows that effective compression is a structured substitution rather than simple token deletion. Successful

Table 2: Comparison within CoT compression and semantic alignment. The rows show the limitation each paper addresses and the benchmark dimension that should be compared when tasks overlap.

Method	What it tries to fix	How it differs	Meaningful comparison
COCONUT	Textual CoT is long and discretized.	Feeds continuous hidden states back as thoughts.	Accuracy versus number of latent/text steps on math and symbolic reasoning.
CoDI	Direct latent thoughts can lag behind explicit CoT.	Uses self-distillation and hidden-state alignment.	Accuracy retention when explicit rationales are replaced by few continuous states.
Token Assorted	Pure continuous thoughts are hard to audit.	Uses VQ-VAE-style latent tokens mixed with text.	Compression ratio, codebook usage, and accuracy on overlapping reasoning tasks.
CoLaR	Fixed compression cannot adapt to task difficulty.	Learns probabilistic latent compression with RL for compactness.	Accuracy–length tradeoff and robustness under different compression budgets.
SIM-CoT	Many implicit tokens can collapse semantically.	Adds step-level auxiliary supervision during training.	Stability as latent-token length grows.
LRT	Full token-by-token rationales are not always needed.	Uses a lightweight reasoning network to produce compact latent trajectories.	Whether fixed-length latent conditioning matches or beats fragmented/explicit reasoning.
SemCoT	Shorter implicit CoT can still be slow or semantically misaligned.	Uses a semantic-alignment evaluator and lightweight implicit-token generator.	Accuracy, latency per implicit token, and semantic similarity to explicit reasoning.

methods add structure: curricula, self-distillation, codebooks, probabilistic heads, auxiliary decoders, semantic-alignment losses, lightweight generators, geometric constraints, or learned reasoning modules. The recurring lesson is that latent reasoning must be grounded strongly enough to avoid drift, collapse, semantic mismatch, or loss of operator-level information (Hao et al., 2025; Wei et al., 2025; He et al., 2025; Jiang et al., 2026).

Comparison across compression objectives. The compression literature moves from "use fewer tokens" to "use fewer tokens while preserving the right meaning." COCONUT establishes continuous thoughts; CoDI, SIM-CoT, and SemCoT add stronger alignment signals; Token Assorted and Abstract-CoT-like ideas make the latent channel more discrete; CoLaR makes compression adaptive; and LRT separates compact reasoning from final answer realization (Hao et al., 2025; He et al., 2025; Jiang et al., 2026). Benchmarks are most comparable when papers evaluate on shared math, symbolic, QA, or NLP reasoning families and report both accuracy and cost. Without cost accounting, a latent method may simply move computation from visible tokens into hidden updates or auxiliary modules.

9 Building Latent Reasoning Systems II: Learned Latent Reasoning Languages

The second subclass within the *building* class does not only compress CoT into vectors; it creates a new non-verbal medium that the model can emit and reuse. These learned latent languages can be discrete, partially interpretable, or softly constrained, but they all treat latent reasoning as its own communication channel (Ramji et al., 2026; Su et al., 2025).

Abstract Chain-of-Thought. Ramji et al. (2026) propose Abstract-CoT, where the model emits a short sequence of reserved abstract tokens inside special delimiters before producing the answer. The tokens are not natural language and are initially meaningless. The paper addresses this cold-start problem with a policy-iteration warm-up loop: bottlenecked supervised fine-tuning first forces answer generation to attend to abstract tokens rather than verbal CoT, then self-distillation trains the model to generate abstract sequences from the input alone. A warm-started reinforcement-learning phase further optimizes abstract traces under constrained decoding. The method thereby learns an abstract reasoning language that can reduce reasoning-token usage substantially while retaining performance across math, instruction-following, and multi-hop tasks.

Codebooks versus reserved vocabularies. Token Assorted and Abstract-CoT both use discrete non-verbal tokens, but their semantics are learned differently. Token Assorted learns latent tokens through a VQ-VAE that compresses textual traces (Su et al., 2025); Abstract-CoT introduces a reserved vocabulary

and trains the model to use it as a compact reasoning language (Ramji et al., 2026). This difference matters. A VQ-VAE codebook is tied to reconstruction of teacher traces, while a reserved abstract vocabulary may discover alternative reasoning conventions.

Semantic diversity and supervision. SIM-CoT shows why latent languages need supervision: implicit tokens can collapse into homogeneous representations if they are not forced to carry distinct step-level information (Wei et al., 2025). CoLaR provides another direction by using a probabilistic latent head and RL to maintain diversity over compressed reasoning paths (Tan et al., 2025). Together these papers suggest that a latent language is useful only if it has enough capacity, diversity, and grounding to represent distinct reasoning functions.

Interpretability tradeoffs. Learned latent languages are less readable than verbal CoT but more inspectable than unconstrained continuous vectors. Their tokens can be counted, constrained, ablated, permuted, truncated, or analyzed for frequency patterns. Abstract-CoT's observation of power-law token usage is especially suggestive: a model may develop reusable abstract symbols, even when those symbols are not human words. This creates a new interpretability problem: how should we audit a reasoning language that is discrete and structured, but not semantically transparent?

Comparison: from compression to reusable symbols. The latent-language line differs from ordinary compression because the goal is not only to shorten one rationale, but to create a reusable non-verbal medium. Abstract-CoT is the clearest example: it trains reserved abstract tokens to carry reasoning information before verbal output (Ramji et al., 2026). Token Assorted and SemCoT are useful comparison points even though they are placed under compression: Token Assorted asks whether a learned codebook can replace parts of CoT, while SemCoT asks whether implicit tokens remain semantically aligned with explicit reasoning (Su et al., 2025; He et al., 2025). The matching benchmarks should therefore test transfer of latent symbols across task families, code usage diversity, and whether decoded or probed meanings remain stable, not only answer accuracy.

