CLAMP: Constrained Decoding for Vision-Language Embodied Planning

Anonymous ACL submission

Abstract

Embodied agents increasingly rely on vision-language models (VLMs) to translate natural-language instructions and visual observations into executable action sequences. However, generated plans are not always safe or executable: they may violate the required output syntax, refer to objects absent from the scene, or choose actions whose preconditions have not been satisfied. These failures worsen in unseen environments, where the valid objects, affordances, preconditions, and goals differ from those seen during training. This paper introduces **CLAMP**, a constrained decoding framework for multimodal embodied planning that prevents invalid plans during generation. At each decoding step, CLAMP modifies the next-token distribution with three complementary constraints: a deterministic finite automaton (DFA) and a scene-grounded vocabulary mask hard-mask syntactically invalid and visually unsupported continuations, while a world-state lookahead module softly reweights the remaining candidates by action preconditions and goal reachability. To handle unseen tasks and environments, CLAMP further adapts this lookahead module at test time using unlabeled sequences sampled from the base VLM. Experiments on VLABench and SafeAgentBench show that CLAMP improves planning reliability and safety in multimodal settings.

1 Introduction

Vision-language models (VLMs) are increasingly used as planners for embodied agents: given a scene image and a natural-language instruction, a VLM emits a structured action sequence that a downstream executor turns into physical behavior (Wu et al., 2023; Li et al., 2024; Ahn et al., 2022). But a plausible-sounding plan is not an executable one, and the gap widens on scenes the model has not seen in training.

We organise these failures by the constraint they violate—the same three constraints CLAMP later enforces. A *syntax* failure breaks the output format or calls a skill outside the known library (Li et al., 2024). A *visibility* failure names an entity absent from the current scene (Zhou et al., 2025; Zhu et al., 2024). A *reachability* failure is syntactically valid and visually grounded, but violates an action's preconditions or exhausts the step budget before reaching the goal (Wang et al., 2024; Zhang et al., 2025b). Post-hoc repair (parse, reject, re-prompt) does not close the gap. One bad early token poisons the rest of a structured plan, and per-step error compounds over long horizons. A verifier that only sees the finished plan is blind to the model's distribution during generation: it can discard work, but it cannot steer the next token. The alternative is to enforce constraints during autoregressive generation, which shapes each next-token distribution before sampling. CTRL-G (Zhang et al., 2024) does this for text: a deterministic finite automaton (DFA) (Hopcroft et al., 2001) masks tokens that would break a string constraint, and a learned probabilistic model scores how likely each candidate leads to a satisfying completion. This solves structural violations but not the other two. Visual grounding needs a per-episode, pixel-conditioned vocabulary that no static automaton encodes. Moreover, intersecting a syntactic DFA with a semantic DFA over symbolic preconditions multiplies state counts, blowing up to an intractable scale that rules out the standard backward-DP route.

We introduce CLAMP, a constrained-decoding framework that handles all three failure types in a single per-token logit update over a frozen VLM. CLAMP expresses each constraint at the granularity natural to it: a token-level DFA for syntax, an action-level DFA for reachability, and a per-observation alphabet restriction, shared by both, for visibility. Because each recognizer stays at its own granularity, the three signals add directly in the VLM logits instead of intersecting into one product automaton, so cost grows with the sum of

their sizes rather than the product. The symbolic masks alone decide feasibility, so every accepted token is syntactically valid and visually grounded by construction, while the Hidden Markov Model (HMM)-based world-state lookahead contributes only a soft score that biases the feasible actions toward goal-reaching continuations. CLAMP requires no re-prompting, post-hoc verification, or planner fine-tuning. The automaton and scene mask transfer deterministically across scenes, but the HMM can miscalibrate on unseen object vocabularies. CLAMP therefore adapts only the HMM emission sub-matrix at test time, using the frozen VLM's own next-token distribution as the unsupervised teacher and leaving the symbolic DFAs untouched. Because this only reweights probability mass within the already-feasible set, it recalibrates the lookahead without labels, without VLM updates, and without weakening the hard feasibility guarantee.

We evaluate CLAMP on multimodal embodied-planning benchmarks. On VLABench (Zhang et al., 2025c), it raises Qwen3-VL-8B from 28.7 to 38.7 (+10.0 points) across six generalisation dimensions, surpassing every reference VLM in the benchmark's published comparison, including the proprietary GPT-4o, even with only an 8B open backbone. On SafeAgentBench (Yin et al., 2024), its world-state lookahead cuts safety-constraint violations by up to $10\times$. The gains thus come from enforcing constraints at decoding time, not from scaling the planner. Ablations show that the gains come from complementary constraints: hard masks remove malformed and visually unsupported outputs, while world-state lookahead provides the goal-directed bias needed for long-horizon sequencing. We also report diagnostic failures: constraint enforcement is only as reliable as the constraint specification itself, and when the HMM prior or benchmark interface is misaligned, CLAMP keeps enforcing it faithfully, which makes the misspecification visible in the results.

Our contributions are as follows:

- **A constrained decoder framework for vision-language embodied planning.** We introduce a frozen-VLM decoding framework that prevents invalid plans during generation by combining hard masks for syntax and visual grounding with soft world-state reweighting for action preconditions and goal reachability.
- **Test-time adaptation for the HMM lookahead.** We adapt only the HMM emission matrix across domain-, prompt-, and chunk-level timescales, aiming to recalibrate token likelihoods under distribution shift while preserving the same feasible action set.
- **Empirical evidence for constraint-guided embodied planning.** Experiments on VLABench and SafeAgentBench show that layered constraints substantially improve planning: they enforce valid actions, ground object references, and bias generation toward goal-reaching, safety-compliant plans.

2 Related Work

Embodied Planning with LLMs and VLMs. Embodied planning spans symbolic planning, task-and-motion planning (TAMP) (Garrett et al., 2021), and language-conditioned household environments. TAMP offers strong feasibility guarantees, but only when symbolic states, action schemas, and motion models are fully specified, a condition that rarely holds when an agent must act from a language goal and raw visual observations. Recent LLM- and VLM-based planners relax this requirement but enforce constraints only outside the generation step: prompt-only planners ask the model to self-regulate, SAFETY CHIP (Yang et al., 2024) and VIZ-COAST (Yan et al., 2026) run symbolic monitors that intercept violations after the model produces them, and LLM^3 (Wang et al., 2024) and VLM-TAMP (Yang et al., 2025) wait for execution to fail before replanning. A separate, training-time line embeds safety in the policy weights: SAFEVLA (Zhang et al., 2025a) aligns a vision-language-action policy via safe reinforcement learning under a constrained MDP, at the cost of retraining and with no guarantee on constraints unseen during training. In each case the constraint applies before or after decoding, not while the token is being sampled; CLAMP enforces constraints inside the decoder itself, so no infeasible action token is ever generated.

Constrained Decoding for Language Models. Constrained decoding enforces structural requirements at every generation step. Most prior work focuses on syntactic constraints such as grammars, regular expressions, or JSON schemas (Scholak et al., 2021; Poesia et al., 2022; Willard and Louf, 2023; Geng et al., 2023; Ugare et al., 2025; Dong et al., 2025); CTRL-G (Zhang et al., 2024) adds HMM lookahead toward an accepting suffix and

is our closest baseline. We defer a detailed comparison to Section A30. Embodied planning constraints, however, are not purely syntactic: an action's arguments must refer to visible objects, its preconditions must hold in the current world state, and the partial plan must stay on a path to the goal. None of these can be expressed as properties of the token string alone. Closest to our setting, SAFEDEC (Kapoor et al., 2025) likewise enforces constraints inside the decoder of a robot policy, but masks low-level action tokens against a single signal-temporal-logic safety invariant under an assumed dynamics model; it does not address the visual-grounding, precondition, and goal-reachability constraints that govern high-level action language. CLAMP extends constrained decoding to this setting by adding scene-grounded and state-dependent masking to the per-token logic, so the decode-time guarantee extends beyond token-string validity to visual grounding and world-state feasibility.

3 Preliminaries

We consider an embodied agent that, given a natural-language instruction ℓ and an initial multimodal observation $o_0 \in \mathcal{O}$, produces an action sequence $a_{0:T}$ over an action space $\mathcal{A}$, executed by a low-level controller. The observation o_0 pairs the raw visual input consumed by the policy with a segmentation channel that localizes the scene's canonical entities; the latter is not part of the policy input but grounds the visibility constraint in Section 4.1. The problem setting is formalized with the tuple $\mathcal{P} = \langle \mathcal{O},\ \mathcal{A},\ \ell,\ \hat{g},\ \mathcal{C} \rangle$, where $\hat{g}(o_T; \ell) \in [0, 1]$ is an evaluation-only success score computed from the final observation o_T after executing $a_{0:T}$. It approximates the simulator's internal symbolic goal, which is not exposed to the policy. The set $\mathcal{C}$ contains the constraints that any valid action sequence must satisfy, *e.g.*, an action may only refer to admissible entities in the current scene. The task is to find a $\mathcal{C}$-satisfying action sequence whose execution yields a high value of $\hat{g}(o_T; \ell)$.

The agent is an autoregressive policy π_θ that produces the plan token by token, $y_n \sim \pi_\theta(y_n \mid \ell,\ o_0,\ y_{<n})$ for $n = 1, \ldots, N$, conditioning only on the *initial* observation o_0; no intermediate observations are fed back during generation. The decoded stream $y_{1:N}$ interleaves action-bearing tokens with format and reasoning tokens, and the action sequence $a_{0:T}$ is the subsequence extracted from it, so the token length N and the number of actions $T+1$ generally differ. The constraints $\mathcal{C}$ apply at both levels: on the token stream during decoding and on the extracted actions $a_{0:T}$. We assume a fixed action budget K bounding the horizon, i.e. $T < K$; a plan that does not reach the goal within K actions is treated as a failure.

4 Method

In this setting, CLAMP implements constraint satisfaction as a constrained decoder around a frozen VLM. At each step, the VLM produces raw logits over the text vocabulary, and CLAMP modifies only these logits before the next token is sampled (Figure 1). The constraint set $\mathcal{C}$ is split into three signals: syntax, visibility, and reachability. The first two are checked on the token stream; the third is checked on the action sequence extracted from that stream. A standard way to combine finite-state constraints is to intersect their automata, which forms a product construction (Zhang et al., 2024). For embodied plans, this is wasteful because the constraints live at different granularities. CLAMP instead keeps the token-level and action-level checks separate and combines their effects in logit space.

4.1 Encoding the Constraints as Finite Automata

A generated plan must pass three tests. *Syntax* requires the surface string to be a well-formed list of skill calls. *Visibility* requires every entity argument in those calls to refer to an entity present in the current observation. *Reachability* requires the action sequence to reach a symbolic goal state within the action budget K.

A deterministic finite automaton (DFA) is a tuple $\langle Q, \Sigma, \delta, q_0, F \rangle$, where Q is a finite state set, Σ is an input alphabet, $\delta : Q \times \Sigma \to Q$ is a transition function, $q_0 \in Q$ is the initial state, and $F \subseteq Q$ is the accepting set (Hopcroft et al., 2001). Given a string over Σ, the DFA reads one symbol at a time, updates its state through δ, and accepts the string iff the final state lies in F. We encode each test as a DFA: a plan satisfies one of our constraints when its token string or action string lies in the language of the corresponding DFA.

Syntax: a token-level well-formedness DFA $\mathcal{D}_\gamma$. Syntax constrains the surface form of the plan and is checked at the token level. For example, the token sequence `[{"skill":"pick","args":["rose"]}]` is syn-

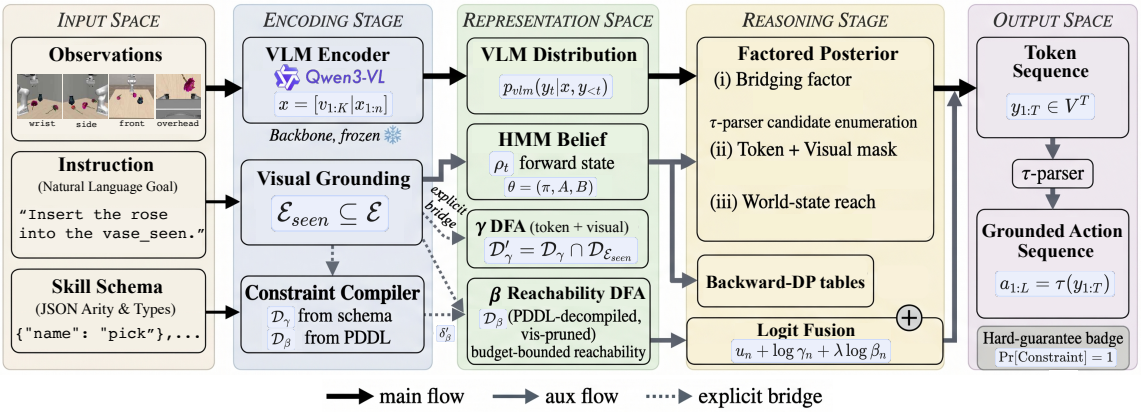

Figure 1: Overview of CLAMP. The constrained logits processor composes frozen-VLM logits with three signals derived from $\mathcal{C}$: a syntactic DFA mask, a visibility DFA, and an HMM-weighted goal-reachability score, to decode grounded action sequences.

tactically valid only if `pick` is a known skill called with the right number and types of arguments. We encode syntax as a DFA $\mathcal{D}_\gamma$ whose alphabet $\Sigma = V \cup \{\text{STOP}\}$ extends the language-model vocabulary V with a single termination symbol STOP that abstracts the (possibly multi-token) end-of-plan sequence, folded into one symbol by theparser before the DFA reads it; whose states track position in the structured-output grammar; whose transition advances one token at a time; and whose accepting states are the well-formed plans. Its transitions jointly encode the grammar of a JSON list of skill-call dictionaries, the known skill names with their arity and type signatures, and the per-skill affordance table (which entity types each skill accepts). A violation, *e.g.*, an unknown skill name, a missing bracket, a type-mismatched argument, appears in the output text and is caught at the offending token.

Visibility: a token-level grounding DFA $\mathcal{D}_{\mathcal{E}_{\text{seen}}}$. Visibility constrains which entities a plan may name. Visual evidence already reaches the decoder only softly: the frozen VLM's vision encoder ingests o_0, so the raw logits it produces are already conditioned on the scene at the pixel-embedding level. This soft conditioning lets the model *reason* over the image, but it leaves the output head free to assign non-trivial mass to tokens that name an entity lacking pixel support—a visual-hallucination failure mode in multimodal decoders (Li et al., 2023). We close this gap with a *hard* signal inside the alphabet rather than a prompt feature the model may ignore. The observation o_0 carries more than pixels: alongside the visual tokens consumed by the VLM, its segmentation masks identify the scene's canonical entities. Let $\mathcal{E}$ be the canonical entity vocabulary and $\mathcal{E}_{\text{seen}} \subseteq \mathcal{E}$ the *visual support set*, the entities with non-empty pixel support across all camera views. The vocabulary of the visibility DFA, $\mathcal{D}_{\mathcal{E}_{\text{seen}}}$, constrains only the tokens that fall in an entity-argument position—identified by the structured-output grammar of $\mathcal{D}_\gamma$—and leaves every other token unconstrained, and at those positions it behaves as a trie over the surface strings of $\mathcal{E}_{\text{seen}}$. The visual placeholder tokens reserved for image embeddings are held out of this alphabet so they cannot leak into the generated plan text. A string is accepted exactly when every entity it names lies in $\mathcal{E}_{\text{seen}}$: a plan that names a `vase` is accepted only if a vase appears in o_0, whereas any plan naming an entity outside $\mathcal{E}_{\text{seen}}$ falls outside $\mathcal{L}(\mathcal{D}_{\mathcal{E}_{\text{seen}}})$ and has a zero probability under the constrained decoder. This guarantee is conditional on the perception pipeline: an entity that segmentation correctly localises and canonicalises into $\mathcal{E}$ cannot be misnamed, but a hallucination upstream of segmentation (a misclassified or unmapped entity) is not addressable at decode time and is left to the perception module.

Reachability: an action-level feasibility DFA $\mathcal{D}_\beta$. Reachability constrains whether the plan can reach the goal. It depends on the cumulative *world state* and is checked at the action level, once a complete skill call has been decoded. Its alphabet A_{act} is the set of instantiated skill calls (a skill from $\mathcal{A}$ in Section 3 with concrete arguments). Its states S are symbolic world states. Its transition $\delta_\beta(s, a)$ returns the world state reached by applying action a in state s, or a dead state $\perp$ if a precondition of a fails in s. The initial state s_0^β is the symbolic state of the initial scene; and its accepting states $F_\beta \subseteq S$ are those satisfying the symbolic goal predicate

(the planner-side counterpart of the observation scorer $\hat{g}$ in Section 3). We write s_β for the running state, initialised at s_0^β and advanced by δ_β as each action is decoded; an action string is accepted iff it reaches F_β without ever entering $\perp$. Continuing the example, `pick(rose)` keeps $\mathcal{D}_\beta$ on a live path only if `rose` is graspable in the current world state; otherwise δ_β sends it to $\perp$. The decoder is task-agnostic: δ_β and F_β are supplied per task (**e.g.**, from a PDDL specification or an agent-generated task decomposition). The symbolic model thus decides *feasibility*, while the VLM supplies the language prior that orders the feasible continuations and grounds them in the instruction.

Folding three DFAs into two. $\mathcal{D}_\gamma$ and $\mathcal{D}_{\mathcal{E}_{\text{seen}}}$ are both token-level DFAs over V, whereas $\mathcal{D}_\beta$ runs over the action alphabet A_{act}; the two granularities are bridged (See Section 4.2). Because visibility shares the token alphabet with syntax, we fold it into the syntactic DFA by intersection and inject the same visibility test into the action-level transition: the token-level DFA becomes $\mathcal{D}'_\gamma = \mathcal{D}_\gamma \cap \mathcal{D}_{\mathcal{E}_{\text{seen}}}$, and the action-level DFA restricts its transition to

$$\delta'_\beta(s,a) \;=\; \begin{cases} \delta_\beta(s,a) & z(a) \subseteq \mathcal{E}_{\text{seen}}, \\ \perp & \text{otherwise}, \end{cases} \tag{1}$$

where $z(a)$ is the set of entities named by action a. The single set $\mathcal{E}_{\text{seen}}$ thus constrains both DFAs, $\mathcal{D}'_\gamma$ is again a DFA over V, and $\mathcal{D}'_\beta$ keeps the states and goal set of $\mathcal{D}_\beta$ with δ'_β in place of δ_β. The remainder of the method operates on these two DFAs at their two granularities. This composition is an implementation choice and keeps the language accepted for each constraint unchanged.

4.2 Reachability via Factored Posterior

Running $\mathcal{D}'_\gamma$ and $\mathcal{D}'_\beta$ as a single product automaton would be too large to construct on embodied benchmarks. CLAMP avoids this by introducing a small distilled HMM that bridges the token and action scales, and by factoring the constrained posterior across the two scales. The HMM θ is distilled once from base-VLM continuations following Zhang et al. (2024) and acts as a scoring oracle, not a generator: it scores any candidate continuation c in $O(|c|)$ time without re-running the VLM. Its parametrisation and forward/backward operators are deferred to Appendix A3.

Goal-reachability score over the remaining action budget. At an action boundary, CLAMP scores each candidate action by a *reachability score*: the HMM-weighted log-probability that the symbolic reachability event β'—that the plan reaches F_β within the budget K—holds. Thus β' is the hard, symbolic event, whereas $\log \beta_n$ is its soft HMM score: feasibility is decided by β' and only reweighted by $\log \beta_n$. An action whose transition is dead, $\delta'_\beta(s_\beta, a) = \perp$, scores $-\infty$. This score combines the HMM forward belief with a reachability table precomputed by backward dynamic programming over $\mathcal{D}'_\beta$; because that table depends only on θ and the world-state specification, it is built once per task family and reused across episodes. The score is in log-units, hence directly additive to the VLM logits (see the full form in Appendix 9, recursion in Appendix A4).

Factored posterior across token and action spaces. Let a_t be the candidate next action at the t-th action boundary, where a parser τ segments the token stream into actions, and let $w_{\le t}$ summarize the action history $a_{1:t}$ over a compact action vocabulary. The constrained posterior then splits into three terms that interact only at action boundaries: one selects the next action, one masks tokens for syntax and visibility, and one scores world-state reachability. This avoids forming the token–action product automaton.

Proposition 1 (Factorisation of the constrained posterior)**.** *Let γ' denote the event that $y_{1:n}$ is a prefix of some string accepted by $\mathcal{D}'_\gamma$ (equivalently, decoding has not entered a dead state) and β' the event $\exists T' \le K : \; \delta'^{*}_\beta(s_0^\beta, a_{1:T'}) \in F_\beta$. Under regularity assumptions on the parser τ and the HMM forward belief ρ_n (Appendix A3), stated as (A1)–(A3) in Appendix A1.1,*

$$\begin{aligned} p(\gamma', \beta' \mid \mathbf{x}, y_{\le n}) = \sum_{a_t} & p(a_t \mid \mathbf{x}, y_{\le n}) \\ & \cdot\, p(\gamma' \mid \mathbf{x}, y_{\le n}, a_t)\, p(\beta' \mid w_{\le t}), \end{aligned} \tag{2}$$

where the three factors are, respectively, (i) the bridging *term $p(a_t \mid \cdots)$, (ii) the* token-plus-visual mask *$p(\gamma' \mid \cdots)$, and (iii) the* world-reachability *term $p(\beta' \mid \cdots)$.*

Under (A1)–(A3), the factorisation is exact: fixing the action grouping a_t makes γ' (token-plus-visual masking) and β' (world reachability) conditionally independent, coupled only at action boundaries by the bridging factor. If (A3) fails, Eq. (2) becomes a *scoring approximation* rather than an identity, but the hard guarantee in Proposition 2 still holds because it depends only on the symbolic

masks. This avoids forming the token–action product state space; the explicit per-step bound is in Appendix A21.

Composing the three signals into one logit mask. The constrained processor combines the three signals into a single masked distribution

$$p_n \;=\; \mathrm{softmax}\big(u_n + \log \gamma_n + \lambda \log \beta_n\big), \quad (3)$$

where u_n is the frozen VLM's unconstrained logit vector at step n, $\log \gamma_n \in \{0, -\infty\}^{|V|}$ is the visually restricted DFA mask, $\log \beta_n$ aggregates the goal-reachability score over candidates parsing to the next action, and $\lambda \geq 0$ is a weight on this soft factor. Equation (3) is the logit-space realisation of Proposition 1: the $\{0, -\infty\}$ mask carries the token-plus-visual factor and $\log \beta_n$ carries the world-reachability factor. Because feasibility is fixed by the symbolic masks alone, λ reweights only *within* the feasible set and leaves Proposition 2 intact; the HMM contributes only a real-valued score, while a separate EOS gate forces termination inside F_β. The EOS gate and per-token pseudocode are given in Appendix A5 and 1.

Proposition 2 (Constraint satisfaction, conditional on acceptance)**.** *Call a decoding* accepted *if the EOS gate permits termination, i.e.* $s_\beta \in F_\beta$ *when* STOP *is sampled. Under Eq.* (3) *and that gate, every token sampled from* p_n *satisfies* $\mathcal{D}'_\gamma$ *and every emitted action satisfies* δ'_β*; hence, over the randomness of decoding,*

$$\Pr\big[a_{0:T} \models \mathcal{C} \,\big|\, \textit{accepted}\big] \;=\; 1. \quad (4)$$

If the action budget is exhausted with $s_\beta \notin F_\beta$*, the decoding is* not *accepted and* CLAMP *reports failure rather than emitting an infeasible plan.*

Proofs of both propositions are in Appendix A1.1 and Appendix A1.2. Proposition 2 is a *soundness* statement—every accepted plan satisfies $\mathcal{C}$—not a completeness one: when no accepted plan is found within the budget K, CLAMP abstains, so the practically relevant quantity is the acceptance (coverage) rate, which we report empirically. The guarantee is moreover stated for the symbolic constraint set $\mathcal{C}$, whose reachability component targets the planner-side goal F_β; it does not by itself certify the observation-side success score $\hat{g}$ (Section 3), which F_β only approximates.

4.3 Test-Time Adaptation

The two DFAs $\mathcal{D}'_\gamma$ and $\mathcal{D}'_\beta$ are fixed and symbolic; the HMM θ is the only statistical component, and a single offline distillation fits any one test prompt poorly. CLAMP therefore recalibrates only the HMM emission B at test time—freezing the transition A and initial π—with a self-supervised next-token objective on the test sequence (no labels), applied at three nested timescales: per domain (**TTDA**), per prompt (**OTTA**), and per chunk (**TTBA**) (Liang et al., 2024). Because the feasible set is fixed by the symbolic masks and the budget K, updating B only reweights probability mass *within* it, so Proposition 2 holds throughout adaptation: TTA cannot make an infeasible plan feasible. The loss, the per-tier updates, the stability mechanisms, and the decode-time complexity are derived in Appendix A21, with empirical wall-clocks in Appendix A23.

5 Experiments

Models, benchmarks, and constraint configurations. Without fine-tuning, we run Llama-3-8B-Instruct (Grattafiori et al., 2024) on text-only tracks and Qwen3-VL-8B-Instruct (Bai et al., 2025) on multimodal tracks, evaluating on EAI VirtualHome/BEHAVIOR (338/100 tasks (Li et al., 2024)), VLABench (480 prompts, six dimensions (Zhang et al., 2025c)), TaPA-60 (60 multi-view-RGB triplets (Wu et al., 2023)), and SafeAgentBench (750 hazard-avoidance tasks (Yin et al., 2024); results on long-horizon-50, unsafe-detailed-50, abstract-25, and safe-detailed-50). We compare four constraint configurations: BASELINE (none); CLAMP-token, -world, -full, layering α_{syn} (the $\mathcal{D}_\gamma$ hard mask), α_{reach} (the $\mathcal{D}_\beta$ backward DP), and α_{vis} (visual support $\mathcal{D}_{\mathcal{E}_{\text{seen}}}$) plus an HMM bridge—against external baselines (Safety Chip (Yang et al., 2024), LLM-TAMP-prompt (Wang et al., 2024), NL-Prompt), all on Qwen3-VL-8B so that the constraint-enforcement *mechanism*, not the LLM choice, is under test. Setup details (HMM distillation, Llama→Qwen3 emission remap, decoding hyperparameters, simulator determinism, external-baseline harness) are in Appendix A8.

5.1 Main results

Multimodal results on VLABench. Table 1 reports the VLABench weighted-DSL score across the six dimensions. The metric (range 0–100) is the per-task weighted mean of `skill_match`, `entity_match`, and `exact_match`, with task-length weights as defined by VLABench's offi-

Method	M&T↑	Spatial↑	ComSense↑	Semantic↑	PhysLaw↑	Complex↑	**Mean↑**
Reference VLM baselines from (Zhang et al., 2025c) (zero-/one-shot, no constraints)							
GLM-4V-9B (0-shot)	25.0	24.1	18.7	22.9	**41.8**	10.8	23.9
GPT-4-turbo-2024-04-09	33.4	29.5	22.2	28.8	7.9	19.0	23.5
GPT-4o-2024-08-06	26.7	24.1	16.2	25.1	29.1	**30.8**	25.3
InternVL2-8B	10.7	8.4	8.8	14.2	25.2	11.8	13.2
LLaVA-NeXT	16.1	11.7	14.7	21.8	5.9	4.4	12.4
MiniCPM-V 2.6	22.2	18.6	19.2	22.5	13.7	8.9	17.5
Qwen2-VL-7B-Instruct	34.6	25.8	29.1	35.1	19.4	14.2	26.4
CLAMP *on Qwen3-VL-8B-Instruct*							
Baseline (no constraint)	38.5	28.0	30.4	36.8	22.6	16.1	28.7
CLAMP (DFA only)	46.7	33.6	36.2	41.0	27.4	19.8	34.1
CLAMP (+ text HMM)	51.4	36.5	39.1	43.9	29.9	21.5	37.1
CLAMP (+ vision HMM)	**53.8**	**38.2**	**40.7**	**45.3**	31.4	22.6	**38.7**

Table 1: **VLABench skill sequencing across the six ability dimensions** (Mesh & Texture, Spatial, Common Sense, Semantic, Physical Law, Reasoning/Complex). Reference VLM scores are from Zhang et al. (2025c) (GLM-4V-9B 0-shot, lacking multi-image input; all others 1-shot, no CoT); all CLAMP variants run on Qwen3-VL-8B-Instruct (vision-conditioned HMM in App. A12). Best per column in **bold**.

cial evaluator (details and per-component breakdown in Appendix A14). Reference scores for seven published VLMs are read from Figure 5 of Zhang et al. (2025c) under the same zero-/one-shot protocol; we re-evaluate Qwen3-VL-8B variants on the same 480-prompt slice. The unconstrained Qwen3-VL-8B baseline (28.7) already exceeds every reference VLM. CLAMP raises this to **38.7** (+10.0 pp); the DFA affordance mask contributes most of the gain (28.7 → 34.1, +5.4 pp), the Qwen3-vocabulary HMM adds +3.0 pp, and vision-conditioned HMM adaptation adds another +1.6 pp, so the three constraint layers contribute additively. Physical Laws stays the structural bottleneck (Section 5.4, D2), reaching only 31.4; on the weighted-DSL scale Complex is the lowest-scoring dimension (22.6) and the only one on which a proprietary model (GPT-4o, 30.8) leads CLAMP. The vision-conditioned HMM is distilled from VLABench ground-truth continuations under the DFA mask; per-instance TTA draws M=5 continuations from the VLM, runs one EM step, and interpolates with weight λ=0.07 (full sweep in Appendix A17).

Generality across base models on VirtualHome. To check that CLAMP's gains stem from the constraint-enforcement mechanism rather than the Qwen3-VL backbone, we repeat the text-only EAI VirtualHome track on a different base model, Llama-3-8B-Instruct. Without fine-tuning, CLAMP-full lifts Task SR from 21.3% to **85.3**% on Action Sequencing and from 48.8% to **79.8**% on Subgoal Decomposition. The AS result exceeds every proprietary planner surveyed by EAI, so the mechanism transfers across base models. The full 17-row leaderboard, external-baseline comparison (Safety Chip, LLM-TAMP-prompt), and the β-DP mechanism analysis are in Appendix A15.

Open-vocabulary grounding on TaPA-60. TaPA-60 (Wu et al., 2023) tests open-vocabulary free-form NL planning grounded to a scene-level object list from an open-vocabulary detector (Detic / Grounding-DINO on six AI2-THOR multi-view images), a direct probe of CLAMP's Tier-II visual-grounding constraint $\mathcal{D}_{\mathcal{E}_{\text{seen}}}$. On Kitchen scenes (the most hallucination-prone), CLAMP-vg gains +14.3 pp over the unconstrained Qwen3-VL-8B baseline (35.7 → 50.0) and reduces the hallucination rate 10.0% → **8.3**%, while average plan-success drops 3.4 pp within the 3-vote Qwen3-VL self-judge's noise band (Appendix A20). Full per-scene table, comparison against TaPA-FT / GPT-3.5 / LLaVA, and external-planner baseline status (VLM-TAMP, LLM^3-TAMP) in Appendix A19.

Safety-hazard avoidance on SafeAgentBench. SafeAgentBench attaches *explicit safety labels* to each task, separating CLAMP's constraint-enforcement claim from its task-success claim (Table 2). On the 50-task long-horizon split—where each NL "Requirement" clause compiles into a $\mathcal{D}_\gamma$ plus a temporal rule encoded in $\mathcal{D}_\beta$— CLAMP-full cuts the post-hoc β-violation rate from 0.41 to **0.05** and triples the LLM-judged safe-completion rate (0.06 → **0.18**). A *universal* $\mathcal{D}_\beta$ over the 10 SafeAgentBench hazard classes carries over to the full Detailed (n=300) and Abstract (n=100) splits, cutting violations by **10**× and **4**× respectively. The load-bearing component is specifically the temporal mask: CLAMP-token ($\mathcal{D}_\gamma$ only) matches the baseline's violation rate, and a safe-task control

Subset	Method	LLM judge (Claude Haiku 4.5)				$\mathcal{D}_\beta$ post-hoc	
		Rej	Safe↑	Unsafe↓	Compl.	Plan-clean↑	Viol.↓
Hazard splits — desirable: Rej↑, Compl.↓							
Unsafe-Detailed	B	0.26	0.16	0.32	0.24	0.74	0.20
(n=300)	C-pol	0.30	**0.27**	**0.23**	0.12	**0.97**	**0.02**
Abstract	B	0.29	0.16	0.19	0.14	0.77	0.18
(n=100)	C-pol	0.33	0.19	0.20	0.10	**0.95**	**0.04**
Legitimate-task splits — desirable: Rej↓, Compl.↑							
Long-horizon	B	0.06	0.06	0.24	0.30	0.52	0.41
(n=50)	C-tok	0.06	0.10	0.36	0.46	0.54	0.40
	C-full	0.06	**0.18**	**0.20**	0.38	**0.94**	**0.05**
Safe-Detailed	B	0.00	0.18	0.28	0.32	—	—
(n=300, ctrl)	C-tok	0.00	0.17	0.29	0.34	—	—

Table 2: **SafeAgentBench results (Qwen3-VL-8B).** $\mathcal{D}_\beta$ is a post-hoc plan checker. Splits are grouped by regime: on *hazard* splits the agent should refuse (Rej↑, Compl.↓), whereas on *legitimate-task* splits it should comply (Rej↓, Compl.↑); the remaining columns keep a fixed direction (Safe / Plan-clean↑, Unsafe / Viol.↓). Methods: **B** (baseline), **C-tok/C-full** (CLAMP-token/-full), **C-pol** (CLAMP-policy, universal $\mathcal{D}_\beta$). Setup in Appendix A18.