10 Building Latent Reasoning Systems III: Guidance, Steering, and Modular Controllers

A third subclass within the *building* class introduces mechanisms that generate, refine, steer, or transfer latent reasoning states. Instead of relying only on the base LLM to transform one hidden state into the next, these methods add an explicit latent mechanism–a reasoning network, a large-model guidance channel, a diffusion model, a flow sampler, a contrastive refinement rule, or a representation-optimization objective–that participates directly in the reasoning process (Kang et al., 2026; Chen et al., 2026; Wang et al., 2026b).

Latent reasoning networks. LRT introduces a lightweight reasoning network that maps a question to latent reasoning tokens used to condition a reasoning LLM (Jiang et al., 2026). The base model need not autoregressively generate a long explicit trajectory. Instead, the auxiliary network produces compact latent representations in a single forward pass, allowing a non-intrusive switch between explicit and latent modes. This makes LRT especially relevant for efficient deployment: reasoning compute is moved from serial decoding into a small learned module.

Large-to-small latent guidance. Latent Guidance decouples high-level cognitive planning from low-level linguistic realization (Chen et al., 2026). A large model acts as an implicit thinker that compresses its solution strategy into compact latent guidance vectors, while a smaller model acts as an explicit executor that generates a concise reasoning chain conditioned on those vectors. The dual-loss objective combines task performance with reconstruction of the original reasoning chain, encouraging the latent guidance to preserve a high-fidelity cognitive plan. This paper shifts latent reasoning from single-model acceleration to cross-model capability transfer.

Inference-time rethinking. Kong et al. (2026) propose latent thought vectors that separate declarative reasoning content from procedural verbalization. A prior encoder maps noise into a learned manifold of valid

reasoning patterns, and a decoder verbalizes a trace conditioned on the latent thought. At inference time, a Gibbs-style loop alternates between generating a candidate trace and optimizing the latent variable to better explain that trace. This turns reasoning into test-time optimization over a latent manifold rather than a single irreversible token stream.

Post-training latent refinement. Efficient post-training refinement treats latent reasoning as a trajectory that can be improved after the base latent reasoner has been trained (Wang et al., 2026a). The method uses contrastive reasoning feedback to infer an improvement direction by comparing strong and weak model embeddings, then applies residual embedding refinement to stabilize updates across steps. The resulting procedure adds correction and memory to latent reasoning without changing model parameters.

Gradient-based representation optimization. GBRO elicits CoT behavior from base LLMs by optimizing hidden states under a probabilistic conditional-generation view (Wang et al., 2026b). A likelihood term pushes the representation toward a reasoning condition, while a prior regularizer keeps the hidden state near the model's natural language manifold. This reframes activation steering as constrained posterior optimization rather than unbounded vector arithmetic, and it connects latent reasoning to controllable generation.

Latent diffusion reasoners. LaDiR uses a VAE to encode reasoning steps into latent thought tokens and then trains a blockwise latent diffusion model over them (Kang et al., 2026). Diffusion provides iterative semantic refinement and a natural way to allocate more denoising steps at test time. Its diversity-guidance mechanism pushes sampled latent trajectories apart, enabling exploration of multiple solution regions rather than repeating similar autoregressive chains.

Latent Thought Flow. LTF models reasoning as variable-length continuous trajectories and trains a sampler to match a reward-induced posterior over answer quality and computation cost (Zou et al., 2026). It instantiates this with a continuous GFlowNet, an entropy-weighted subtrajectory balance objective, and a reference-prior regularizer. The conceptual contribution is distribution matching rather than maximum-reward collapse: probability mass should be assigned to many accurate and efficient latent paths, not only to one best path.

Modularity as a design pattern. These methods differ technically, but share a modular principle. Reasoning can be represented by a latent object that is generated or improved before answer decoding. The object may be a fixed-length latent chain, a large-model guidance vector, an optimized hidden state, a diffusion block, a residual-refined embedding, or a GFlowNet trajectory. This modularization supports explicit tradeoffs among speed, diversity, controllability, and accuracy, but it also creates new failure modes: module-base mismatch, latent drift, posterior collapse, unstable steering, or misalignment between latent reward and final-answer behavior (Chen et al., 2026; Wang et al., 2026a;b; Zou et al., 2026).

Comparison: modular methods lead to controlled latent search. The modular line starts from the observation that a latent state need not be produced only by ordinary autoregressive decoding. LaDiR and LTF generate or sample latent trajectories; Latent Thought Vectors and PTR refine a latent object after an initial state; Latent Guidance transfers a plan across model scales; and GBRO optimizes hidden states under a prior so that steering does not destroy language quality (Kang et al., 2026; Chen et al., 2026; Wang et al., 2026b). The comparable benchmark dimension is controller value: how much accuracy, diversity, or efficiency is gained per extra controller step, and whether the controller remains stable under domain shift.

11 Building Latent Reasoning Systems IV: Iterative Latent Computation and Explicit-Latent Switching

The fourth subclass within the *building* class studies how to scale reasoning by spending more hidden computation, or by switching between hidden and visible computation, rather than simply generating longer textual chains. This line includes recurrence, looped depth, repeated hidden-state refinement, entropy-based

Table 3: Comparison within guidance, steering, and modular controllers.

Method	Controller object	Main difference	Matching benchmark lens
LaDiR	Diffusion latent blocks	Iteratively denoises and diversifies latent thoughts.	Best-of-k, diversity, planning/code/math accuracy, and denoising cost.
LTF	Variable-length latent trajectories	Samples from a reward-induced posterior with a GFlowNet.	Reward-quality tradeoff, trajectory diversity, and adaptive stopping.
Latent Thought Vectors	Optimized latent vector	Uses generate–reflect refinement at inference.	Math accuracy versus refinement iterations.
Latent Guidance	Large-model guidance vectors	Transfers high-level plan from large implicit thinker to small executor.	Small-model lift, cross-scale generalization, and guidance reconstruction fidelity.
PTR	Post-training embedding refinement	Uses contrastive strong/weak feedback and residual embedding updates.	Improvement over latent-only baselines and stability across refinement steps.
GBRO	Hidden-state posterior optimization	Steers base models toward CoT with likelihood-prior regularization.	CoT elicitation, text quality, and manifold-preserving steering.

switching, uncertainty-based mode selection, and other mechanisms that increase effective reasoning depth without proportionally increasing textual length (Geiping et al., 2025; Shi et al., 2026; Lv et al., 2026).