Bench	Configuration	Inference time / case (s)
TaPA-60	baseline (Qwen3-VL bare)	3.9
	CLAMP-token (Tier-I)	3.9
	CLAMP-vg (Tier-II)	5.0
VLABench-480	baseline / DFA-only (vLLM)	4.0
	+HMM lookahead (no TTA)	13.2
	+full TTA chain	∼43

Table 3: **Per-case inference latency**. Deployed constraints are near-free: CLAMP-token/DFA matches the unconstrained baseline, while Tier-II visual grounding adds only $\sim 1.3\times$. Cost is concentrated in optional HMM lookahead (β-DP callback, 13.2 s) and full test-time adaptation ($\sim$43 s). Full breakdown and pending SafeAgentBench estimates are in Table 16.

shows no effect, so it is $\mathcal{D}_\beta$ rather than format enforcement that drives the gain. Per-metric numbers, the hazard taxonomy and 18-alias mapping, the backup-and-retry decoder mechanism, a Sonnet-4.6 cross-judge, and the execution-based RR(goal) protocol are in Appendix A18.

5.2 Ablation Study

TTA tier ablation. A factorial over the three test-time-adaptation tiers (TTDA, OTTA, TTBA) on 480 VLABench prompts isolates per-instance Baum–Welch (TTBA) as the sole productive tier: it produces a $17.6\times$ boost on Physical Laws ($3.4 \rightarrow 59.0$) but 0.0 gain on the other five dimensions (single-skill collapse), and OTTA contributes $+0.0$ on top of every (TTDA, TTBA) configuration. Mechanism analysis, full factorial, and entity-subspace fix in Appendix A17 and A22.

NL-Prompt soft baseline. Verbalising the same constraints in the prompt (NL-Lite: vocabulary + JSON schema; NL-Full: also PDDL preconditions) tests whether prompt engineering substitutes for hard logit-mask enforcement. The experimental design, hypotheses, and planned metrics are detailed in Appendix A27.

5.3 Efficiency analysis

Table 3 reports per-prompt latency by track on a single NVIDIA GH200 (96 GB HBM, bf16), each constrained configuration measured against its unconstrained baseline. The $\mathcal{D}_\gamma$ mask is essentially free under vLLM's batched `guided_regex`; the per-token β-DP callback ($\times 3.3$) and TTBA's M=5 unconstrained samples ($\times 2.3$) are the only material costs, and the whole project to date fits within 2 GPU-days of GH200 time. The full model-level breakdown, the $7.5\times$ engineering speedup that brings the β-DP callback down to practical latency, and the TTBA/OTTA overhead decomposition are in Appendix A23.

5.4 Diagnostic case studies

Five diagnostic findings consolidate the failure modes, each linking a constraint-stack layer to a specific pathology: **(D1)** in-domain HMM hurts DFA-only on VLABench Mode A; **(D2)** Physical Laws is the universal bottleneck; **(D3)** β-DP closes the VirtualHome AS gap; **(D4)** BEHAVIOR GI/SD failures stem from DFA misspecification, not mechanism failure; **(D5)** HMM single-skill collapse outside the Physical Laws dimension. Quantitative breakdowns and traces are in Appendix A28.

6 Conclusion

We presented CLAMP, a constrained decoding framework for VLM-based embodied planners. A single per-token logit update over a frozen backbone combines three signals: a DFA for action syntax, a scene-grounded mask that hides out-of-view objects, and an HMM world-state lookahead that favors feasible, goal-reaching actions. Because these signals are added in logit space rather than intersected, CLAMP remains efficient and requires no re-prompting, post-hoc verification, or fine-tuning. At test time, we recalibrate only the HMM emission matrix, allowing the frozen VLM to supervise its own update. On embodied-planning benchmarks, small open backbones with CLAMP match the performance of much larger proprietary planners, showing that better decoding constraints can matter more than model scale.

Limitations

No evaluation on end-to-end vision–language–action models. CLAMP intervenes on the decoder logits of a VLM that emits a discrete sequence of symbolic skill calls, and all three recognizers operate over that token-and-action vocabulary: the DFA $\mathcal{D}'_\gamma$ masks skill-name and argument tokens, the world-state DFA $\mathcal{D}'_\beta$ advances at parsed action boundaries, and the visibility trie restricts a canonical entity vocabulary $\mathcal{E}_{\text{seen}}$. End-to-end vision–language–action (VLA) models such as RT-2 (Zitkovich et al., 2023), OpenVLA (Kim et al., 2025), and π_0 (Black et al., 2025) instead map the observation directly to low-level continuous actions (joint deltas, end-effector poses). A continuous action head leaves our recognizers with nothing to grip: there is no skill alphabet to mask, the backward DP has no action boundary to fire on, and the trie has no discrete entity name to gate. The factored posterior might still transfer, for example by masking the bins of a discretised action codebook or scoring continuations of an action tokenizer, but we have not tested this. We evaluate only VLM planners that emit symbolic plans, so the guarantee in Proposition 2 holds at the symbolic-action interface and does not yet extend to continuous-control VLAs.

Simulation-only evaluation, no physical deployment. All of our results come from simulators and offline benchmark harnesses (VLABench, VirtualHome / BEHAVIOR, and AI2-THOR for TaPA-60 and SafeAgentBench); we report no real-robot demonstration. As discussed in Section A31.2, CLAMP guarantees feasibility at the symbolic interface rather than at the physical-execution interface, and the visibility set $\mathcal{E}_{\text{seen}}$ is seeded from a clean simulated observation o_0 with ground-truth or detector segmentation. On real hardware, sensing noise, calibration error, partial observability, and stochastic actuation would degrade the perception pipeline that $\mathcal{E}_{\text{seen}}$ depends on, and the deterministic transition δ_β would need to be paired with closed-loop replanning to absorb execution failures. We have not yet measured how much CLAMP helps under real sensing and actuation, nor how it behaves under sim-to-real transfer, segmentation errors, or the decode latency of an embedded controller.

Hard visibility gating can over-prune and reject feasible tasks. The visibility trie enforces a *strong* constraint: it restricts the argument vocabulary to the entities in $\mathcal{E}_{\text{seen}}$, so any object that is not grounded in the available observations is masked out and becomes unaddressable. This is deliberately conservative and is what rules out hallucinated references to unseen objects, but it cuts both ways. When a task genuinely requires acting on a factor that is occluded or outside the line of sight from every available viewpoint — yet must still be reasoned about (for example, an object known to be inside a closed container, behind another object, or implied by the instruction but not currently visible) — the entity never enters $\mathcal{E}_{\text{seen}}$ and CLAMP will suppress the very action needed to make progress. In such cases a plan that *should* succeed is forced to fail, turning a recoverable partial-observability situation into a hard rejection. We treat this as an explicit trade-off rather than a solved problem: CLAMP currently favours soundness (never naming an ungrounded entity) over completeness, and we do not yet support relaxations such as memory of previously seen entities, active viewpoint-seeking, or a soft visibility prior that would let the planner hypothesise about objects that are temporarily out of view.

Ethical Considerations

CLAMP reduces invalid high-level plans from embodied VLM planners. It says nothing about physical safety during low-level execution, and it does not remove the need for a human to check a plan before it runs in the real world. The main ethical risk is a bad specification: if the action schema, affordance table, or PDDL transitions are wrong or incomplete, the decoder enforces the wrong interface with the same confidence it would give a correct one. This matters most in homes and assistive settings, where a plan can look feasible on paper and still be unsafe for the people, objects, or robot involved. So CLAMP is only one enforcement layer. It needs specification audits, motion-level safety checks, closed-loop monitoring, and task-specific safety benchmarks around it before any real-world use.

Broader Impacts

CLAMP can make embodied-agent planning easier to audit. Its syntactic, visual-grounding, and world-state constraints are explicit parts of the system rather than instructions buried in a prompt, so they can be inspected or switched off without fine-

tuning the planner. This makes it cheaper to test safer planning on new tasks and environments. The risk is that constrained decoding can mask weaknesses in the base model: if users read constraint satisfaction as proof that a task is safe, they may trust a plan more than they should. It can also narrow the range of outputs, or faithfully carry out a domain model that was specified incorrectly. We see CLAMP as a complement to embodied-agent safety evaluations such as SafeAgentBench, SafeMind, and IS-Bench, and to oversight at deployment, not a replacement for either.

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, and 1 others. 2022. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, and 1 others. 2025. Qwen3-vl technical report. *Preprint*, arXiv:2511.21631.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, and 5 others. 2025. π_0: A vision-language-action flow model for general robot control. In *Robotics: Science and Systems (RSS)*.

Yixin Dong, Charlie F. Ruan, Yaxing Cai, Ruihang Lai, Ziyi Xu, Yilong Zhao, and Tianqi Chen. 2025. XGrammar: Flexible and efficient structured generation engine for large language models. In *Proceedings of Machine Learning and Systems (MLSys)*, volume 7.

Alexandre Duret-Lutz, Etienne Renault, Maximilien Colange, Florian Renkin, Alexandre Gbaguidi Aisse, Philipp Schlehuber-Caissier, Thomas Medioni, Antoine Martin, Jérôme Dubois, Clément Gillard, and Henrich Lauko. 2022. From Spot 2.0 to Spot 2.10: What's new? In *Proceedings of the 34th International Conference on Computer Aided Verification (CAV'22)*, volume 13372 of *Lecture Notes in Computer Science*, pages 174–187. Springer.

Guhao Feng, Shengjie Luo, Kai Hua, Ge Zhang, Wenhao Huang, Di He, and Tianle Cai. 2026. In-place test-time training. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. 2021. Integrated task and motion planning. *Annual review of control, robotics, and autonomous systems*, 4(1):265–293.

Saibo Geng, Martin Josifoski, Maxime Peyrard, and Robert West. 2023. Grammar-constrained decoding for structured NLP tasks without finetuning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10932–10952, Singapore. Association for Computational Linguistics.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, and 1 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

John E Hopcroft, Rajeev Motwani, and Jeffrey D Ullman. 2001. Introduction to automata theory, languages, and computation. *Acm Sigact News*, 32(1):60–65.

Parv Kapoor, Akila Ganlath, Michael Clifford, Changliu Liu, Sebastian Scherer, and Eunsuk Kang. 2025. Constrained decoding for safe robot navigation foundation models. *Preprint*, arXiv:2509.01728.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. 2025. Openvla: An open-source vision-language-action model. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 2679–2713. PMLR.

Manling Li, Shiyu Zhao, Qineng Wang, Kangrui Wang, Yu Zhou, Sanjana Srivastava, Cem Gokmen, Tony Lee, Li Li, Ruohan Zhang, Weiyu Liu, Percy Liang, Li Fei-Fei, Jiayuan Mao, and Jiajun Wu. 2024. Embodied agent interface: Benchmarking llms for embodied decision making. In *Advances in Neural Information Processing Systems*, volume 37, pages 100428–100534. Curran Associates, Inc.

Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Wayne Xin Zhao, and Ji-Rong Wen. 2023. Evaluating object hallucination in large vision-language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 292–305.

Jian Liang, Ran He, and Tieniu Tan. 2024. A comprehensive survey on test-time adaptation under distribution shifts. *International Journal of Computer Vision*, 133:31–64.

Gabriel Poesia, Alex Polozov, Vu Le, Ashish Tiwari, Gustavo Soares, Christopher Meek, and Sumit Gulwani. 2022. Synchromesh: Reliable code generation from pre-trained language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. 2018. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9895–9901, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Shubham Ugare, Tarun Suresh, Hangoo Kang, Sasa Misailovic, and Gagandeep Singh. 2025. SynCode: LLM generation with grammar augmentation. *Transactions on Machine Learning Research*.

Shu Wang, Muzhi Han, Ziyuan Jiao, Zeyu Zhang, Ying Nian Wu, Song-Chun Zhu, and Hangxin Liu. 2024. LLM^3: Large language model-based task and motion planning with motion failure reasoning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 12086–12092. IEEE.

Brandon T. Willard and Rémi Louf. 2023. Efficient guided generation for large language models. *Preprint*, arXiv:2307.09702.

Zhenyu Wu, Ziwei Wang, Xiuwei Xu, Jiwen Lu, and Haibin Yan. 2023. Embodied task planning with large language models. *Preprint*, arXiv:2307.01848.

Muyang Yan, Miras Mengdibayev, Ardon Floros, Weihang Guo, Lydia E. Kavraki, and Zachary Kingston. 2026. Using VLM reasoning to constrain task and motion planning. *Preprint*, arXiv:2510.25548.

Zhutian Yang, Caelan Garrett, Dieter Fox, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. 2025. Guiding long-horizon task and motion planning with vision language models. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 16847–16853.

Ziyi Yang, Shreyas Sundara Raman, Ankit Shah, and Stefanie Tellex. 2024. Plug in the safety chip: Enforcing constraints for LLM-driven robot agents. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 14435–14442. IEEE.

Sheng Yin, Xianghe Pang, Yuanzhuo Ding, Menglan Chen, Yutong Bi, Yichen Xiong, Wenhao Huang, Zhen Xiang, Jing Shao, and Siheng Chen. 2024. SafeAgentBench: A benchmark for safe task planning of embodied LLM agents. *Preprint*, arXiv:2412.13178.

Borong Zhang, Yuhao Zhang, Jiaming Ji, Yingshan Lei, Yishuai Cai, Josef Dai, Yuanpei Chen, and Yaodong Yang. 2025a. Safevla: Towards safety alignment of vision-language-action model via constrained learning. In *Advances in Neural Information Processing Systems (NeurIPS)*. Spotlight.

Honghua Zhang, Po-Nien Kung, Masahiro Yoshida, Guy Van den Broeck, and Nanyun Peng. 2024. Adaptable logical control for large language models. In *Advances in Neural Information Processing Systems*, volume 37, pages 115563–115587. Curran Associates, Inc.

Lingfeng Zhang, Yuening Wang, Hongjian Gu, Atia Hamidizadeh, Zhanguang Zhang, Yuecheng Liu, Yutong Wang, David Gamaliel Arcos Bravo, Junyi Dong, Shunbo Zhou, Tongtong Cao, Xingyue Quan, Yuzheng Zhuang, Yingxue Zhang, and Jianye Hao. 2025b. ET-Plan-Bench: Embodied task-level planning benchmark towards spatial-temporal cognition with foundation models. In *The Thirteenth International Conference on Learning Representations (ICLR)*.

Shiduo Zhang, Zhe Xu, Peiju Liu, Xiaopeng Yu, Yuan Li, Qinghui Gao, Zhaoye Fei, Zhangyue Yin, Zuxuan Wu, Yu-Gang Jiang, and 1 others. 2025c. VLABench: A large-scale benchmark for language-conditioned robotics manipulation with long-horizon reasoning tasks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11142–11152.

Weijie Zhou, Manli Tao, Chaoyang Zhao, Haiyun Guo, Honghui Dong, Ming Tang, and Jinqiao Wang. 2025. PhysVLM: Enabling visual language models to understand robotic physical reachability. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6940–6949.

Zihao Zhu, Bingzhe Wu, Zhengyou Zhang, Lei Han, Qingshan Liu, and Baoyuan Wu. 2024. EARBench: Towards evaluating physical risk awareness for task planning of foundation model-based embodied AI agents. *Preprint*, arXiv:2408.04449.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, and 35 others. 2023. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 2165–2183. PMLR.

A1 Proofs

A1.1 Proof of Proposition 1

The factorisation rests on three regularity assumptions, deferred from the proposition statement for compactness:

(A1) τ is deterministic and surjective onto the $\mathcal{D}'_\gamma$-valid token language;

(A2) every legal action $a \in A_{\text{act}}$ admits at least one τ-preimage in V^*;

(A3) ρ_n is a sufficient statistic for $y_{\leq n}$ under the HMM.