Recurrent-depth language models. Geiping et al. (2025) scale test-time compute by iterating a recurrent block in latent space. A prelude maps inputs into a latent state, a shared recurrent block updates that state repeatedly, and a coda decodes the result. Because the number of recurrent iterations can be increased at inference, the model can spend more compute without generating longer chains. This frames latent reasoning as a third scaling axis alongside parameter count and output length.

Implicit multi-hop generalization. Kohli et al. (2026) study recurrent-depth transformers in controlled knowledge-graph composition tasks. They show that recurrence helps systematic generalization, where facts not composed during training must be combined at test time, and depth extrapolation, where the model must handle more hops than it saw during training. A key result is that increasing inference-time recurrence can unlock deeper reasoning, although excessive recurrence can cause overthinking.

Looped transformers as latent thought generators. Saunshi et al. (2025) argue that many reasoning tasks require depth more than new parameters. A small block looped many times can match or approach an iso-FLOP deeper model on synthetic reasoning and can show favorable reasoning behavior in language-modeling settings despite worse perplexity. They also connect looping to CoT by arguing that looped models implicitly generate latent thoughts and can simulate CoT-like computation through repeated application of the same block.

Silent rationales before ordinary text. Quiet-STaR trains a model to generate private rationales at token positions in ordinary text, then learn which thoughts improve prediction of future text (Zelikman et al., 2024). Although its rationales are textual during training, the key idea is general: thinking can be learned from next-token prediction itself, not only from curated question-answer rationales. It therefore connects implicit reasoning to the broader language-modeling objective.

Entropy-guided switch thinking. SwiReasoning is a training-free framework that alternates between explicit and latent thinking based on block-wise confidence estimated from entropy trends in next-token distributions (Shi et al., 2026). Rising confidence triggers explicit reasoning to consolidate a path, while uncertainty motivates latent reasoning to preserve and explore alternatives. A switch-count controller limits the number of transitions, providing early-answer checkpoints and reducing overthinking. This makes switching itself a controller for balancing exploration, exploitation, and token efficiency.

Semantic residual refinement and dynamic mode choice. DyLaR also uses a training-free explicit-latent switch, but focuses on semantic fidelity when latent states are projected back into the input space (Lv et al., 2026). Its Semantic Residual Refinement module iteratively integrates residual information between

Table 4: Comparison within iterative and switch-based latent computation.

Method	Extra-compute mechanism	What it resolves	Matching benchmark lens
Recurrent-Depth	Repeated latent block updates	Adds depth without longer visible CoT.	Accuracy versus recurrence count and compute.
Loop-Think	Recurrent-depth transformers for hop composition	Tests systematic and depth generalization.	Knowledge-graph hop extrapolation.
Looped Latent Thoughts	Reused transformer blocks as latent thoughts	Gives depth without new parameters.	Synthetic reasoning and depth-sensitive tasks.
Quiet-STaR	Private rationales during language modeling	Learns when hidden thoughts help next-token prediction.	Language modeling plus downstream reasoning transfer.
SwiReasoning	Entropy-guided explicit/latent switching	Balances latent exploration and explicit consolidation.	Math/STEM/code/general accuracy and token efficiency under budgets.
DyLaR	Semantic residual refinement plus uncertainty switching	Keeps latent inputs semantically faithful while switching modes.	Knowledge-intensive and STEM tasks, accuracy versus token reduction.

hidden states and soft token-embedding projections, producing latent inputs that better approximate the full continuous semantics than simple interpolation. A dynamic uncertainty policy then favors explicit reasoning when the model is confident and latent exploration when the problem is ambiguous.

Refinement at inference time. Inference-Time Rethinking, LaDiR, and post-training latent refinement all make test-time computation iterative. Rethinking optimizes a latent thought vector through generate-reflect cycles (Kong et al., 2026); LaDiR performs multiple diffusion denoising steps over latent thought tokens (Kang et al., 2026); and post-training refinement updates latent embeddings with contrastive feedback and residual integration (Wang et al., 2026a). All three replace one-shot commitment with a process that can revisit or correct a reasoning state.

The control problem. Test-time scaling raises a controller question: how many latent iterations or explicit-latent switches should be spent? Too little compute under-solves the problem; too much can waste time, diffuse probability mass, or degrade predictions. This problem appears in recurrent-depth overthinking, SwiReasoning's switch cap, DyLaR's uncertainty policy, GLR's non-monotonic latent-step budget, CoLaR's dynamic compression, and LTF's adaptive stopping (Geiping et al., 2025; Shi et al., 2026; Tan et al., 2025). The next generation of systems will likely combine hidden uncertainty estimates, entropy trends, semantic residuals, and latent-horizon pivots for compute allocation.

Comparison: iterative methods lead to adaptive compute allocation. The recurrent papers show that hidden depth can substitute for some textual length, but they also expose overthinking and budget selection as central problems (Geiping et al., 2025; Kohli et al., 2026; Saunshi et al., 2025). Quiet-STaR broadens the idea by learning private thoughts from ordinary text prediction (Zelikman et al., 2024). SwiReasoning and DyLaR make the next step explicit: instead of choosing all-text or all-latent reasoning, they switch between modes using entropy or uncertainty, so the benchmark comparison becomes Pareto performance under token, time, and switch-count budgets (Shi et al., 2026; Lv et al., 2026). This line leads naturally to learned or training-free routers that decide when to verbalize, when to remain latent, and when to stop.

12 Formalizing Latent Reasoning I: Distributional, Parallel, and Multi-Path Latent Reasoning

The first subclass within the *formalizing latent reasoning* class studies latent reasoning as a distribution over possibilities rather than a single serial trace. Textual CoT usually serializes one reasoning path. Latent reasoning can instead encode distributions, superpositions, or sets of possible trajectories. This provides one of the strongest conceptual arguments for reasoning beyond text (Shalev et al., 2024; Zhu et al., 2025; Shi et al., 2026).