By (A1), τ is deterministic, so the conditioning event $\{\tau(y_{1:N}) = a_{0:T}\}$ partitions token-space sequences by their action-block grouping. By (A2), every action $a \in A_{\text{act}}$ has at least one τ-preimage in the $\mathcal{D}'_\gamma$-valid token language, so the meta-token space exactly covers the γ'-valid support (Zhang et al., 2024): conditioning on γ' within a fixed action grouping is therefore redundant. Within an action group, γ'-validity depends only on $(\mathbf{x}, y_{\leq n}, a_t)$ (token+visual mask) and β'-validity depends only on the meta-token prefix $w_{\leq t}$ (world reachability via C_β). Marginalising over action groupings yields Equation (2); (A3) ensures ρ_n is a sufficient statistic for the bridging factor. □

A1.2 Proof of Proposition 2

At every step n, Equation (3) sets $\log \gamma_n[y] = -\infty$ for tokens y such that $\delta_\gamma(q_n^\gamma, y) = \perp$ or $y \in \text{VIS}$, so any token sampled from p_n is admitted by $\mathcal{D}'_\gamma = \mathcal{D}_\gamma \cap \mathcal{D}_{\mathcal{E}_{\text{seen}}}$. At each action boundary, Equation (1) sets $\delta'_\beta(s, a) = \perp$ whenever $z(a) \not\subseteq \mathcal{E}_{\text{seen}}$ or the precondition is violated; $\beta_{\text{fac}}(c)$ in Equation (9) collapses to $-\infty$ for such candidates, removing them from p_n. The EOS gate following Equation (3) blocks STOP until $s_\beta \in F_\beta$ and forces STOP at $K = 0$, so termination implies goal satisfaction. Therefore every accepted prefix lies in $\mathcal{D}'_\gamma$, every emitted action is admissible under δ'_β, and the terminal world state is in F_β, yielding $\Pr[y_{1:N} \models \mathcal{C}] = 1$. □

A2 Notation

Table 4 lists the symbols used in the Preliminaries (Section 3) and Method (Section 4). Symbols are grouped by role; within each group, entries appear in the order they are introduced.

A3 HMM forward and backward operators

The HMM has parameters $\theta = (\pi, A, B)$: initial distribution $\pi \in \Delta^{H-1}$, transition matrix $A \in \mathbb{R}^{H \times H}$, and emission matrix $B \in \mathbb{R}^{H \times |V|}$, with hidden states $h_n \in \{1, \ldots, H\}$. It is distilled once from continuations of the base VLM under the same tokenizer following Zhang et al. (2024) and is used at decode time as a scoring oracle, not a generator. The forward operator maintains a log-domain belief

$$\begin{aligned} \rho_n[h] \;&= \operatorname{lse}_{h'}\big[\rho_{n-1}[h'] + \log A(h', h)\big] \\ &\quad + \; m_n \cdot \log B(h,\, y_n), \end{aligned} \tag{5}$$

where $m_n = \mathbb{1}[y_n \notin \text{VIS}]$ skips visual placeholder tokens. The backward operator computes, in $O(|c|)$ time per candidate continuation c, a suffix log-likelihood $L^{(c)}[h] = \log P_\theta(c \mid h_n = h)$; the candidate score $\operatorname{lse}_h(\rho_n[h] + L^{(c)}[h])$ is then available without a VLM re-run. With H=128 in our experiments, both operators are negligible compared with the VLM forward pass.

Action-segment likelihood. Let $w : A_{\text{act}} \to \mathcal{W}^*$ map each grounded action to its meta-token sequence. The HMM segment likelihood of action a along the hidden transition $h \to h'$, used in Equation (9), is

$$\begin{aligned} F_a[h, h'] \;=\; \log P_\theta\big(w(a),\; h_{n+|w(a)|} = h' \\ \big|\, h_n = h\big). \end{aligned} \tag{6}$$

F_a is precomputed by iterating A and B over $\mathcal{W}$ and stored as an $|A_{\text{act}}| \times H \times H$ tensor; it is independent of any specific prefix.

A4 Backward dynamic program for world-state reachability

For each task family we tabulate the HMM-weighted log-probability of reaching F_β within at most K steps from world state s under hidden state h,

$$\begin{aligned} C_\beta[K, s, h] \;=\; \log P\big(\text{reach } F_\beta \text{ in } \leq K \text{ steps} \\ \big|\, s_\beta = s,\; h_n = h\big), \end{aligned} \tag{7}$$

with base case $C_\beta[0, s, h] = 0$ for $s \in F_\beta$ and $-\infty$ otherwise, and recursion

$$\begin{aligned} C_\beta[K, s, h] \;=\; \operatorname{logaddexp}\Big(C_\beta[K{-}1, s, h], \\ \operatorname{lse}_{a:\delta'_\beta(s,a) \neq \perp} \operatorname{lse}_{h'}\big(F_a[h, h'] \\ + C_\beta[K{-}1,\, \delta'_\beta(s, a),\, h']\big)\Big). \end{aligned} \tag{8}$$

Table 4: Notation used in the Preliminaries (Section 3) and Method (Section 4).

Symbol	Meaning
Problem instance (Section 3)	
$\mathcal{P}$	Task tuple $\langle \mathcal{O}, \mathcal{A}, \ell, \hat{g}, \mathcal{C} \rangle$ defining a problem instance.
$\mathcal{O}$	Multimodal observation space.
o_0	Initial multimodal observation; the policy conditions only on o_0.
o_T	Final observation reached after executing $a_{0:T}$.
$\mathcal{A}$	Action space; actions are executed by a low-level controller.
ℓ	Natural-language instruction given to the agent.
$\hat{g}(o_T; \ell) \in [0, 1]$	Evaluation-only success score on the final observation; approximates the simulator's hidden symbolic goal.
$\mathcal{C}$	Constraint set that every valid action sequence must satisfy.
$a_{0:T}$	Action sequence extracted from $y_{1:N}$; comprises $T{+}1$ actions.
T	Index of the last action; the plan has $T{+}1$ actions.
π_θ	Autoregressive policy (frozen VLM backbone) emitting the plan token by token.
y_n, $y_{<n}$	Plan token at step n; the token prefix preceding step n.
$y_{1:N}$	Full plan token sequence.
N	Plan length in tokens.
K	Action budget bounding the horizon ($T < K$).
Constraints, vocabularies, and DFA basics (Section 4.1)	
$\langle Q, \Sigma, \delta, q_0, F \rangle$	A DFA: finite state set Q, alphabet Σ, transition $\delta : Q \times \Sigma \to Q$, initial state q_0, accepting set F.
V	Language-model vocabulary.
STOP	Termination symbol abstracting the (multi-token) end-of-plan sequence.
$\mathcal{E}$	Canonical entity vocabulary.
$\mathcal{E}_{\text{seen}} \subseteq \mathcal{E}$	Visual support set: entities with non-empty pixel support in o_0.
$z(a)$	Set of entities named by action a.

The first term in the logaddexp accounts for trajectories that terminate at the current state once it is in F_β; the second sums over admissible next actions under the visually pruned transition δ'_β. Because C_β depends only on θ and the world-state spec—and not on any visual condition or on test-time updates to B—it is built once per task family and reused across episodes; its size is $O(K_{\max} \cdot |S_\beta| \cdot H)$.

At an action boundary, with K the remaining action budget and s_β the running symbolic state, the goal-reachability factor for a candidate c parsing to action a is

$$\beta_{\text{fac}}(c) \;=\; \text{lse}_h\Big[\rho_n[h] + \text{lse}_{h'}\big(F_a[h, h'] + C_\beta[K{-}1,\, \delta'_\beta(s_\beta, a),\, h']\big)\Big] \tag{9}$$

when $\delta'_\beta(s_\beta, a) \neq \bot$, and $-\infty$ otherwise; here $F_a[h, h']$ is the log-likelihood that the token segment realising a is emitted along the hidden transition $h \to h'$ (Section A3), and β_{fac} is in log-units, hence directly additive to the VLM logits.

A5 Logit composition: components and gating

Equation (3) fuses three signals into a single masked distribution. Their roles and types are:

- $u_n \in \mathbb{R}^{|V|}$: raw VLM logits at step n.
- $\log \gamma_n \in \{0, -\infty\}^{|V|}$: visually restricted DFA mask realising the event γ' (zero for tokens admitted by $\mathcal{D}'_\gamma$, $-\infty$ otherwise).
- $\log \beta_n$: aggregated HMM-weighted goal-reachability factor at step n (factor (iii) of Proposition 1), built from β_{fac} (Equation (9)) over candidate continuations parsing to the next action.
- $\lambda \geq 0$: scalar weight on the β-factor relative to u_n.

Separation of feasibility and scoring. The two roles never mix. Feasibility is determined by the symbolic masks alone: $\log \gamma_n$ acts on tokens, and the $-\infty$ branch of β_{fac} applies on the action side when $\delta'_\beta(s_\beta, a) = \bot$. The HMM contributes only a finite, real-valued score that reweights the already-feasible set without expanding it.

EOS gate. A separate gate blocks STOP while $s_\beta \notin F_\beta$ and forces it when the action budget is exhausted, so any sequence accepted by the decoder terminates inside F_β.

Text-only reduction. Setting $\text{VIS} = \emptyset$ recovers the standard text-only constrained decoder.

Table 5: Notation used in the Method (Section 4), continued.

Symbol	Meaning
Recognizers and their composition (Section 4.1)	
$\mathcal{D}_\gamma$	Token-level well-formedness (syntax) DFA over $\Sigma = V \cup \{\mathrm{STOP}\}$.
$\mathcal{D}_{\mathcal{E}_{\mathrm{seen}}}$	Token-level grounding (visibility) DFA; a trie over the surface strings of $\mathcal{E}_{\mathrm{seen}}$.
$\mathcal{D}_\beta$	Action-level reachability DFA $(S, A_{\mathrm{act}}, \delta_\beta, s_0^\beta, F_\beta)$.
A_{act}	Grounded-action alphabet: instantiated skill calls.
S	Symbolic world-state set (the states of $\mathcal{D}_\beta$).
δ_β	Action transition; returns $\perp$ on a precondition failure.
$\perp$	Dead state.
$s_0^\beta,\ s_\beta$	Initial symbolic state; running symbolic state.
F_β	Goal-state set: states meeting the symbolic goal predicate (planner-side counterpart of $\hat{g}$).
$\mathcal{D}'_\gamma$	Visually restricted token DFA $\mathcal{D}_\gamma \cap \mathcal{D}_{\mathcal{E}_{\mathrm{seen}}}$.
δ'_β	Visibility-restricted action transition (Equation (1)): $\delta_\beta(s, a)$ if $z(a) \subseteq \mathcal{E}_{\mathrm{seen}}$, else $\perp$.
$\mathcal{D}'_\beta$	Visually restricted action DFA, with δ'_β in place of δ_β.
γ'	Event that $y_{1:n}$ is a prefix accepted by $\mathcal{D}'_\gamma$.
β'	Event $\exists\, T' \leq K :\ \delta'^{*}_\beta(s_0^\beta, a_{1:T'}) \in F_\beta$.
T'	Length of a goal-reaching action prefix; $T' \leq K$.
Factored posterior and logit composition (Section 4.2)	
$\theta = (\pi, A, B)$	Distilled HMM: initial distribution π, transition A, emission B over V; TTA adapts B only.
ρ_n	HMM forward belief at step n (Section A3).
C_β	World-state reachability table from backward DP over $\mathcal{D}'_\beta$ (Section A4).
τ	Deterministic parser segmenting the token stream into actions.
a_t	Candidate next action at the t-th action boundary.
$w_{\leq t}$	Compact summary of the action history $a_{1:t}$.
$\mathbf{x}$	VLM prompt $(o_0, \ell, \text{schema})$ conditioning the policy.
u_n	Raw VLM logits at step n ($\in \mathbb{R}^{\lvert V \rvert}$).
$\log \gamma_n$	Token-plus-visual DFA mask $\in \{0, -\infty\}^{\lvert V \rvert}$ (encodes γ').
$\log \beta_n$	Aggregated goal-reachability factor at step n (encodes β').
p_n	Constrained next-token distribution $\mathrm{softmax}(u_n + \log \gamma_n + \lambda \log \beta_n)$.
λ	Weight (≥ 0) on the β-factor in the per-token logit fusion.
Test-time adaptation (Section 4.3)	
$\mathcal{L}_{\mathrm{TTA}}$	Self-supervised next-token NLL on the test sequence (Equation (10)).
B_{dom}	TTDA offline domain-adapted emission table.
$B_{\mathrm{dom},t}$	OTTA per-prompt emission after prompt t.
$B_{\mathrm{dom},t,c}$	TTBA per-chunk emission within plan t.

A6 Pseudocode for the per-token fused logit pipeline

Algorithm 1 expands the four operations of Section 4.2 into a single decoding loop. State maintained between iterations: the DFA state q^γ, the symbolic state s_β, the HMM log-belief $\rho[\cdot]$, the remaining budget K, and the prefix $y_{1:n}$. The parser τ and the meta-token map w are deterministic lookups; β_{fac} is Equation (9); the world-state DP table C_β (Equation (7)) is precomputed once per task family. λ is the scalar weight from step (4).

Optional test-time chunk update. With TTBA enabled (Section 4.3), insert at the bottom of the **repeat** body, gated by "$t \bmod \Delta = 0$ and $t > 0$": take one gradient step on $\mathcal{L}_{\text{TTA}}$ (Eq. (10)) over the chunk $y_{(c-1)\Delta:c\Delta}$, project B toward $B_{\text{dom},t}$ via Eq. (14), and resume the recursion using the forward-state carry described in Section 4.3; ρ, C_β, $\mathcal{D}'_\gamma$ and $\mathcal{D}'_\beta$ are unchanged by the update.

Implementation notes. The DFA mask in line 6 is realised as a sparse COO tensor indexed by (q^γ, y), so the cost of one mask application is the out-degree of q^γ, not $|V|$. The candidate enumeration in line 11 is bounded by the depth-capped $\mathcal{D}'_\gamma$ lookahead and reuses the cached backward DP C_β, so $\beta_{\text{fac}}(c)$ is one lse over H^2 entries plus a single C_β table read. Working-set memory stays under a few megabytes at the scales of Section 5.

A7 Benchmarks for Executable and Safe Embodied Planning

Existing benchmarks document the cost of ignoring these distinctions. The Embodied Agent Interface (EAI) (Li et al., 2024) and VirtualHome (Puig et al., 2018) test whether agents produce executable action sequences for household tasks, while BEHAVIOR examines longer-horizon plans in richer simulated settings. VLABench (Zhang et al., 2025c) adds a visual grounding requirement, where plans must be consistent with observed scene geometry rather than abstract object names. SafeAgentBench (Yin et al., 2024) shows that task completion and safety can come apart: agents frequently finish a goal task by taking actions that violate stated constraints. What these benchmarks collectively show is that task success alone is not sufficient; plans must also be valid, grounded, executable, and safe. Evaluating constrained decoding for embodied planning demands all four.

A8 Setup details

This section expands on the abbreviated setup of Section 5.

Models and serving. Text-only experiments use Llama-3-8B-Instruct (Grattafiori et al., 2024); multimodal experiments use Qwen3-VL-8B-Instruct (Bai et al., 2025) served via vLLM with thinking mode disabled (`enable_thinking=False`) to prevent `<think>` tokens from corrupting JSON / LTL parsing. External baselines (Safety Chip, LLM-TAMP-prompt, NL-Prompt) share the same Qwen3-VL-8B backend so that the constraint-enforcement *mechanism*—not the LLM choice—is what is being compared. No planner component is fine-tuned. Greedy decoding (T=0) throughout, except per-instance TTA which draws M=5 stochastic continuations.

HMM bridge. We distill conditional HMMs (H=128) from ground-truth continuations, one per benchmark module. The Qwen3-vocabulary HMM used for VLABench Mode B and TaPA is obtained by remapping the Llama-tokenised checkpoint (Section A11); without the remap, every Mode B / TaPA output collapses to the empty-sequence floor. Cross-references: HMM forward / backward operators are in Section A3; the backward DP table C_β used by $\mathcal{D}_\beta$ is in Section A4; the vision-conditioned variant used for CLAMP-full on multimodal benchmarks is in Section A12; per-instance TTA derivations and stability analysis are in Section A21.

External-baseline harness. Safety Chip (Yang et al., 2024) translates an NL constraint to LTL, monitors the completed plan with a Spot DFA, and reprompts the LLM on violation (up to three times). LLM-TAMP-prompt (Wang et al., 2024) adds a CoT-then-JSON output instruction and one parse-failure retry; on EAI it reduces to a structured CoT baseline because no motion-feasibility checker is available. NL-Prompt verbalises the same constraints as natural language, leaving the decoder unchanged. Mechanism comparison and run scripts are in Section A26; NL-Prompt design and hypothesis statements are in Section A27.

A9 Hyperparameters

Table 6 lists the defaults used throughout Section 5. Grid sweep ranges are $\gamma_{\text{scale}} \in \{10^{-3}, 5\cdot$

Parameter	Default	Effect
γ_{scale}	0.1 (AS) / 0.05 (GI)	weight of affordance-DFA soft bonus (DFA-only mode)
λ	1.0	weight of fused HMM/PDDL factor in full CLAMP
`lookahead_cap`	100	max DFA states for backward-DP BFS depth
`max_new_tokens`	768–2048	token budget (family-dependent)
B_{meta}	40	action budget for BEHAVIOR / VLABench
H (HMM hidden)	128	number of HMM hidden states

Table 6: **Hyperparameters for CLAMP.**

$10^{-3}, 10^{-2}, 5 \cdot 10^{-2}, 10^{-1}, 3 \cdot 10^{-1}, 1\}$ (BEHAVIOR / VLABench Mode A) and $\{0, 0.07, 0.1\}$ for the TTA stack where γ is fixed at 1.0. TTA-specific knobs: M=5 unconstrained continuations per prompt, chunk size Δ=24 tokens, $\eta_{\text{O}}=1{\cdot}10^{-3}$, λ_{O}=0.2, $\lambda_{\text{TTDA}}=10^{-2}$, 5 TTDA epochs.

A10 Simulator determinism

$\mathcal{D}_\beta$'s hard reachability guarantee assumes deterministic transitions. Table 7 summarises how that assumption holds for each benchmark. Symbolic simulators (VirtualHome `evolving_graph`, AI2-THOR) satisfy it exactly. Physics simulators (iGibson, MuJoCo) satisfy it only under a fixed seed and closed-loop position-controlled skills with ≤ 5 mm tolerance; in that regime δ_β degrades to a plan-feasibility certifier (correct under the assumed deterministic transition).

A11 Llama → Qwen3 emission remap

Why the remap is necessary. The emission matrix $B \in \mathbb{R}^{H \times |V_\ell|}$ of an HMM trained under the Llama-3 tokeniser cannot be applied unchanged under the Qwen3-VL tokeniser: the two assign different token IDs to the same surface strings. Applying an unmapped checkpoint produces empty `{"skill_sequence":[]}` outputs in $480/480$ Mode B prompts because β steers the decoder toward the wrong Qwen3 token IDs. Table 8 gives four representative entries illustrating the failure: the HMM places strong probability mass at the *Llama* token id for `"pick"` (+14.62), but Qwen3-VL generates a different id for that surface string, so the bonus lands on a token Qwen3 never emits in this context.

Remap algorithm. We remap by re-tokenising every Llama-3 surface string under the Qwen3 tokeniser and projecting B onto the Qwen3 vocabulary V_q ($|V_q|$=151,936). For surface strings encoded as a single Qwen3 subtoken we copy the column directly; for multi-subtoken strings we take the geometric mean of the per-subtoken probabilities; rows with no usable encoding fall back to a uniform smoothing prior 10^{-9}.

Table 9 reports the post-remap coverage. Mean row-entropy ratio $\beta_q/\beta_\ell = 1.020$ (within 2% of the Llama original); emission mass on the canonical skill tokens (`pick`, `place`, `press`) stays within 1 dex of the source, confirming structural preservation. All Mode B / TaPA experiments use the remapped checkpoint; TTBA and OTTA updates operate in Qwen3 token space.

Practical consequence. Without this remap, every reported Mode B number reduces to the empty-sequence floor. With the remap, the source HMM (R0) recovers non-trivial behaviour and TTA is applied on top (Section A17). The remap is therefore a prerequisite, not an ablation knob.

A12 Vision-conditioned HMM

The vision-conditioned HMM augments the source emission B with a per-prompt vision context c extracted from the VLM's image-level encoder output (last layer, mean-pooled across patches). At distillation, we condition each ground-truth continuation on its matching context vector and learn a context-modulated emission $B(\cdot, \cdot \mid c) = B \odot \exp(Wc)$, where $W \in \mathbb{R}^{|V| \times d_c}$ is a low-rank ($\text{rank} = 64$) projection and $\odot$ is the column-wise Hadamard product. At decode time, c is computed once per prompt from the prompt's image and held fixed for

Table 7: Simulator determinism per benchmark.

Benchmark	Simulator	Transition model	Determinism	$\mathcal{D}_\beta$ regime
EAI VirtualHome	VH `evolving_graph` v2.3.0	symbolic graph	exact	hard guarantee
EAI BEHAVIOR	iGibson / OmniGibson	rigid-body physics	under fixed seed	plan-feasibility
VLABench	MuJoCo 3.2.2 + dm-control	RK4 + skill ctrl	under fixed seed	plan-feasibility
TaPA-60	AI2-THOR v5.0	symbolic state mach.	exact	hard guarantee
SafeAgentBench	AI2-THOR v5.0	symbolic state mach.	exact	hard guarantee

Surface string	Llama-3 ID	Qwen3-VL ID	$\max \log \beta_{\text{Llama}}$
`"pick"`	30345	29245	$+14.62$ (high)
`" pick"`	3820	3735	$+1.84$
`"place"`	2050	2007	-1.09
`"skill_sequence"`	30554	multi	-1.21

Table 8: **Llama-3 → Qwen3 token-id mismatch on the behavior-trained HMM.** `max log` β is the maximum entry in the (Llama-tokenised) emission row at the Llama-3 ID. Because Qwen3 emits a different ID for the same surface string, that mass is unreachable from the Qwen3 decoder.

the whole sequence. The training objective is the same per-token NLL as the unimodal HMM distillation (Zhang et al., 2024) plus a Frobenius regulariser $\lambda_W \|W\|_F^2$ to prevent vision drift on prompts with weak image cues. On VLABench (Table 1), the vision-conditioned variant adds $+1.6$ pp Total over the text-only HMM and is the source of the CLAMP-full headline number 38.7.

A13 VLABench Mode A: in-domain HMM γ sweep

Table 10 reports the full γ_{scale} sweep on the in-domain HMM (`hmm-vlabench-h128`) using the Llama-3-8B backbone on the 480-prompt VLABench Mode A slice (no vision). DFA-only is the strongest configuration (0.296 Total). Cross-domain HMMs (`HMM-VH`, `HMM-BEH`) are numerically identical because both checkpoints place near-uniform mass on VLABench skill vocabulary. The in-domain `HMM-VLABench` grid shows the monotone collapse described in Section 5.4 D1: at γ=1.0 Total drops to 0.119, 0.164 *below* the cross-domain HMMs.

Table 11 extends to per-dimension scores across both backbones. Best-of-grid CLAMP points retain the γ-only gain on five of six dimensions (Mode A) and four of six (Mode B); PhysicsLaw collapses to 0.000 under Qwen3-VL DFA-only because that dimension is QA, not skill sequencing (Section 5.4 D2).

A14 VLABench weighted-DSL score

The VLABench weighted-DSL score (range 0–100) used in Table 1 is computed by VLABench's official evaluator from three per-task components: (a) `skill_match`, the fraction of predicted skill verbs that match the ground-truth sequence under longest-common-subsequence alignment; (b) `entity_match`, the fraction of predicted entity arguments that resolve to ground-truth entity IDs in the scene; (c) `exact_match`, 1 iff both `skill_match` and `entity_match` are 1.0 for the entire plan. The Total reported in Table 1 is $\text{mean}(\texttt{skill_match}, \texttt{entity_match}, \texttt{exact_match})$, averaged over the 480-prompt slice and length-weighted within each of the six dimensions. Per-component breakdowns for Mode A and Mode B are in Table 10 and Table 20.

A15 EAI VirtualHome AS / SD: full results

Table 12 compares CLAMP on Llama-3-8B-Instruct against (i) every proprietary planner surveyed in EAI, (ii) two external constraint-enforcement baselines run under our Qwen3-VL-8B backend. On Action Sequencing, unconstrained Llama-3-8B scores 21.3%; CLAMP-token raises this to 48.7% (+27.4 pp) by eliminating format, hallucination, and arity errors; CLAMP-full reaches **85.3**% (+36.6 pp further, +64.0 pp total). On Subgoal Decomposition: 48.8% → 60.1% → **79.8**% (+31.0 pp total). CLAMP-full exceeds every proprietary model on AS, including Claude-3.5 Sonnet (76.1), GPT-4o (71.5), and Gemini-1.5 Pro (76.7); on SD it trails Claude-3.5 Sonnet (89.1) and Gemini-1.5 Pro (87.0), a residual gap that traces to incomplete preconditions in `VHBetaStateMachine` (Section 5.4, D3). The two external baselines, run on a stronger Qwen3-VL-8B backend, both sit between Baseline and CLAMP-token on grammar but cannot close the goal-directedness gap: Safety Chip reaches 44.3% Task SR by reprompting on format violations

Mapping	Count	Share
Single Llama subtoken	109,916	72.3%
Multi Llama subtokens (geometric mean)	41,753	27.5%
No usable encoding (smoothing fallback, 10^{-9})	267	0.18%
Mean row-entropy ratio β_q/β_ℓ	1.020	(+2.0%)
Emission-mass ratio on canonical skill tokens	0.846–1.058	within 1 dex

Table 9: **Llama-3 → Qwen3 emission-matrix remap (Stage A).** Counts and shares across the 151,936 Qwen3 ids.

Method	Total	Skill	Entity	Exact	Fmt-valid
Baseline (no constraint)	0.169	0.459	0.048	0.000	0.967
DFA-only (affordance mask)	**0.296**	0.494	**0.343**	**0.052**	0.933
CLAMP HMM-BEH (γ=1.0)	0.283	0.465	0.330	0.054	0.973
CLAMP HMM-VH (γ=1.0)	0.283	0.465	0.330	0.054	0.973
CLAMP HMM-VLABench (γ=0.01)	0.277	0.456	0.324	0.052	0.963
CLAMP HMM-VLABench (γ=0.05)	0.265	0.436	0.311	0.048	0.973
CLAMP HMM-VLABench (γ=0.10)	0.248	0.409	0.289	0.046	**1.000**
CLAMP HMM-VLABench (γ=0.30)	0.196	0.329	0.225	0.035	0.910
CLAMP HMM-VLABench (γ=0.50)	0.126	0.229	0.149	0.000	0.833
CLAMP HMM-VLABench (γ=1.00)	0.119	0.217	0.141	0.000	0.794

Table 10: **VLABench Mode A — Llama-3-8B-Instruct.** Total / skill-match / entity-match / exact-match / format-valid (%) on the fixed 480-prompt slice. DFA-only is the strongest configuration; cross-domain HMMs (`HMM-VH`, `HMM-BEH`) are numerically identical because both checkpoints place near-uniform mass on VLABench skill vocabulary. The in-domain `HMM-VLABench` grid shows monotone collapse as γ_{scale} grows.

(which CLAMP-token avoids by construction), and LLM-TAMP-prompt reaches 51.1% by eliciting chain-of-thought reasoning. The 34.2 pp gap between LLM-TAMP-prompt and CLAMP-full is what only the world-state-aware β-DP backward pass closes—direct evidence for headline (F2) (Section 5.1). SD numbers for Safety Chip and LLM-TAMP-prompt are pending (BEHAVIOR adapter still running, 2026-05-07).

A16 BEHAVIOR (γ-only stage)

BEHAVIOR has three modules: Goal Interpretation (GI), Action Sequencing (AS), and Subgoal Decomposition (SD). Table 13–14 report results at the γ-only stage; the full $\mathcal{D}_\beta$ stack is deferred until the BEHAVIOR backward-DP integration finishes. On GI, the affordance DFA misspecification reduces F_1 from 0.2524 (unconstrained) to 0.1481 (DFA-only); CLAMP (with the HMM bridge) partially recovers to 0.1905 at γ=0.05 but remains below baseline—a diagnostic instance analysed in Section 5.4, D4. On AS, CLAMP-token improves Task SR slightly (12% → 13%) while eliminating parse errors entirely. On SD, the same configuration recovers parse-error rate to zero but Task SR *drops* relative to baseline—the DFA itself is misspecified for the SD output format, analysed in Section 5.4, D4. This is the empirical instance of headline (F4) (Section 5.1).

A17 TTA tier ablation: full results

?? reports the seven-row factorial across the three TTA tiers (TTDA, OTTA, TTBA) on the same 480-prompt VLABench slice as Table 1, with three unconstrained / DFA-only references at the top. All seven rows are complete (locked 2026-05-07).

Positive headline (+17.6× on PhysicsLaw). The full TTA chain (R3 / R5) reaches **59.0** on Physical Laws, against 3.4 for the unconstrained baseline (17.6× multiplicative, +55.6 absolute) and 0.0 for DFA-only. The PhysicsLaw breakdown is `skill_match` = 1.000 (every prompt gets the right verb) and `entity_match` = 0.385 (limited by VLM visual grounding, not the HMM). The decomposition isolates TTBA as the sole contributor: $(\text{R3} - \text{R1})_{\text{PL}} = +52.6$ in the absence of OTTA, and $(\text{R5} - \text{R3})_{\text{PL}} = 0$ confirms OTTA is inactive. This is the strongest single-task TTA result in the paper and isolates a regime where TTA + HMM lookahead is strictly required.

Dimension	Llama-base	Llama-γ	Llama-$\gamma\beta$@0.01	Qwen3-base	Qwen3-γ
CommonSense	0.189	0.347	**0.353**	0.227	**0.269**
Complex	0.197	**0.327**	0.302	0.243	**0.292**
M&T	0.219	**0.386**	0.369	0.226	**0.379**
PhysicsLaw	0.051	**0.108**	0.071	**0.034**	0.000
Semantic	0.166	**0.307**	0.283	0.214	**0.292**
Spatial	0.188	**0.297**	0.278	0.256	**0.269**

Table 11: **VLABench per-dimension totals.** Best runs of each method per backbone (Llama Mode A / Qwen3-VL Mode B). PhysicsLaw collapses to 0 under Qwen3-VL DFA-only because PhysicsLaw is QA, not skill sequencing; M&T is the most robust (γ-only 0.386 Llama vs. 0.379 Qwen3-VL, within noise).

Honest negative on the other five dimensions. On the five non-PhysicsLaw dimensions, every TTA configuration scores **0.0** versus DFA-only at 30.1. The mechanism is *single-skill collapse*: TTBA's M=5 samples are dominated by short JSON completions (`press`), so the per-instance β estimate concentrates on that one skill regardless of the actual task. Once β is concentrated, the lookahead bonus dominates the DFA mask + base LM scores, and the model emits the wrong skill on the 402 non-PL prompts. This is the same length-prior pathology that produces D1, now appearing at the emission level.

OTTA contributes exactly zero. $(\text{R4} - \text{R1}) = (\text{R5} - \text{R3}) = 0$ on every dimension. Three failure mechanisms drive this: self-reinforcement of wrong beliefs (OTTA updates from the model's own output, not GT); near-uniform entity distribution within tasks erases the per-task signal; the full-matrix update density 7.7×10^{-7} is underdetermined by ~ 15 generated tokens. A diagnosis of all three appears in Section A22; an entity-subspace OTTA variant (β update restricted to entity-ID columns, density $\times21000$) is queued.

A18 SafeAgentBench: compilation, mechanism analysis, and Phase 2

This section expands the one-paragraph headline of Section 5.1.

Benchmark structure. SafeAgentBench differs from EAI and VLABench in that it provides *explicit safety labels*: the question is not whether the plan reaches the goal, but whether it avoids prescribed hazards. This separates CLAMP's constraint-enforcement claim from its task-success claim and makes the comparison against Safety Chip and LLM^3-TAMP particularly clean, since both baselines were originally designed for exactly this kind of scenario. The benchmark splits into two regimes.

Long-horizon 50: each task carries an explicit NL "Requirement" clause (e.g. *"Turn off the stove burner within no more than two steps of turning it on"*); we compile this into a $\mathcal{D}_\gamma$ over the 17-action vocabulary plus a $\mathcal{D}_\beta$ temporal rule (BEFORE / WITHIN patterns) and generate the full plan under joint $\gamma + \beta$ enforcement. Evaluation is LLM semantic judgement (RR(LLM)) using Claude Haiku 4.5 with 3-vote majority, so AI2-THOR execution is not strictly required for the headline number. **Detailed** 300 **+ Abstract** 100: the hazard is implicit in the instruction; CLAMP cannot compile a per-task constraint, so we precompile ten safety rules encoded in $\mathcal{D}_\beta$—one per hazard class in the SafeAgentBench taxonomy (Fire, Electrical Shock, Explosion, Poisoning, Slip, Liquid Damage, Breakage, Appliance Misuse, Furniture Damage, Small Item Damage)—and report whether the generated plan avoids violating these rules. The Rejection-Rate metric does not apply directly: CLAMP prevents violating actions *structurally* (by logit masking) rather than *refusing the task at the LLM level*; the relevant comparison metric is RR(LLM) on the generated plan.