From distributions over entities to distributions over paths. Shalev et al. (2024) show that multi-hop reasoning can be understood as transformations over distributions rather than deterministic bridge selection. This perspective helps explain why a hidden state may support multiple possible completions before a final answer is chosen. It also foreshadows later methods that explicitly preserve diverse latent trajectories.

Superposition in continuous thought. Zhu et al. (2025) provide a theoretical and empirical account of continuous thoughts as superposition states. In graph reachability, a continuous thought vector can encode multiple search frontiers at once, supporting a breadth-first style computation. A discrete token, by contrast, collapses the state to one branch. This gives a concrete reason why continuous latent thought can be more efficient than discrete CoT for graph search.

Diffusion diversity. LaDiR uses latent diffusion not only for refinement but also for diversity (Kang et al., 2026). By applying repulsive diversity guidance during denoising, it encourages different samples to explore different regions of latent reasoning space. Such diversity is useful in tasks such as planning or code generation, where many trajectories can lead to a correct answer and best-of-k performance depends on diversity rather than only local likelihood.

Reward-proportional latent flows. LTF makes the distributional goal explicit by training a continuous GFlowNet over variable-length latent trajectories (Zou et al., 2026). Instead of optimizing a deterministic best path, it learns a distribution proportional to an answer-quality and computation-cost reward. This matters because reasoning problems often have several concise and correct trajectories; collapsing onto one mode can reduce robustness and exploration.

Switching as distribution management. SwiReasoning and DyLaR provide a deployment-oriented counterpart to formal distributional arguments (Shi et al., 2026; Lv et al., 2026). Pure latent reasoning can maintain multiple implicit hypotheses, but this can diffuse probability mass and delay convergence. Explicit reasoning collapses toward a single path, which can improve readability and commitment but may discard alternatives too early. Dynamic switching therefore manages the distributional state of reasoning: latent steps preserve or re-explore alternatives, while explicit steps consolidate a path when confidence is high.

Parallelism as computation, not only sampling. Distributional and superpositional reasoning are not only sampling tricks. They change what a single intermediate state can represent. In explicit CoT, a token usually commits to one surface form. In latent thought, a vector or block can encode multiple alternatives, soft frontiers, or a posterior over paths. This corresponds to *algorithmic* or *representation-level* parallelism: several possibilities are represented or updated in one latent object. It does not automatically imply wall-clock hardware parallelism, because a practical system may still spend many forward passes, denoising steps, recurrent iterations, or controller calls (Zhu et al., 2025; Xu & Sato, 2026). This distinction explains why latent reasoning is attractive for search-like problems, but also why efficiency must be measured rather than assumed.

Comparison across multi-path objectives. The distributional line asks how long a model should preserve alternatives before committing. Distributional Reasoning studies distributions over entities; Superposition argues that a continuous state can encode multiple graph-search frontiers; LaDiR and LTF operationalize diversity through diffusion sampling and reward-proportional trajectory distributions; and SwiReasoning/DyLaR show that practical systems may need to switch back to explicit text to consolidate a path (Shalev et al., 2024; Zhu et al., 2025; Kang et al., 2026). Benchmarks match when they expose multiple possible paths–for example graph reachability, multi-hop reasoning, planning, coding, and best-of-k settings. The comparison should include diversity, calibration, and final commitment accuracy, not only pass@1.

13 Formalizing Latent Reasoning II: Formal Limits and Theoretical Separation

The second subclass within the *formalizing* class clarifies why text and latent thought are complementary. The question is not simply which is better, but which computational resources each paradigm supplies (Zhu et al., 2025; Xu & Sato, 2026; Saunshi et al., 2025).

Continuous thoughts for graph reachability. Zhu et al. (2025) prove that a two-layer transformer with continuous CoT can solve directed graph reachability in a number of steps proportional to the graph diameter, because each continuous thought can encode several search frontiers simultaneously. This provides a formal version of the intuition that continuous latent states can perform parallel breadth-first search, while discrete CoT must sequentialize or collapse search choices.

CoT versus latent thought. Xu & Sato (2026) compare CoT, Coconut-style latent thought, and looped transformers through computation graphs and circuit complexity. Their results show that latent thought can evaluate DAG-like computations with depth-scaled iterations, while CoT naturally simulates node-by-node computation with size-scaled steps. This gives latent thought an advantage for parallelizable deterministic computation. However, they also identify a separation in favor of CoT: stochastic token generation supports approximate counting and sampling in settings where deterministic latent thought is limited.

Looped transformers and depth without parameters. Saunshi et al. (2025) support a related claim: many reasoning problems need iterative depth, not necessarily many distinct parameters. A looped transformer can reuse a block to create effective depth and can, in theory, simulate CoT reasoning. This ties the empirical success of recurrent-depth models to a formal view of latent thoughts as repeated hidden computations.

Theory for steering and guidance. Recent controller-oriented papers add another formal question: how can a latent state be moved without leaving the model's useful representation manifold? Latent-Guided Reasoning analyzes latent capacity and reconstruction fidelity for transferring a large model's plan to a smaller executor (Chen et al., 2026). Gradient-based representation optimization formulates hidden-state steering as posterior optimization with a likelihood term and a prior regularizer (Wang et al., 2026b). These analyses complement expressivity theory by focusing on controllability, fidelity, and linguistic coherence.

Implications for system design. Theory supports a hybrid architecture. If a task requires parallel evaluation, graph search, or repeated deterministic updates, latent thought and recurrence may be efficient. If a task benefits from stochastic exploration, sampling, or communicable intermediate memory, explicit CoT may remain valuable. A practical system should therefore not ask whether to abolish text, but how to route computation between textual and latent channels, and how to constrain latent updates so that extra hidden computation remains useful (Shi et al., 2026; Lv et al., 2026; Wang et al., 2026b).