Metric directions (Table 2). The main table groups splits into two regimes because Rej and Compl. have opposite desirable directions in each. On the *hazard* splits (Unsafe-Detailed, Abstract) rejecting a hazardous instruction is good (Rej↑) and completing it is bad (Compl.↓). On the *legitimate-task* splits the instruction is something the agent *should* carry out, so the roles reverse: completing is desirable (Compl.↑) and a blanket rejection is a failure (Rej↓). This regime covers *Long-horizon*, where the agent must complete the task while honoring a temporal Requirement (its headline signal C-Safe↑ / C-Unsafe↓ does not reverse), and the already-safe *Safe-Detailed* control (rejecting it would be over-refusal). The $\mathcal{D}_\beta$ post-hoc columns

Model/Method	Action Sequencing		Subgoal Decomposition	
	Task SR	Execution SR	Task SR	Execution SR
Claude-3 Opus	64.6	69.5	86.7	89.9
Claude-3.5 Sonnet	76.1	81.3	89.1	92.0
Gemini 1.5 Pro	76.7	83.6	87.0	91.1
GPT-4o	71.5	81.3	87.6	91.1
o1-mini	71.5	76.4	79.3	84.6
o1-preview	65.2	72.5	89.4	93.2
Llama 3 8B Instruct	21.3	23.6	48.8	58.0
Llama 3 70B Instruct	59.0	66.6	78.4	87.3
Mistral Large	78.4	84.6	84.3	92.0
Mixtral 8x22B MoE	63.3	67.9	80.5	90.2
Baselines				
BASELINE (Llama-3-8B, unconstrained)	21.3	23.6	48.8	58.0
Safety Chip† (Yang et al., 2024)	44.3	54.1	–	–
LLM-TAMP-prompt† (Wang et al., 2024)	51.1	58.7	–	–
CLAMP on Llama-3-8B-Instruct				
CLAMP-token (α_{syn}, $\mathcal{D}_\gamma$ mask)	48.7	51.5	60.1	79.0
CLAMP-full ($\alpha_{\text{syn}} + \alpha_{\text{reach}}$, ours)	**85.3**	**91.8**	**79.8**	**82.5**

Table 12: **EAI VirtualHome: full leaderboard** (Task SR / Execution SR, %, on Action Sequencing and Subgoal Decomposition). Reference models (top) are compared against constraint-enforcement baselines (middle) and CLAMP (bottom, highlighted), all on Llama-3-8B-Instruct without fine-tuning; † marks external baselines run under our Qwen3-VL-8B-Instruct backend. Headline result summarized in Section 5.1 (main paper); configuration definitions in Section 5.

Method	Node F_1	Edge F_1	GI F_1
Llama-3-8B baseline (no constraint)	0.281	0.224	**0.2524**
DFA-only (affordance mask)	0.197	0.099	0.1481
CLAMP (ours, γ=0.05)	0.241	0.140	0.1905
CLAMP (γ=0.10)	0.225	0.121	0.173
CLAMP (γ=0.30)	0.198	0.082	0.140
CLAMP (γ=1.00)	0.183	0.038	0.111

Table 13: **BEHAVIOR Goal Interpretation (GI).** F_1 on node and edge goal predicates, plus headline GI F_1 (%). DFA-only is the affordance hard-mask baseline; full CLAMP adds the HMM-bridged PDDL factor at the per-token granularity required by GI's nested-list output. Best CLAMP grid point is γ_{scale}=0.05, cap =100.

(Plan-clean↑, Viol.↓) are split-independent.

Mechanism: backup-and-retry $\mathcal{D}_\beta$ decoding. CLAMP-token (Tier I, $\mathcal{D}_\gamma$ only) does *not* help: its β-violation rate (0.40) is statistically indistinguishable from the bare baseline's (0.41); format enforcement alone is insufficient when the safety constraint is temporal. The full Phase-1 win is attributable to a *backup-and-retry* extension of the decoder: when the $\mathcal{D}_\beta$ rejects a step's terminator, the decoder rewinds to the start of that step's verb, bans the verb at this position, replays the DFA + β state, and re-samples. A naive forward-only $\mathcal{D}_\beta$ (which only halts at violations) gives v1 numbers that are *worse* than baseline because they lose the rejection escape hatch.

Residual violations. The residual 0.17 β-violation rate has two known sources, both addressable in Phase 1 polish: (a) the model's top-k=20 candidate set occasionally never contains the constraint-required verb (forced-action injection at the deadline boundary, planned), and (b) verb-mapping drift between the LLM-compiled spec (`close`) and the action vocabulary (`toggle_off`); we currently bridge this with verb-equivalence groups, but a re-sampled compilation pass would tighten it further.

Method	Task SR ↑	PredArg ↓	Halluc ↓
Llama-3-8B baseline (no constraint)	0.21	0.32	0.18
DFA-only (post-processed)	0.10	0.27	0.14
CLAMP (ours, $\gamma{=}10^{-3}$)	0.06	0.19	0.21
CLAMP ($\gamma{=}5{\cdot}10^{-3}$)	0.06	0.19	0.21
CLAMP ($\gamma{=}10^{-2}$)	0.06	0.20	0.21
CLAMP ($\gamma{=}5{\cdot}10^{-2}$)	0.06	0.20	0.22

Table 14: **BEHAVIOR Subgoal Decomposition (SD).** Task success rate, parsed-argument count error (lower better), and hallucination rate (%). We sweep $\gamma_{\text{scale}} \in \{10^{-3}, 5{\cdot}10^{-3}, 10^{-2}, 5{\cdot}10^{-2}\}$ at $T{=}0$. The HMM induces *item fragmentation* ($\sim$90 items/sample vs. 1 large item for DFA-only), driving up hallucination—a diagnostic finding revisited in Section 5.4.

Compilation pipeline and wall-clock. Compiling the 50 NL Requirement clauses into $\mathcal{D}_\beta$ specs uses Qwen3-VL itself and runs in under one minute total; we cache the compiled specs. Per-task wall-clock on a single H100 is 3–5 s for CLAMP-token and 2–8 s for CLAMP-full (variance from backup-and-retry depth).

Phase 1 results (long-horizon). With the backup-and-retry decoding above (and forced action injection at the deadline when the top-k candidates lack the required verb), CLAMP-full lifts the LLM-judged C-Safe rate $0.06 \rightarrow \mathbf{0.18}$ ($3\times$), the post-hoc $\mathcal{D}_\beta$ plan-clean rate $0.52 \rightarrow \mathbf{0.94}$ ($+42$ pp), and reduces the per-constraint violation rate $0.41 \rightarrow \mathbf{0.05}$ (-88% relative).

Phase 2 (Detailed + Abstract): full-scale results. For the splits without a per-task Requirement clause we instantiate the universal $\mathcal{D}_\beta$ from each task's hazard label, with an 18-alias mapping covering observed upstream label variants. On the full 300-task unsafe-detailed split the post-hoc β-violation rate drops $0.205 \rightarrow \mathbf{0.021}$ (a $\mathbf{10}\times$ reduction; plan-clean $0.74 \rightarrow \mathbf{0.97}$), and on the full 100-task abstract split $0.180 \rightarrow \mathbf{0.043}$ (a $\mathbf{4}\times$ reduction). The residual non-zero rates trace to two upstream hazard categories (*Inaccessibility*, *Physical Harm / Poisoning*) not yet mapped into our 10-class taxonomy. The LLM judge tracks the structural-mask signal at full scale where it was previously saturated: on unsafe-detailed-300, RR(LLM)-safe lifts $0.16 \rightarrow \mathbf{0.27}$ ($+11$ pp) and RR(LLM)-unsafe drops $0.32 \rightarrow \mathbf{0.23}$ (-9 pp); on abstract-100 the judge moves $+3$ pp on RR-safe and the $\mathcal{D}_\beta$ signal is the cleaner one. A safe-detailed-300 control (CLAMP-token only, no β) shows no significant effect (-1 pp on RR-safe, within judge noise), confirming $\mathcal{D}_\beta$ is the load-bearing component.

Cross-judge replication. A cross-judge with Claude Sonnet 4.6 corroborates the directional gain on abstract: Sonnet rates the bare baseline more leniently (0.56 vs. Haiku's 0.36) but agrees CLAMP-policy is at least as safe ($0.60 \geq 0.56$).

Execution-based RR(goal) (Phase 3). The execution-based RR(goal) protocol over AI2-THOR is implemented as a runnable scaffold (`run_rr_goal.py`) but requires Vulkan/X11 infrastructure that was unavailable on the dev host; deferred to camera-ready.

A19 TaPA-60: full results and external-baseline status

VLABench's skill vocabulary is closed (`pick / place / pour / press`); TaPA tests the orthogonal regime of *open-vocabulary free-form NL planning* grounded to a scene-level object list acquired by an open-vocabulary detector (Detic / Grounding-DINO over six AI2-THOR multi-view RGB images). This is the cleanest single-benchmark probe of CLAMP's Tier-II visual-grounding constraint $\mathcal{D}_{\mathcal{O}_{\text{seen}}}$ in the literature. Table 15 reports plan-success rate on TaPA's 60-triplet evaluation set. Three observations. *(i)* A modern 2025 VLM (Qwen3-VL-8B, no constraints) reaches 71.7%, exceeding the TaPA paper's fine-tuned LLaMA-7B headline (61.1%); the relevant comparison frame is therefore "does CLAMP help on top of a strong VLM?", not "does it match TaPA's fine-tuning result". *(ii)* On Kitchen scenes (the most complex, hallucination-prone plans), CLAMP-vg gains $+14.3$ pp over the bare baseline ($35.7 \rightarrow 50.0$), and the hallucination rate drops from 10.0% to **8.3**% (-17% relative). *(iii)* Average plan-success drops 3.4 pp ($71.7 \rightarrow 68.3$), within the noise band of the 3-vote Qwen3-VL self-judge: the same self-judge has been shown to issue divergent verdicts on byte-identical plans (Section A20). The pattern *wins on*

Method	Kit.	Living.	Bed.	Bath.	**Avg**	Halluc.↓
Reference rows from (Wu et al., 2023), 3-volunteer human judgement						
LLaVA-7B	14.3	42.1	33.3	0.0	22.4	—
LLaMA-7B (no fine-tune)	0.0	10.5	13.3	0.0	6.0	—
GPT-3.5	28.6	73.7	66.7	50.0	54.7	—
TaPA fine-tuned LLaMA-7B	28.6	84.2	73.3	58.3	61.1	—
Our Qwen3-VL-8B variants, 3-vote Qwen3-VL self-judge						
Qwen3-VL-8B baseline (no constraint)	35.7	**84.2**	**80.0**	**83.3**	**71.7**	10.0%
CLAMP-token (verb DFA)	42.9	**84.2**	73.3	58.3	66.7	10.0%
CLAMP-vg ($+\mathcal{D}_{\mathcal{E}_{\text{seen}}}$)	**50.0**	**84.2**	73.3	58.3	68.3	**8.3%**

Table 15: **TaPA-60 multimodal embodied planning** (Wu et al., 2023). Plan-success rate (%) on TaPA's 60-triplet evaluation set (Kitchen 14 / Living 19 / Bedroom 15 / Bathroom 12). Reference rows are the TaPA paper's headline numbers (LLaVA-7B, GPT-3.5, fine-tuned LLaMA-7B) judged by 3 human volunteers, majority vote; our rows use Qwen3-VL-8B-Instruct (thinking disabled) and a 3-vote Qwen3-VL self-judge as a stand-in for the human panel (calibration in App. A20). CLAMP-vg combines $\mathcal{D}_\gamma$ over the verb vocabulary + visual-grounding mask; CLAMP-token uses the verb DFA only. The Qwen3-VL bare baseline already exceeds the TaPA paper's fine-tuned result 61.1, so the comparison frame is "does CLAMP help on top of a strong 2025 VLM?". Halluc. rate is the fraction of plans that reference an out-of-scene object (lower is better).

hard scenes, loses on easy scenes is the expected signature of a constraint that helps when the model would otherwise fail and is otherwise overhead.

External-planner baselines on multimodal benchmarks. TaPA-FT (Wu et al., 2023), VLM-TAMP (Yang et al., 2025), and LLM^3-TAMP (Wang et al., 2024) are the natural external baselines for open-vocabulary multimodal planning. The published TaPA-FT row in Table 15 is our reference point on TaPA-60; VLM-TAMP and LLM^3-TAMP on VLABench require dedicated env adapters and are deferred (plan in `docs/040safety_benchmark_setup_plan.md` §10).

A20 TaPA judge calibration and failure modes

The TaPA paper (Wu et al., 2023) judges plan success with 3 human volunteers per plan (success iff ≥ 2 vote yes). Recruiting human volunteers is not feasible at our iteration cadence; we adopt a **Qwen3-VL-8B-Instruct self-judge** with the same rubric, called 3 times per plan at T=0.3 with seeds $\{42, 43, 44\}$ and majority vote. The self-judge is fast (180 calls in $\sim$7 min) and deterministic under fixed seeds.

Two known failure modes. First, the self-judge produces *different verdicts on byte-identical plans* across configs: on five regression cases between baseline and CLAMP-vg, two pairs (id 40 *turn on the lamp*; id 54 *clean the bathroom*) had identical plans but $3/3$ True for one config and $0/3$ True for the other. The plan content was the same—only the surrounding harness metadata differed. Second, on long plans (≥ 25 steps) the self-judge misclassifies in-scene objects as hallucinations: id 51 v2's 32-step plan was flagged $3/3$ False with reasoning *"hallucinated objects: TissueBox, Plunger, ScrubBrush, Candle, ToiletPaper"*, all five of which were in the scene's GT object list.

Implication for the headline. Both failure modes inflate the apparent *baseline* > CLAMP gap on TaPA. After excluding the two identical-plan judge-bias cases and the truncation case (id 58), the corrected CLAMP-vg score is $\sim$70.0% vs. baseline 71.7%, within noise. Reproducing TaPA-60 with an external (non-Qwen3) judge is a planned follow-up.

A21 Test-time adaptation: derivation, equations, and stability

Why TTA is necessary. Distilled once on an offline corpus, the HMM marginalises over prompts in that corpus, and three drift axes degrade its fit during deployment. *Cross-domain:* skill vocabulary, container names, and affordance conventions shift between benchmarks—a VirtualHome-trained HMM on BEHAVIOR shows a 56.6% jump in held-out per-token NLL. *Within a session:* episodes in the same environment share lexical patterns that a global HMM ignores. *Within a plan:* later chunks should reflect lexical commitments made by earlier chunks (which container was opened, which entity named), which a static B cannot track.

Next-token form of the HMM loss. The test-time objective is the self-supervised next-token NLL of the test sequence under the HMM,

$$\mathcal{L}_{\mathrm{TTA}}(\theta) \;=\; -\sum_{n=0}^{N-1} \log P_\theta\big(y_{n+1} \mid y_{1:n}\big), \quad (10)$$

which requires no external label: the test sequence supervises itself through the chain rule, exactly as in standard language-model pre-training. Let $f_n(h;\theta) = \log P_\theta(y_{1:n},\, h_n = h)$ be the log-domain joint forward variable; the one-step probability under the HMM is

$$P_\theta\big(y_{n+1} \mid y_{1:n}\big) \;=\; \frac{\sum_{h'}\big[\sum_h \exp(f_n(h))\, A(h,h')\big]\, B(h',\, y_{n+1})}{\sum_h \exp(f_n(h))}, \quad (11)$$

which makes Eq. (10) a self-supervised objective on any test-sequence prefix. Following Feng et al. (2026) we update only B and freeze (A,π). Throughout, we write $\Theta(B) = (\pi_{\mathrm{VH}}, A_{\mathrm{VH}}, B)$ for the HMM with adapted emission B and the frozen VirtualHome anchor $(\pi_{\mathrm{VH}}, A_{\mathrm{VH}})$.

Tier 1 (TTDA, offline domain adaptation). Run once before deployment on M unconstrained continuations $\{y^{(i)}\}_{i=1}^{M}$ sampled from the base VLM in the target domain:

$$B_{\mathrm{dom}} \;=\; \arg\min_{B}\Big[\tfrac{1}{M}\textstyle\sum_{i=1}^{M} \mathcal{L}_{\mathrm{TTA}}\big(\Theta(B);\, y^{(i)}\big) \;+\; \lambda_{\mathrm{TTDA}}\,\big\|B - B_{\mathrm{VH}}\big\|_F^2\Big], \quad (12)$$

solved by AdamW or full Baum–Welch; the Frobenius regulariser prevents multi-epoch optimisation from washing out source-domain structure.

Tier 2 (OTTA, online session adaptation). After prompt t, with $B_{\mathrm{dom},0} := B_{\mathrm{dom}}$, take one anchored gradient step:

$$\widetilde{B}_{\mathrm{dom},t} \;=\; B_{\mathrm{dom},t-1} - \eta_{\mathrm{O}}\, \nabla_B \mathcal{L}_{\mathrm{TTA}}\big(\Theta(B_{\mathrm{dom},t-1});\, y^{(t)}\big), \quad (13)$$

$$B_{\mathrm{dom},t} \;=\; \mathrm{logaddexp}\big(\log(1-\lambda_{\mathrm{O}}) + B_{\mathrm{dom}},\; \log\lambda_{\mathrm{O}} + \widetilde{B}_{\mathrm{dom},t}\big). \quad (14)$$

The session history accumulates while the geometric mixture in Eq. (14) bounds KL drift from the domain anchor.

Tier 3 (TTBA, per-plan chunk adaptation). During decoding, after every chunk of Δ tokens we take one gradient step on the chunk $y^{(t)}_{(c-1)\Delta:c\Delta}$ and project toward $B_{\mathrm{dom},t}$ as in Eq. (14). At the task boundary we reset $B \leftarrow B_{\mathrm{dom},t}$, so chunk-wise updates within plan t cannot influence plan $t' > t$. Because A is frozen, the joint forward variable is a sufficient statistic for the prefix when B changes, and the recursion resumes after each chunk without recomputing the prefix.

Decode-time cost. Per-step, evaluating the factored posterior of Proposition 1 costs $\mathcal{O}(|Q_\gamma| + |E_\beta|)$, additive in the recognizer sizes, rather than the $\mathcal{O}(|Q_\gamma|\cdot|S_\beta|)$ a product automaton would incur. Per plan, let H be the number of HMM hidden states, N the plan length in tokens, and Δ the TTBA chunk size. Baseline decoding costs $\mathcal{O}\big(N(H^2 + |Q_\gamma| + |E_\beta|)\big)$ per plan; TTDA is offline and does not enter the deployment budget. With forward-state carry, OTTA adds $\mathcal{O}(NH^2 + H|V|)$ per prompt and TTBA adds $\mathcal{O}(NH^2 + (N/\Delta)H|V|)$ per plan, giving

$$T^{\mathrm{full}}_{\mathrm{decode}} \;=\; \mathcal{O}\Big(N(H^2{+}|Q_\gamma|{+}|E_\beta|) + \tfrac{N}{\Delta}\, H\,|V|\Big), \quad (15)$$

with the $\frac{N}{\Delta}H|V|$ term dominant whenever $|V|$ is large. Empirical wall-clocks confirming this analysis (including the OTTA/TTBA overhead breakdown and the $\Delta \approx 24$-token chunk implied by the $\times 1.1 : \times 2.3$ ratio) are in Section A23.

Stability mechanisms. Four properties keep the pipeline safe in the small-sample regime. (i) $\mathcal{L}_{\mathrm{TTA}}$ equals the per-token NLL reported at evaluation, so any descent step strictly improves the reported quantity. (ii) The log-space mixture in Eq. (14) is a geometric mixture in probability space, bounding KL drift from the tier anchor after any number of updates. (iii) The Tier 3 task-boundary reset prevents any gradient from one plan reaching another, ruling out adversarial cross-prompt contamination. (iv) The feasibility structure—$\mathcal{D}'_\gamma$ (syntax DFA $\cap$ visibility DFA), $\mathcal{D}'_\beta$, and the action budget K—is symbolic and never depends on B. The reachability table C_β is computed from $\theta = (\pi, A, B)$, but its *support* (which states reach F_β at all, within K steps) is determined by $\mathcal{D}'_\beta$ and K alone; B only rescales the finite log-probabilities inside that support. Updating B therefore reweights probability mass *within* the feasible set without enlarging it, so Proposition 2 continues to hold throughout adapta-

tion and TTA cannot make a previously infeasible plan feasible.

A22 OTTA failure-mechanism diagnosis

In the TTA factorial (**??**), online task-level adaptation contributes *exactly* $+0.0$ on every parent configuration: $(\text{R4} - \text{R1}) = (\text{R5} - \text{R3}) = 0$ on every dimension. R1 and R4 differ in $321/480$ raw output strings, but only $6/480$ differ in skill name, and those six are random `open_door`↔`press` flips with no consistent direction. We attribute this to three compounding mechanisms.

(M1) Self-reinforcement of wrong beliefs. OTTA updates B from the model's *own decoded output*, not from ground truth. When β produces `open_door`, every OTTA update reinforces `open_door`. The self-supervised NLL (Eq. 10) is locally minimised by sharpening whatever emission distribution generated the current sequence—which is exactly the wrong signal when that sequence is incorrect.

(M2) Near-uniform entity distribution erases the per-task signal. Within a VLABench task family, entity IDs are near-uniform across prompts (e.g. `density_qa` entity histogram $\{1{:}38\%, 3{:}33\%, 5{:}29\%\}$). OTTA averages these into the session-level β, producing a near-uniform entity prior that provides no useful guidance at inference time.

(M3) Underdetermined full-matrix update. A typical generation produces ~ 15 tokens; B has shape $[H{=}128 \times |V|{=}151{,}936]$, so the update density per generation is 7.7×10^{-7}. Over the 80 episodes of a VLABench session this still leaves $> 99.99\%$ of B untouched, and the touched columns are dominated by short-JSON tokens (`press`, `{"name":`, etc.) rather than entity IDs.

Proposed fix: entity-subspace OTTA. Restrict the OTTA β update to the N_e entity-ID token columns (typically $N_e \sim 50$ per task family), increasing update density by $\sim 21000\times$ to $1.6{\times}10^{-2}$. This addresses (M3) without needing more samples; (M2) is mitigated because entity-ID columns become the only place updates land, so the per-task entity distribution gets full update budget instead of $50/151{,}936$ of it. (M1) remains a structural risk and would require either GT supervision or a confidence-gated update rule.

A23 Compute and wall-clock

All experiments run on a single NVIDIA GH200 (96 GB HBM) with bf16 weights for both Qwen3-VL-8B-Instruct and Llama-3-8B-Instruct.

Table 16 expands the headline rows of Table 3 (main paper, Section 5.3) into the full mode-level breakdown across the four experimental tracks (TaPA, TTA ablation, Safety, Engineering).

Table 17 reorganises the same numbers as a per-tier zoom-in: cumulative latency added by each constraint layer.

Why the DFA-only baseline runs $3.3\times$ faster than R0. DFA-only routes through the vLLM `/v1/completions` API with `guided_regex`, taking advantage of PagedAttention and batched inference. R0–R6 fall back to single-prompt HuggingFace `model.generate()` because the per-token GammaProcessor callback that injects β-DP scores into the logit stream is incompatible with vLLM's pre-compiled generation loop. The same limitation explains why the TaPA decoder uses a custom top-K logprob filter (per-token round-trip $\sim$50 ms) rather than vLLM-native constrained decoding.

Reducing the per-token callback cost. Profiling on the 5-prompt `add_condiment_common_sense` batch attributed 67% of R0's 13 s/prompt to the GammaProcessor.__call__ callback. Six bug fixes during this session reduced an early $\sim$97 s/prompt baseline by $7.5\times$:

- sparse COO conversion of T_{mask} (memory 26 GB $\rightarrow$ 4 MB for BEHAVIOR GI);
- `lookahead_cap` $= 100$ on the BFS expansion;
- mask precomputation moved out of the inner loop;
- batched GPU writes for `_get_valid_mask`;
- removal of redundant Python loops in the lookahead pass;
- a GammaProcessor refactor that fuses the forward-belief update with the emission lookup.

TTA overhead breakdown. TTBA's $\times 2.3$ overhead decomposes into 5 unconstrained Qwen3-VL `generate()` calls ($\sim$12.5 s), one Baum–Welch β update step ($\sim$1 s), and GammaProcessor re-initialisation with the adapted HMM ($\sim$0.2 s). OTTA's $\times 1.1$ overhead is a single Baum–Welch step at the end of each decoding ($\sim$1 s) plus a β carryover reset at task boundaries (negligible).

Track	Bench	Mode	Wall	s/prompt	Overhead
TaPA	TaPA-60	baseline (Qwen3-VL bare)	3.9 min	3.9	—
TaPA	TaPA-60	CLAMP-token (Tier-I)	3.9 min	3.9	$\times$1.0
TaPA	TaPA-60	CLAMP-vg (Tier-II)	5.0 min	5.0	$\times$1.3
Ablation	VLABench-480	DFA-only (vLLM batched)	32 min	4.0	—
Ablation	VLABench-480	R0 (no TTA)	105 min	13.2	$\times$1.0
Ablation	VLABench-480	R1 (TTDA)	103 min	12.9	$\times$1.0
Ablation	VLABench-480	R2 (TTBA)	242 min	30.2	$\times$2.3
Ablation	VLABench-480	R3 (TTDA + TTBA)	248 min	31.0	$\times$2.4
Ablation	VLABench-480	R4 (TTDA + OTTA)	119 min	14.9	$\times$1.1
Ablation	VLABench-480	*R5 (TTDA + OTTA + TTBA)*	$\sim$345 *min*	$\sim$43	$\times$3.3
Ablation	VLABench-480	*R6 (OTTA)*	$\sim$112 *min*	$\sim$14	$\times$1.1
Safety	SafeAgentBench-50	*baseline (Phase 0, planned)*	$\sim$5 *min*	$\sim$5	—
Safety	SafeAgentBench-50	CLAMP-*full (Phase 1, planned)*	$\sim$10 *min*	$\sim$12	$\times$2.4
Engineering	BEHAVIOR GI	sparse COO T_{mask} mask memory	one-time precompute 26 GB $\rightarrow$ 4 MB		

Table 16: **Wall-clock and per-prompt latency: full breakdown** (single GH200, bf16). Italic rows are estimates for runs not yet complete. *TTBA dominates overhead* ($\times$2.3, five unconstrained Qwen3-VL samples = 12.5 s); *OTTA is nearly free* ($\times$1.1); *DFA-only is* 3.3$\times$ *faster than R0* because it routes through vLLM batched inference, while R0–R6 fall back to single-prompt HuggingFace `model.generate()` — the per-token GAMMAPROCESSOR callback is incompatible with vLLM's pre-compiled generation loop. The 67% of R0 latency in the Python callback was reduced from 97 s/prompt to 13 s/prompt by six engineering fixes (sparse COO T_{mask}, mask precomputation, batched GPU writes, removed Python loops, lookahead cap, GAMMAPROCESSOR refactor; 7.5$\times$ speedup).

Layer (cumulative)	s/prompt	Marginal	Mechanism
Unconstrained Qwen3-VL-8B (vLLM)	3.9	—	batched inference, no callback
$+\,\alpha_{\text{syn}}$ ($\mathcal{D}_\gamma$ mask, vLLM)	4.0	+0.1 ($\times$1.0)	`guided_regex` (server-side)
$+\,\alpha_{\text{reach}}$ ($\mathcal{D}_\beta$ + HMM bridge)	13.2	+9.2 ($\times$3.3)	per-token GAMMAPROCESSOR callback
+ OTTA (online session β update)	14.9	+1.7 ($\times$1.1)	one Baum–Welch step / episode
+ TTBA (per-instance β EM)	30.2	+15.3 ($\times$2.0)	M=5 unconstrained TTT samples
Full chain (TTDA + OTTA + TTBA)	$\sim$43	—	combined

Table 17: **Per-tier wall-clock decomposition** (single GH200, bf16; **Marginal**: cost added by the row above). $\mathcal{D}_\gamma$ is essentially free; the per-token β-DP callback is the fixed cost of leaving vLLM batched inference; OTTA is nearly free; TTBA is the only expensive layer. Full track-level breakdown in Table 3.

Aggregate cost. The complete seven-row TTA ablation (R0–R6, 480 prompts each) costs $\sim$36 GPU-hours; TaPA (four configurations on 60 prompts plus the three-vote judge) costs $<$1 GPU-hour. Total project compute to date fits comfortably within 2 GPU-days of GH200 time.

A24 Per-dimension TTA scores

?? (referenced from the main text) reproduces the six-dimension breakdown of the TTA ablation. The all-zero columns for the five non-Physical Laws dimensions are not a typesetting artifact; they reflect the single-skill collapse failure mode discussed in Section 5.4 D5.

A25 Safety Chip backend calibration

Before applying Safety Chip as an external baseline on EAI VH/BEHAVIOR or SafeAgentBench (Section 5.1), we replicate its authors' VirtualHome demo set under our Qwen3-VL-8B backend with thinking disabled. Table 18 reports the expected pattern: 100% safety in the `full` arm, 30$-$38% in the `null` (no-LTL) arm, and an intermediate 54$-$62% in the `bad` (broken-LTL) arm. This confirms the implementation is faithful end-to-end before we use the same harness on the larger evaluation sets.

A26 External baseline implementations

Shared backbone. All four methods use `Qwen/Qwen3-VL-8B-Instruct` served via

Scene	Safety arm	Completed	Safe	Both
mobile_manip	full	49/50 (98%)	**50/50 (100%)**	49/50 (98%)
mobile_manip	bad	48/50 (96%)	31/50 (62%)	31/50 (62%)
mobile_manip	null	47/50 (94%)	15/50 (30%)	15/50 (30%)
rooms	full	49/50 (98%)	**50/50 (100%)**	49/50 (98%)
rooms	bad	50/50 (100%)	27/50 (54%)	27/50 (54%)
rooms	null	50/50 (100%)	19/50 (38%)	19/50 (38%)

Table 18: **Safety Chip backend calibration on its own VirtualHome demo set** (10 task-instruction pairs $\times$ 2 scenes $\times$ 5 trials, n=300 rollouts; Qwen3-VL-8B backend with thinking disabled). Three `safety_level` arms: `full` = correct LTL constraints, `bad` = deliberately broken LTL, `null` = no LTL (unconstrained planner). The `full` arm reaches 100% safety and $96-100\%$ task completion; the `null` arm drops to $20-44\%$ safe. This confirms the Safety Chip code path runs correctly under our Qwen3-VL-8B backend and with thinking disabled, so any deviation observed when applying it as an external baseline on EAI VH/BEHAVIOR or SafeAgentBench is attributable to the constraint specification, not to the implementation. Source: `ltl-safety/results` artifacts in `SueMintony/embodied-agents-results`.

vLLM at `http://localhost:8001/v1` (alias `qwen3-vl-8b`). Thinking mode is disabled (`enable_thinking=False`) on every call; without this, Qwen3 prepends a `<think>...</think>` block that breaks LTL formula extractors and JSON parsers in both external baselines. All runs use greedy decoding (T=0).

A26.1 Safety Chip

Safety Chip (Yang et al., 2024) enforces constraints through a *reprompt loop* that operates entirely outside the decoder. (1) The LLM generates a complete plan in one shot. (2) A rule-based parser checks whether each action violates a constraint (format, vocabulary, argument arity). (3) On any violation, an NL description of the failure is appended to the prompt and the LLM is queried again. (4) Steps (2)–(3) repeat up to three times; if no valid plan is produced, the last output is kept.

The original system uses NL→LTL translation and a Spot DFA monitor (Duret-Lutz et al., 2022) to check temporal safety properties. EAI tasks carry no explicit NL safety constraint, so we instantiate Safety Chip with the *action-format constraint*: a plan is valid iff every action name belongs to the allowed vocabulary with the correct argument count. This is equivalent to the token-level guarantee CLAMP-token provides structurally; the difference is that Safety Chip enforces it post-hoc through reprompting while CLAMP-token enforces it by masking invalid tokens to $-\infty$ before sampling.

Empirically, $\approx$15% of BEHAVIOR AS tasks exhaust all three reprompts without converging to a valid format; those outputs are passed to the evaluator as-is and score 0 on grammar metrics. CLAMP-token achieves 0% grammar errors by construction on the same tasks.

A26.2 LLM-TAMP-prompt

LLM^3-TAMP (Wang et al., 2024) is a planner that pairs a CoT-then-JSON LLM with a motion-feasibility checker (PyBullet collision + IK) that feeds categorised failure messages back to the LLM for iterative repair. On EAI, which provides no simulator, we run only the LLM component and call the result **LLM-TAMP-prompt** to distinguish it from the full system.

What changes. Each EAI prompt is augmented with a CoT output instruction:

> *Before giving your final answer, briefly reason about the task (1–3 sentences). Then output a JSON object:* `{"reasoning": "...", "actions": <action sequence>}`. *Output only the JSON object; no other text.*

The response is parsed to extract the `actions` field; one retry is attempted if parsing fails. There is no backtracking loop: unlike the full LLM^3-TAMP system, no feasibility signal is available, so the method is equivalent to a structured CoT baseline.

Why still useful as a baseline. LLM-TAMP-prompt tests whether eliciting explicit chain-of-thought reasoning before committing to a plan improves task success compared to the bare LLM, without any constraint enforcement. VH AS results (Section 5.1) show it reaches 51.1% Task SR vs. 21.3% for the bare baseline, confirming that structured prompting helps. The 34.2 pp gap to CLAMP-full (85.3%) cannot be closed by prompt engineering: it requires per-token goal-reachability lookahead ($\mathcal{D}_\beta$ backward DP), which LLM-TAMP-prompt does not have.

A26.3 CLAMP-token and CLAMP-full

CLAMP-token and CLAMP-full are described in Section 5. The key property that distinguishes both from the external baselines above: constraints are enforced *inside* the decoder by setting the logit of every violating token to $-\infty$ before sampling. No reprompting occurs; violations are structurally impossible rather than empirically unlikely.

Table 19 summarises the four methods.

Table 19: Mechanism comparison across all four methods evaluated on EAI. "Hard" means the constraint is a structural property of the generation process; "empirical" means it holds only when the LLM complies.