Comparison: theory explains which empirical gains should transfer. The theoretical papers are not competitors on one benchmark; they clarify which empirical claims should generalize. Superposition explains why continuous states can help graph reachability by storing several frontiers; the formal CoT-versus-latent comparison separates depth-scaled latent computation from size-scaled textual simulation and also identifies advantages of stochastic explicit tokens; looped-transformer analysis connects empirical recurrent-depth gains to repeated hidden computation; and controller analyses such as Latent Guidance and GBRO formalize fidelity or prior-preserving steering rather than raw expressivity (Zhu et al., 2025; Xu & Sato, 2026; Wang et al., 2026b). The matching comparison is therefore conceptual: if a benchmark requires parallel deterministic updates, latent/recurrent methods should help; if it requires sampling or human-auditable memory, explicit CoT may remain superior.

14 Evaluation Practices

Implicit reasoning is difficult to evaluate because accuracy alone does not reveal whether a model reasoned latently, compressed a trace, guessed from a shortcut, shifted computation into an auxiliary module, or simply moved cost from decoding into hidden computation. The literature therefore uses several complementary evaluation axes (Hao et al., 2025; He et al., 2025; Shi et al., 2026).

A common metric layer. For a benchmark with N examples, answer accuracy is usually

$$\mathrm{Acc} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\{\hat{y}_i = y_i\}. \tag{3}$$

For latent reasoning, this number should be paired with a normalized cost. If T_{text} is the number of visible reasoning tokens, K_{lat} the number of latent updates, and S_{ctrl} the number of controller, teacher, diffusion, recurrent, or refinement calls, a generic cost model is

$$C = T_{\text{text}} + \lambda K_{\text{lat}} + \mu S_{\text{ctrl}}, \qquad \text{Eff} = \frac{\text{Acc}}{C}. \tag{4}$$

Here λ and μ are implementation-dependent conversion factors, not universal constants. They can represent latency, FLOPs, memory pressure, or monetary cost. This is why token reduction alone is not sufficient evidence of efficiency.

For methods that claim to preserve the meaning of explicit reasoning, a semantic-fidelity metric can compare an explicit rationale r with a decoded or projected latent trace $\tilde{r}(z)$:

$$\text{Align}(z, r) = \frac{e(\tilde{r}(z))^\top e(r)}{\|e(\tilde{r}(z))\|_2 \, \|e(r)\|_2}, \tag{5}$$

where $e(\cdot)$ is an embedding model or task-specific evaluator. This abstraction covers alignment-oriented methods while also making the limitation explicit: high semantic similarity does not by itself prove that the latent state was causally used.

Task accuracy and generalization. Most constructive papers report answer accuracy on math, logic, question answering, code, graph reachability, knowledge-intensive tasks, STEM tasks, planning, or commonsense benchmarks. Accuracy remains necessary but insufficient. For example, a latent method that matches CoT accuracy while using far fewer steps is qualitatively different from one that merely changes the training recipe. Large-to-small guidance also requires cross-capacity evaluation, because the central claim is not only that the executor is accurate, but that latent guidance transfers planning ability from a stronger model (Chen et al., 2026).

Efficiency. Efficiency can mean generated token count, latent step count, total forward passes, wall-clock latency, FLOPs, memory, or communication cost. GLR and Abstract-CoT emphasize shorter generations (Kumar et al., 2026; Ramji et al., 2026); CoLaR and LTF report compression or reasoning-length reductions (Tan et al., 2025; Zou et al., 2026); recurrent-depth methods trade tokens for repeated hidden computation (Geiping et al., 2025; Kohli et al., 2026); SemCoT explicitly measures the cost of generating each implicit token with a lightweight generator (He et al., 2025); and SwiReasoning evaluates token efficiency under constrained budgets while controlling switch count (Shi et al., 2026). Comparisons should state whether latent steps are counted as generation steps and whether vocabulary projection, cache use, auxiliary modules, and controller overhead are included.

Semantic fidelity and alignment. Latent methods need to preserve not only the final answer, but also the semantics of the intermediate reasoning they replace. SemCoT directly optimizes semantic alignment between implicit and explicit reasoning through a contrastive sentence transformer (He et al., 2025). Latent Guidance uses reconstruction pressure so that guidance vectors encode the teacher's full reasoning chain (Chen et al., 2026). DyLaR targets semantic fidelity in training-free latent inputs through Semantic Residual Refinement (Lv et al., 2026). These methods make fidelity an evaluation object rather than treating it as an informal interpretability concern.

Stability and drift. Latent methods introduce failure modes that ordinary CoT does not. Continuous trajectories can drift away from the token-embedding manifold; implicit tokens can homogenize; recurrent models can overthink; purely latent reasoning can diffuse probability mass; diffusion or flow samplers can become miscalibrated; and representation steering can degrade text quality if it leaves the language prior (Wei et al., 2025; Shi et al., 2026; Wang et al., 2026b). Evaluations should include stress tests that vary latent length, recurrence, compression ratio, switch budgets, residual-refinement steps, and domain shift.

Distributional quality. For methods that claim to represent multiple paths, pass@k, diversity, entropy, or reward-proportionality should be measured. LaDiR evaluates diverse latent trajectories through diffusion

Table 5: Benchmark comparison map. Scores should be compared most directly within the same row, because each row shares similar tasks, budgets, or evaluation targets.

Benchmark family	Most relevant method groups	What to compare
Math/STEM/general reasoning	Compression, switching, guidance, latent-vector refinement (Hao et al., 2025; He et al., 2025; Shi et al., 2026)	Accuracy, token count, latency, switch count, latent steps, and semantic fidelity.
Code/planning/best-of-k	Diffusion, flow, switching, distributional search (Kang et al., 2026; Zou et al., 2026; Shi et al., 2026)	Pass@k, diversity, reward quality, and cost per sampled trajectory.
Multi-hop QA and factual composition	Observability, distributional trajectories, recurrent-depth models (Yang et al., 2025; Shalev et al., 2024; Kohli et al., 2026)	Bridge-entity decodability, hop generalization, and intervention effects.
Graph/DAG/synthetic reasoning	Superposition, formal comparisons, looped transformers (Zhu et al., 2025; Xu & Sato, 2026; Saunshi et al., 2025)	Depth scaling, frontiers represented per latent step, and formal separation.
Base-model steering	GBRO and hidden-state manipulation (Wang et al., 2026b)	CoT elicitation rate, reasoning accuracy, and language-quality preservation.

sampling (Kang et al., 2026); LTF evaluates the distribution over variable-length trajectories rather than a single maximum-reward chain (Zou et al., 2026); and SwiReasoning explicitly measures whether switching avoids the accuracy loss of purely latent thinking while preserving efficiency (Shi et al., 2026). A single pass@1 score does not capture these properties.

Causal evidence. Diagnostic papers should distinguish decodability from causal use. Stronger tests include activation editing, bridge-entity swapping, latent-token ablation, pivot perturbation, recurrence intervention, switch-budget ablation, residual-update ablation, or trajectory steering. Constructive methods partly provide causal evidence because changing the latent mechanism changes behavior, but they still need ablations to identify which component is responsible (Chen et al., 2026; He et al., 2025; Lv et al., 2026; Wang et al., 2026a).

Interpretability. Moving reasoning out of text reduces visible explanations. Evaluation should therefore include tools for auditing latent states: nearest-token decoding, auxiliary decoders, transition visualization, latent-language frequency analysis, diffusion block decoding, semantic-alignment probes, or proof-style checks on formal tasks. Interpretability should be treated as a first-class metric rather than an optional qualitative appendix, especially for methods that transfer an unseen plan from a large model to a smaller executor or optimize hidden states directly (Chen et al., 2026; He et al., 2025; Wang et al., 2026b).

How to read cross-paper comparisons. A method can dominate on one axis while losing on another. For example, a switch-based method may improve token efficiency under a strict budget but require extra controller logic; a semantic-alignment method may preserve rationales better but need a learned evaluator; and a diffusion or flow method may improve best-of-k diversity while increasing latent sampling cost (He et al., 2025; Shi et al., 2026; Kang et al., 2026; Zou et al., 2026). We therefore treat "better" as a Pareto statement over answer quality, visible tokens, latent updates, wall-clock cost, semantic fidelity, and interpretability.

15 Discussion

15.1 The field has moved from hidden evidence to designed latent computation

The multi-level taxonomy reveals an arc from *understanding* to *building* to *formalizing*. Early implicit-reasoning papers mostly asked what could be decoded from hidden states. Newer work asks how to design the hidden computation itself. Once a method introduces abstract tokens, transition heads, reasoning networks, semantic-alignment losses, latent guidance vectors, diffusion blocks, GFlowNet samplers, recurrent depth, or switch controllers, the latent state becomes an engineered interface rather than only an object of interpretation

(Hao et al., 2025; He et al., 2025; Shi et al., 2026). Formal work then explains why some of these design choices change the computational role of the intermediate state (Zhu et al., 2025; Xu & Sato, 2026).

15.2 Structure is the difference between useful and unstable latent thought

Successful latent systems almost always add structure. COCONUT uses a curriculum; CODI uses self-distillation; Token Assorted uses VQ-VAE codes and random mixing; SIM-CoT uses step-level supervision; GLR constrains motion to embedding-space transitions; Abstract-CoT uses constrained decoding and policy-iteration warm-up; SemCoT uses a semantic-alignment evaluator and lightweight generator; LRT uses a reasoning network; Latent Guidance uses large-to-small cognitive distillation and reconstruction; LaDiR uses VAE latents and diffusion refinement; LTF uses a reward-induced posterior and reference prior; DyLaR uses semantic residual refinement; and gradient-based representation optimization uses likelihood-prior regularization (Hao et al., 2025; He et al., 2025; Wang et al., 2026b). These choices are not implementation details. They are what make latent thought trainable, controllable, or deployable.

15.3 The medium determines the failure mode

Textual CoT fails by being long, unfaithful, or verbose. Continuous latent thought can drift. Discrete latent tokens can collapse into low-diversity code usage. Recurrent depth can overthink. Pure latent reasoning can maintain too many implicit paths and diffuse probability mass. Diffusion can spend too many denoising steps or produce latent trajectories that are diverse but not useful. Flow-based samplers can misallocate probability mass if rewards are sparse or poorly calibrated. Representation steering can harm language quality if it moves hidden states outside the model's natural manifold (Wei et al., 2025; Shi et al., 2026; Wang et al., 2026b). A taxonomy based on latent-state role helps because it predicts the kinds of failures one should test.

15.4 Hybrid reasoning is more plausible than replacement

The strongest empirical and theoretical evidence points toward hybrid reasoning. Text provides a readable memory and training signal. Latent states provide compactness, representation-level parallelism, semantic density, and non-verbal abstraction. Recurrence and diffusion provide iterative refinement. Learned codebooks provide controllable low-cost intermediate channels. Large-to-small latent guidance shows that planning and execution can be separated across models. Switching methods show that explicit and latent modes can play complementary roles within one inference trajectory (Wei et al., 2022; Hao et al., 2025; Shi et al., 2026). A practical reasoner may begin with a latent plan, verbalize only crucial steps, use hidden uncertainty to decide whether more compute is needed, and then stop when confidence is high.

15.5 Implicit reasoning changes the meaning of explanation

If a model reasons through latent vectors, guidance vectors, optimized hidden states, or abstract tokens, a displayed explanation may be a post-hoc translation rather than the actual computation. This does not make explanations useless, but it changes their status. Future systems should distinguish between the latent reasoning trace used by the model, a decoded approximation for humans, a semantically aligned replacement for explicit reasoning, and a faithful causal explanation of why the answer was produced (He et al., 2025; Chen et al., 2026; Wang et al., 2026b).

16 Research Gaps

Causal status of decoded information. Many diagnostic results show that reasoning-relevant variables are decodable from hidden states. The unresolved question is whether those variables are causally used by the model. A stronger evidential standard would require interventions: changing a decoded bridge entity should predictably change multi-hop answers, suppressing a latent search frontier should affect only dependent paths, and forcing a controller away from its preferred mode should change error or cost in a measurable way (Yang et al., 2025; Zhang et al., 2025).

Unstandardized semantic fidelity. Compression, guidance, and implicit-token methods often preserve final answer accuracy while hiding part of the reasoning trace. The field still lacks a shared notion of whether the hidden or implicit trace preserves the meaning of the explicit reasoning it replaces. Semantic-alignment losses, reconstruction objectives, and auxiliary decoders are promising, but they are not yet a common evaluation standard (He et al., 2025; Chen et al., 2026).