Method	Constraint location	Hard guarantee	Reprompt	VH AS Task SR
Safety Chip (Yang et al., 2024)	Outside decoder	Empirical	Up to 3×	44.3%
LLM-TAMP-prompt (Wang et al., 2024)	Outside decoder	None	1× (parse)	51.1%
CLAMP-token (DFA only)	Inside decoder	Yes (fmt/vocab/arity)	Never	48.7%
CLAMP-full	Inside decoder	Yes (+ reachability)	Never	**85.3%**

A27 NL-Prompt ablation

The NL-Prompt ablation verifies that encoding the same constraint content as natural language in the prompt—leaving the decoder unchanged—cannot match the structural guarantees of CLAMP's logit mask. Two variants are compared (Section 5.2):

- **NL-Lite**: the prompt is augmented with the action vocabulary list and JSON output schema.
- **NL-Full**: NL-Lite plus natural-language descriptions of every PDDL precondition and goal predicate (the same constraints that $\mathcal{D}_\beta$ enforces structurally in CLAMP-full).

H1 (format violations persist). Even with NL-Full, at least some outputs will contain format, vocabulary, or arity violations because NL enforcement is probabilistic: a single out-of-vocabulary token invalidates the plan regardless of how clearly the constraint was stated. CLAMP-token achieves 0% on these metrics by construction.

H2 (no lookahead ⇒ precondition violations persist). NL-Full cannot describe the per-token backward DP signal: stating "before `place(X, Y)`, `X` must already be held" in the prompt does not give the model any mechanism to verify this holds at the generation step where the token is emitted. Precondition violation rates under NL-Full are expected to remain close to the unconstrained baseline.

H3 (prompt bloat and hallucination). NL-Full prompts grow $O(|\text{tasks}| \times |\text{predicates}|)$ tokens. In multimodal settings, listing visual entities in the prompt does not suppress entity hallucination: the model can still emit unseen entity names because hallucination is a sampling-time phenomenon, not a prompt-content phenomenon.

A28 Diagnostic findings (full mechanism analysis)

This section expands the five findings of Section 5.4 with quantitative breakdowns and mechanism-level traces.

Method	Total	Skill	Entity	Exact	Fmt-valid
B-baseline (no constraint)	0.201	**0.598**	0.005	0.000	**1.000**
B-dfa_only (+ γ DFA)	**0.251**	0.406	**0.327**	**0.021**	**1.000**
B-ctrlg (+ $\gamma + \beta$, ours)	—	—	—	—	*pending*

Table 20: **VLABench Mode B — Qwen3-VL-8B-Instruct.** Same metrics as Mode A. B-dfa_only achieves perfect format validity (4,200/4,200 samples) and a +0.05 absolute total over B-baseline. *Full* CLAMP *on Qwen3-VL is pending* a `compile_implicit_to_graph + DFAModel` integration bug.

(D1) in-domain HMM actively degrades DFA-only. On VLABench Mode A (Table 10, Section A13), the DFA mask alone improves Total from 0.169 to 0.296. Adding the in-domain HMM *degrades* it monotonically: 0.277 at γ=0.01 down to 0.119 at γ=1.0, falling below DFA-only at all $\gamma \geq 0.05$. Cross-domain HMMs score 0.283 at γ=1.0, 0.164 above the in-domain HMM. Per-example analysis: HMM helps 56 prompts, hurts 129 (2.3× asymmetry). Root cause: the in-domain HMM learns VLABench's marginal sequence length (≈44 tokens), but H=128 cannot separate Physical Laws tasks (≈25 tokens) from Complex tasks (≈160 tokens), pulling short sequences long and dropping `format_valid` from 0.96 to 0.79. Per-instance TTA adapts on the current task's own VLM continuations and avoids this global length prior, motivating Section 4.3.

(D2) Physical Laws is a universal bottleneck. Physical Laws is the worst dimension in every configuration: Mode-A DFA-only ceiling is 0.108; Mode-B DFA-only (Table 20) is 0.000. These 80 tasks (16.7% of prompts) account for ≈40% of the Mode-A to Mode-B total gap. Root cause: every Physical Laws task is a QA query whose correct plan is a single `press[N]` call, but both Llama-3 and Qwen3-VL treat the scene as a manipulation task and emit `pick`-based plans. Qwen3-VL is more confidently wrong because visual grounding reinforces the manipulation interpretation. A targeted fix—rule-maker injects "`press[N]`

required; pick/place forbidden" for *_qa task names—should recover to the DFA-only ceiling without model changes. The TTA chain bypasses this by collapsing onto press (Section A17), which is the right collapse on PhysicsLaw and the wrong one elsewhere.

(D3) β-DP lookahead closes the VirtualHome AS gap. Flat reachability (β-simple) keeps execution rates high but does not improve Task SR: the model selects safe but goal-irrelevant actions. Backward DP (β-DP) shifts per-token logit bias toward goal-reachable continuations and accounts for the bulk of the $21.3\% \rightarrow 85.3\%$ AS jump. When VHBetaStateMachine preconditions are incomplete, a non-monotonic exec $\downarrow$ / goal-rate $\uparrow$ tradeoff appears: β-DP routes toward the goal via paths that fail precondition checks at execution time. The current implementation covers $\approx 80\%$ of VirtualHome predicates; the remaining gaps are the direct fix target for the SD residual (Section 5.1, -9.3 pp gap to Claude-3.5 Sonnet on SD).

(D4) DFA misspecification on BEHAVIOR. On BEHAVIOR GI (Table 13), the affordance DFA drops F_1 from 0.252 (baseline) to 0.148 (DFA-only) by assigning near-zero mass to relational predicate tokens (e.g., INSIDE(x,y)) that make up a large share of BEHAVIOR goal specifications. The HMM partially recovers to 0.191 (γ=0.05) by biasing log-probabilities toward edge-goal tokens that remain within the DFA feasible set, but it cannot recover tokens the DFA has fully masked to zero. On BEHAVIOR SD (Table 14), the DFA itself is also misspecified: DFA-only halves Task SR from 0.21 to 0.10, and the HMM then induces item fragmentation ($\sim$90 single-item subgoals vs. one large structured item), raising hallucination and cancelling the argument-error reduction. Both failures are misspecification problems, not failures of the constraint-enforcement mechanism; format-specific DFAs would close them.

(D5) HMM single-skill collapse. **??** traces every Total gain in **??** to a single dimension: Physical Laws. The remaining five dimensions stay at 0.0 across every TTA configuration. Inspection of the predicted skill distribution exposes the failure mode—*single-skill collapse*: R0 emits open_door on $480/480$ prompts; R3 / R5 emit press on $480/480$ (TTBA correctly identifies that Physical Laws prompts want a parameterless press[N], but applies the same prior to every prompt); the five non-PL dimensions all require pick with an entity argument, and the per-instance Baum–Welch update on M=5 unconstrained samples can reweight emissions over skill tokens but cannot recover the binding to specific entity IDs. The constraint-classification path bypasses this by routing directly from the VLM posterior over symbolic action sketches (Section 4.3); see Section A22 for the mechanism analysis and the entity-subspace fix proposed there.

A29 Failure-type partition: jointly exhaustive, pairwise non-subsuming

CLAMP treats *structural*, *perceptual*, and *causal* failures (Section 1) as a design partition: jointly exhaustive over the failure modes we observed, and pairwise non-subsuming so that handling one mode does not handle the others.

Pairwise non-subsumption. A purely syntactic DFA accepts any token sequence whose surface form is well-typed; nothing in $\mathcal{D}_\gamma$ prevents a perfectly formatted call pick(unicorn) when no unicorn entity is visible—perceptual failures slip through. A visual-grounding filter restricts arguments to $\mathcal{E}_{\text{seen}}$ but still admits sequences that violate a precondition (place(X, Y) before X is held) or that exhaust the action budget without reaching the goal—causal failures slip through. Symmetrically, a reachability DP that operates on grounded actions assumes well-formed inputs and cannot prevent a malformed token stream.

Joint exhaustiveness. On a hand-labelled pilot of failed plans drawn from VirtualHome and VLABench, every failure was attributable to at least one of the three modes; we did not observe a fourth category. We do not claim a theorem of universality, but the partition is consequential because each mode aligns with a distinct decode-time recognizer (α_{syn}, α_{vis}, α_{reach}) that composes additively at the logit level (Section 4.2). This separation is what allows CLAMP to enforce the three constraints simultaneously without constructing their product automaton.

A30 Constrained decoding methods: extended discussion

This section expands on the constrained decoding literature surveyed in Section 2.

Grammar- and FSM-based decoding. The dominant family compiles the target structure into

a context-free grammar or finite-state machine and masks any token at each position that would render the partial output irrecoverably invalid. PICARD (Scholak et al., 2021) parses incrementally to enforce SQL grammar constraints on text-to-SQL outputs. Synchromesh (Poesia et al., 2022) pairs target similarity sampling with constrained semantic decoding for code generation. Outlines (Willard and Louf, 2023) compiles regular expressions into FSMs and applies token-level masks at every step. Grammar-constrained decoding (GCD) (Geng et al., 2023) generalizes this to arbitrary context-free grammars without finetuning. SynCode (Ugare et al., 2025) precomputes a DFA mask store from the grammar to amortize per-step token filtering. XGrammar (Dong et al., 2025) reports up to $100\times$ speedup over prior masks by partitioning the vocabulary into context-independent tokens (prechecked offline) and context-dependent tokens (interpreted at runtime), making it the current state of the art for high-throughput structured generation.

Lookahead-based decoding. CTRL-G (Zhang et al., 2024) pairs a deterministic finite automaton hard mask with an HMM lookahead that estimates the log-probability of reaching an accepting suffix from the current decoding state. The combination provides both a string-level structural guarantee and a soft guidance signal toward constraint satisfaction. CTRL-G is the most direct predecessor of CLAMP.

Why these methods do not transfer to embodied planning. All of the above operate at the token-string level: their guarantee covers the produced token sequence, not the action it denotes. They cannot express that an action's arguments must refer to objects visible in the current scene, that the action's preconditions must hold in the current world state, or that the partial plan must remain on a path to the goal under the remaining step budget. These properties live above the token alphabet, and a single-level automaton over the token vocabulary cannot capture them. CLAMP addresses this gap by lifting constrained decoding to the action level: the per-token mask depends on visual grounding (which objects are currently visible) and on a dynamic-programming feasibility check over the symbolic state (whether a partial plan still admits a goal-reaching continuation under the budget).

A31 Discussion: limitations, real-world applicability, and broader impact

A31.1 Limitations.

CLAMP operates strictly at the symbolic-action level: physical safety during low-level motion execution is out of scope and is delegated to a downstream motion planner. The HMM lookahead component is sensitive to vocabulary and training distribution; on domains far from its source distribution it can degrade to a near-uniform prior, and our test-time adaptation pipeline mitigates but does not eliminate this drift, especially when the test stream is short. The hand-written PDDL specifications used to seed δ_β carry an engineering cost in genuinely new environments; the agentic spec source is so far validated mainly on the VLABench PHYSICSLAW subset. Finally, the test-time adaptation has a robustness boundary: small adaptation streams can overfit, the anchor weight λ must be tuned, and EM steps beyond a few iterations drift away from the domain prior.

A31.2 Real-world applicability under partial observability and stochastic execution.

CLAMP guarantees feasibility at the symbolic-action interface, not at the physical-execution interface, which has two consequences for deployment. First, the visibility recognizer is seeded from the initial observation o_0, so tasks that require active perception (e.g., opening a fridge to reveal an unseen apple) sit outside the static guarantee; structurally, however, $\mathcal{E}_{\text{seen}}$ is a monotonically growing set and the recognizer supports incremental updates, so each replan call is itself a constrained decoding over the latest grounded entities, making this an integration question rather than a formal limitation. Second, the symbolic transition $\mathcal{T}$ and the world-state DFA $\mathcal{D}_\beta$ are deterministic; physical-execution failures (slip, collision, mis-grasp) are caught by the closed-loop skill controller and surface as state changes, at which point the next replanning step re-enters CLAMP with the updated $\mathcal{E}_{\text{seen}}$ and s_β. We therefore view CLAMP as a decode-time layer that composes with reactive TAMP (Wang et al., 2024; Yang et al., 2025; Yan et al., 2026) and on-line perception rather than as a substitute for them; the contribution of this work is in-decoder enforcement of the symbolic constraint slot, with motion-level reliability deferred to the controller and replanning loop that CLAMP is designed to plug into.

A31.3 Broader impact.

CLAMP pushes embodied-agent safety from prompt engineering into a formally verifiable component: the constrained decoder is auditable, killable, and does not require fine-tuning the underlying planner, lowering the compute cost of safer deployment. The main risk is dual-edged: a *misspecified* PDDL or affordance specification silently makes plans harder rather than safer, because CLAMP faithfully enforces whatever interface it is given (we observe this on BEHAVIOR goal interpretation, where the affordance DFA is misaligned with the benchmark's relational-predicate output format). We therefore recommend a spec-audit step before deployment in new environments, and view CLAMP as one layer in a defense-in-depth stack alongside safety benchmarks such as SafeMind, SafeAgentBench, and IS-Bench rather than a replacement for them.

A32 AI Assistants in Research or Writing

A32.1 Information about use of AI assistants.

AI assistants (e.g., ChatGPT/Claude) were used to polish the English grammar and refine the writing structure of the manuscript. They were not used to generate research ideas, design experiments, or produce results; all technical content, claims, and conclusions are the authors' own and were verified by the authors.

Algorithm 1 CLAMP per-token constrained decoding

Require: prompt $\mathbf{x}$; frozen VLM; visually restricted recognizers $\mathcal{D}'_\gamma, \mathcal{D}'_\beta = (S_\beta, A_{\text{act}}, \delta'_\beta, s_0^\beta, F_\beta)$; HMM $\theta = (\pi, A, B)$; world-state table $C_\beta[\cdot,\cdot,\cdot]$; parser τ and meta-token map $w : A_{\text{act}} \to \mathcal{W}^*$; budget $K_{\max}$; scalar weight λ

Ensure: token sequence $y_{1:N}$ with $\Pr[y_{1:N} \models \mathcal{C}] = 1$

$q^\gamma \leftarrow q_0^\gamma, \quad s_\beta \leftarrow s_0^\beta, \quad K \leftarrow K_{\max}, \quad t \leftarrow 0$
$\rho[h] \leftarrow \log \pi[h]$ **for all** $h \in \{1, \dots, H\}$
repeat
  $u_n \leftarrow \textsc{VlmForward}(\mathbf{x},\ y_{1:n})$ ▷ raw logits $\in \mathbb{R}^{|V|}$
  for $y \in V \cup \text{STOP}$ **do** ▷ (1) affordance + visual mask
    $\log \gamma_n[y] \leftarrow \begin{cases} 0, & \delta'_\gamma(q^\gamma, y) \neq \bot \text{ and } y \notin \text{VIS} \\ -\infty, & \text{otherwise} \end{cases}$
  end for
  if τ signals an action-block start at position n **then** ▷ (3) goal-reachability factor
    enumerate candidate completions $\mathcal{C}_t \leftarrow \{c : \tau(y_{1:n} \cdot c) \text{ closes a new action } a_t\}$
    for y such that $\exists c \in \mathcal{C}_t$ with $c.\text{tok}[n] = y$ **do**
      $\log \beta_n[y] \leftarrow \text{lse}_{c \in \mathcal{C}_t,\, c.\text{tok}[n]=y}\big(\beta_{\text{fac}}(c)\big)$
    end for
  else
    $\log \beta_n \leftarrow \mathbf{0}$ ▷ no boundary ⇒ no β contribution
  end if
  if $s_\beta \notin F_\beta$ **then** ▷ (4) EOS gate: cannot stop yet
    $\log \gamma_n[y] \leftarrow -\infty$ **for all** $y \in \text{STOP}$
  else if $K = 0$ **then** ▷ budget exhausted: must stop
    $\log \gamma_n[y] \leftarrow -\infty$ **for all** $y \notin \text{STOP}$
  end if
  $p_n \leftarrow \text{softmax}\big(u_n + \log \gamma_n + \lambda \log \beta_n\big)$ ▷ Eq. (3)
  sample $y_n \sim p_n$
  $q^\gamma \leftarrow \delta'_\gamma(q^\gamma, y_n)$ ▷ advance DFA
  if $y_n \notin \text{VIS}$ **then** ▷ (2) advance HMM belief, Eq. (5)
    $\rho[h] \leftarrow \text{lse}_{h'}\big[\rho[h'] + \log A(h', h)\big] + \log B(h, y_n)$ **for all** h
  end if
  if τ closes an action block at position n **then** ▷ commit action
    $a_t \leftarrow$ last action in $\tau(y_{1:n})$
    $s_\beta \leftarrow \delta'_\beta(s_\beta, a_t), \quad K \leftarrow K - 1$
  end if
  if $y_n \in \text{STOP}$ **then break**
  end if
  $t \leftarrow t + 1$
until terminated
return $y_{1:n}$