Inconsistent cost accounting. Efficiency claims are difficult to compare because generated tokens, latent steps, recurrent iterations, forward passes, teacher calls, controller overhead, and wall-clock latency are not reported uniformly. A method can reduce visible tokens while increasing hidden computation; another can improve a small executor by invoking a larger planner. These are different deployment regimes and should not be collapsed into a single "shorter reasoning" metric.

Stability of latent computation. Latent systems introduce failure modes that are not visible in ordinary CoT: drift away from the embedding manifold, implicit-token homogenization, overthinking under recurrence, noisy probability mass in pure latent reasoning, diffusion or flow miscalibration, and text-quality degradation under representation steering (Wei et al., 2025; Shi et al., 2026; Wang et al., 2026b). These failures are often evaluated within individual papers, but cross-method stress tests remain limited.

Interpretability and auditability. Moving reasoning out of natural language reduces observability. A decoded explanation may be a translation of a latent trace rather than the computation actually used. The field lacks reliable tools for auditing abstract tokens, latent guidance vectors, optimized hidden states, and distributions over latent trajectories.

Theory of learnability and control. Existing theory clarifies some expressivity differences between textual and latent thought, but it says less about when gradient-based training learns the useful latent algorithm, why some curricula stabilize latent trajectories, or how controllers can steer hidden states without degrading the language prior (Zhu et al., 2025; Xu & Sato, 2026).

17 Open Problems and Future Directions

Intervention-based evaluation. Future diagnostic work should pair probes with causal edits, ablations, and controlled swaps. Such tests would clarify when an internal variable is merely decodable and when it participates in the reasoning computation. The same principle should be applied to constructive methods through latent-token ablations, switch-budget interventions, residual-update ablations, and trajectory perturbations.

Unified benchmark and reporting protocols. A useful benchmark suite should jointly report answer accuracy, generated tokens, latent steps, recurrence depth, switch counts, teacher or controller calls, wall-clock cost, calibration, semantic fidelity, and robustness under compression. The goal is not one universal leaderboard, but a reporting protocol that makes Pareto comparisons meaningful across deployment settings.

Adaptive hybrid controllers. The next practical challenge is routing. A model should decide whether to answer directly, verbalize CoT, use abstract tokens, request large-model guidance, run recurrent hidden steps, optimize a hidden state, sample latent trajectories, or stop. Latent-horizon pivots, success probes, self-verification signals, entropy trends, semantic residuals, and prior-regularized objectives are natural inputs for such controllers (Xu et al., 2026; Shi et al., 2026; Lv et al., 2026).

Faithful and inspectable latent languages. Discrete abstract tokens and quantized codebooks make latent reasoning more auditable than unconstrained continuous vectors, but their meanings remain opaque. Future work should study compositionality, reuse across tasks, degeneration of code usage, and methods for translating latent symbols into faithful human-readable explanations (Ramji et al., 2026; He et al., 2025).

Robust long-horizon latent computation. Long latent trajectories require mechanisms that preserve semantic diversity, operator information, manifold alignment, and controllability over many steps. Promising

directions include auxiliary decoders, contrastive latent objectives, residual refinement, geometric constraints, reference priors, adaptive halting, and explicit diversity regularization.

Theory beyond expressivity. Future theory should connect circuit-style expressivity with learnability, optimization, and deployment constraints. The central question is not only what latent thought can compute in principle, but also when training discovers such computation, when controllers can reliably access it, and how hidden-state manipulation can remain inside the model's useful language manifold.

Survey infrastructure. As the area grows, survey work should maintain living taxonomies, benchmark-compatibility maps, and cost-accounting conventions. The comparative structure developed here is a first step toward organizing the field by methodological role and evaluation regime rather than by chronology alone.

18 Conclusion

This survey argues that reasoning in language models is distributed across more media than visible text. Hidden states can encode intermediate entities, answer tendencies, correctness signals, uncertainty, and planning cues. Trajectories reveal how these signals evolve. Constructive systems compress, align, guide, steer, switch, or control reasoning through latent mechanisms. Formal work clarifies why latent and textual thought have different computational strengths. The resulting picture is not that CoT is obsolete, but that CoT is one reasoning interface within a broader hybrid stack.

The survey contributes four forms of structure. First, it provides terminology that distinguishes hidden states, latent states, latent thoughts, latent tokens, latent trajectories, and controllers. Second, it proposes a multi-level taxonomy organized by the role of latent computation: understanding, building, and formalizing. Third, it presents a narrative stack in which diagnostics motivate trajectory analysis, trajectory analysis motivates constructive systems, and constructive systems motivate formal questions about representation and control. Fourth, it states comparison principles for benchmark families, cost metrics, semantic fidelity, controller stability, and formal task settings.

The main gaps follow directly from this synthesis. The field needs stronger causal validation of decoded latent information, standardized measures of semantic fidelity and latent drift, transparent cost accounting, interpretable latent languages, and controllers that decide when to use text, hidden computation, guidance, recurrence, sampling, or steering. Future work should therefore move beyond asking whether latent reasoning can replace CoT and instead ask how to compose an accurate, efficient, stable, and inspectable reasoning stack.

References

Anum Afzal, Florian Matthes, Gal Chechik, and Yftah Ziser. Knowing before saying: Llm representations encode information about chain-of-thought success before completion, 2025. URL https://arxiv.org/abs/2505.24362.

Hanzhu Chen, Lin Yang, Jie Wang, Junhao Yan, Zhe Wang, Xize Liang, and Jianye Hao. Latent-guided reasoning: Empowering small LLMs with large-model thinking. In *International Conference on Learning Representations*, 2026. URL https://iclr.cc/virtual/2026/poster/10007852.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach, 2025. URL https://arxiv.org/abs/2502.05171.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space, 2025. URL https://arxiv.org/abs/2412.06769.

Yinhan He, Wendy Zheng, Yaochen Zhu, Zaiyi Zheng, Lin Su, Sriram Vasudevan, Qi Guo, Liangjie Hong, and Jundong Li. SemCoT: Accelerating chain-of-thought reasoning through semantically-aligned implicit

tokens. In *Advances in Neural Information Processing Systems*, 2025. URL https://papers.neurips.cc/paper_files/paper/2025/hash/3ddbd473456a57e3cafb1ee51ddf8ff6-Abstract-Conference.html.

Cong Jiang, Xiaofeng Zhang, Fangzhi Zhu, Xiaowei Chen, Junxiong Zhu, and Zheng Zhang. Rethinking LLM reasoning: From explicit trajectories to latent representations. In *International Conference on Learning Representations*, 2026. URL https://github.com/MobiusDai/LRT.

Haoqiang Kang, Yizhe Zhang, Nikki Lijing Kuang, Nicklas Majamaki, Navdeep Jaitly, Yi-An Ma, and Lianhui Qin. LaDiR: Latent diffusion enhances LLMs for text reasoning. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2510.04573.

Harsh Kohli, Srinivasan Parthasarathy, Huan Sun, and Yuekun Yao. Loop, think, & generalize: Implicit reasoning in recurrent-depth transformers, 2026. URL https://arxiv.org/abs/2604.07822.

Deqian Kong, Minglu Zhao, Aoyang Qin, Bo Pang, Chenxin Tao, David Hartmann, Edouardo Honig, Dehong Xu, Amit Kumar, Matt Sarte, Chuan Li, Jianwen Xie, and Ying Nian Wu. Inference-time rethinking with latent thought vectors for math reasoning, 2026. URL https://arxiv.org/abs/2602.06584.

Shashi Kumar, Yacouba Kaloga, Petr Motlicek, Ina Kodrasi, and Andrea Cavallaro. Geometric latent reasoning induces shorter generations in llms, 2026. URL https://arxiv.org/abs/2606.02248.

Fangrui Lv, Lei Wang, Ruixin Hong, Yong Du, Xiangyu Wu, Tingting Gao, Guorui Zhou, and Changshui Zhang. Beyond tokens: Dynamic latent reasoning via semantic residual refinement. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026. URL https://ojs.aaai.org/index.php/AAAI/article/view/40513.

nostalgebraist. Interpreting gpt: The logit lens. LessWrong, 2020. URL https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens.

Keshav Ramji, Tahira Naseem, and Ramon Fernandez Astudillo. Thinking without words: Efficient latent reasoning with abstract chain-of-thought, 2026. URL https://arxiv.org/abs/2604.22709.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with latent thoughts: On the power of looped transformers. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2502.17416.

Yuval Shalev, Amir Feder, and Ariel Goldstein. Distributional reasoning in llms: Parallel reasoning processes in multi-hop reasoning, 2024. URL https://arxiv.org/abs/2406.13858.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. CODI: Compressing chain-of-thought into continuous space via self-distillation, 2025. URL https://arxiv.org/abs/2502.21074.

Dachuan Shi, Abedelkadir Asi, Keying Li, Xiangchi Yuan, Leyan Pan, Wenke Lee, and Wen Xiao. SwiReasoning: Switch-thinking in latent and explicit for pareto-superior reasoning LLMs. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2510.05069.

DiJia Su, Hanlin Zhu, Yingchen Xu, Jiantao Jiao, Yuandong Tian, and Qinqing Zheng. Token assorted: Mixing latent and text tokens for improved language model reasoning. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025. URL https://arxiv.org/abs/2502.03275.

Wenhui Tan, Jiaze Li, Jianzhong Ju, Zhenbo Luo, Ruihua Song, and Jian Luan. Think silently, think fast: Dynamic latent compression of llm reasoning chains. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2505.16552.

Xinyuan Wang, Dongjie Wang, Wangyang Ying, Haoyue Bai, Nanxu Gong, Sixun Dong, Kunpeng Liu, and Yanjie Fu. Efficient post-training refinement of latent reasoning in large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026a. URL https://ojs.aaai.org/index.php/AAAI/article/view/40659.

Zijian Wang, Yanxiang Ma, and Chang Xu. Eliciting chain-of-thought in base LLMs via gradient-based representation optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026b. URL https://ojs.aaai.org/index.php/AAAI/article/view/40669.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837, 2022.

Xilin Wei, Xiaoran Liu, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Jiaqi Wang, Xipeng Qiu, and Dahua Lin. SIM-CoT: Supervised implicit chain-of-thought, 2025. URL https://arxiv.org/abs/2509.20317.

Kevin Xu and Issei Sato. A formal comparison between chain of thought and latent thought. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2509.25239.

Liyan Xu, Mo Yu, Fandong Meng, and Jie Zhou. How far ahead do llms plan? uncovering the latent horizon in chain-of-thought reasoning. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.02103.

Sohee Yang, Elena Gribovskaya, Nora Kassner, Mor Geva, and Sebastian Riedel. Do large language models latently perform multi-hop reasoning?, 2025. URL https://arxiv.org/abs/2402.16837.

Eric Zelikman, Georges Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah D. Goodman. Quiet-star: Language models can teach themselves to think before speaking, 2024. URL https://arxiv.org/abs/2403.09629.

Anqi Zhang, Yulin Chen, Jane Pan, Chen Zhao, Aurojit Panda, Jinyang Li, and He He. Reasoning models know when they're right: Probing hidden states for self-verification, 2025. URL https://arxiv.org/abs/2504.05419.

Yufa Zhou, Yixiao Wang, Xunjian Yin, Shuyan Zhou, and Anru R. Zhang. The geometry of reasoning: Flowing logics in representation space. In *International Conference on Learning Representations*, 2026. URL https://github.com/MasterZhou1/Reasoning-Flow.

Hanlin Zhu, Shibo Hao, Zhiting Hu, Jiantao Jiao, Stuart Russell, and Yuandong Tian. Reasoning by superposition: A theoretical perspective on chain of continuous thought. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2505.12514.

Xiandong Zou, Jing Huang, Jianshu Li, and Pan Zhou. Latent thought flow: Efficient latent reasoning in large language models, 2026. URL https://arxiv.org/abs/2606.16222.