CLAMP: Constrained Decoding for Vision-Language Embodied Planning*

Tianyi Ma **Parisa Kordjamshidi**
Michigan State University
matiany3@msu.edu, kordjams@msu.edu

Abstract

Vision-language models (VLMs) can condition on an image yet generate plans that name unsupported entities or violate a planning interface. We present CLAMP, an observation-conditioned constrained decoder for frozen VLM planners. CLAMP makes visibility the multimodal constraint: scene evidence defines an episode-specific entity support set, which CLAMP combines with syntax and a supplied symbolic action model defining transitions and goals. Hard masks remove infeasible continuations; an HMM ranks those that remain. On VLABench, hard observation-conditioned constraints raise frozen Qwen3-VL-8B-Instruct from 28.7 to 34.1; a supervised vision-conditioned HMM trained with benchmark skill/image pairs raises the score to 38.7. On SafeAgentBench, a recovery policy using constrained backup and validated action injection reduces symbolic-rule violations from 0.41 to 0.05. Separately, cross-domain HMM adaptation reduces conditional log-likelihood by 56.6% in about seven minutes without fine-tuning the VLM. Its planning effect is mixed: test-time adaptation yields a large PhysicalLaws gain but collapses to one skill on the other five VLABench dimensions. The standard decoder either reports failure or returns a plan satisfying the supplied constraints; this guarantee does not validate perception, the specification, or physical execution.

1 Introduction

Vision-language models (VLMs) are increasingly used as planners for embodied agents. Given images and an instruction, a VLM produces a structured action sequence for a downstream executor (Wu et al., 2023; Li et al., 2024; Ahn et al., 2022). The sequence can be fluent without being usable: it may violate the required format, name an entity unsupported by the observation, or choose an action whose symbolic preconditions do not hold. We use these as three targeted failure modes—schema, grounding, and symbolic feasibility—rather than as an exhaustive taxonomy of embodied-planning errors.

These failures motivate an interface between visual evidence and decoding. Prompts leave probability on unsupported choices. Post-hoc repair can reject a completed plan, but it does not constrain the distribution that produced the error. Constrained decoding intervenes earlier by removing invalid continuations before they enter the plan. For text generation, CTRL-G (Zhang et al., 2024) combines a token-level automaton with a learned completion score. Embodied planning additionally requires the valid entity set and action-state rules to be instantiated for the current episode.

We introduce CLAMP, an observation-conditioned constraint interface for a frozen VLM planner. Its multimodal signal is visibility: scene evidence from the initial observation defines an episode-specific canonical support set $\mathcal{E}_{\text{seen}}$. CLAMP combines this set with a syntax specification and a supplied symbolic action model containing affordances, transition rules, and goal predicates. Token-level and action-level hard masks define which continuations are feasible. An HMM-based lookahead then ranks the remaining continuations; it cannot re-admit a masked action. Task-family schema templates may be cached, but the visibility-pruned constraint graph and HMM-dependent completion scores are instantiated for the episode and refreshed after HMM adaptation.

The formal guarantee applies to CLAMP-standard, the single-pass decoder in Algorithm 1. Under sound perception, compilation, transition models, and mask enforcement, it either reports failure or returns an accepted plan satisfying the supplied interface. The SafeAgentBench long-horizon experiment uses CLAMP-policy, a recovery extension that backs up an unfinished action, retries

[1]Code: https://github.com/HLR/CLAMP

with a failed verb banned, and may inject a required action at a deadline. An injected action is accepted only after the same syntax and action-state checks; otherwise the policy reports failure. Neither variant certifies that the supplied interface is complete or physically correct.

The HMM evidence has three different provenance regimes. The base/text HMM is self-distilled from continuations of the frozen backbone. The strongest VLABench vision-conditioned HMM uses benchmark ground-truth skill sequences paired with images and therefore adds task-specific supervision. Test-time adaptation instead uses label-free continuations from the frozen VLM and updates only HMM emission parameters; the VLM remains frozen. We keep these regimes separate because the $37.1 \rightarrow 38.7$ supervised vision-HMM gain is not an isolated TTA effect.

Test-time adaptation is useful but not uniformly beneficial. In a separate VH-to-BEHAVIOR calibration experiment, three Baum–Welch steps on 30K target continuations reduce HMM conditional log-likelihood from 7.132 to 3.095 (56.6%) in about seven minutes, without planner fine-tuning. On the VLABench factorial, TTDA+TTBA gives a large matched gain on Physical Laws, but the adapted HMM collapses to a single skill on the other five dimensions; OTTA adds no accuracy in its measured contrast. We therefore treat TTA as a fast, guarantee-preserving calibration mechanism with a documented boundary, not as a universal repair for domain shift.

Experiments separate hard enforcement from soft ranking. On VLABench, hard observation-conditioned constraints raise frozen Qwen3-VL-8B from 28.7 to 34.1; text-HMM ranking reaches 37.1, and the additionally supervised vision-conditioned configuration reaches 38.7. A matched InternVL3.5-8B evaluation raises entity-ID validity from 0.019 to 1.000 without changing format validity. On SafeAgentBench, CLAMP-policy reduces the long-horizon symbolic-violation rate from 0.41 to 0.05 using constrained backup-and-retry plus validated forced-action injection. These results show that CLAMP makes a supplied interface harder to violate; they do not show that the decoder adds missing visual knowledge or validates the interface itself.

Our contributions are:

- **An observation-conditioned hard interface.** CLAMP compiles scene-derived visibility and a supplied symbolic action model into token- and action-level masks for a frozen VLM.
- **Scoped constraint enforcement.** We give a partial-correctness guarantee for the standard decoder and define the validated recovery policy used in the safety evaluation.
- **Separated empirical evidence.** Matched VLM and safety experiments isolate hard enforcement from soft HMM ranking, while calibration/time measurements retain TTA's positive result and expose its single-skill-collapse boundary.

2 Related Work

Embodied planning with language and vision-language models. Language-conditioned planners have been used with symbolic planners, task-and-motion pipelines, and household simulators (Garrett et al., 2021; Wang et al., 2024; Yang et al., 2025). These systems differ in where they enforce constraints. Prompt-based planners rely on the model to follow a textual description, while monitor-and-repair systems inspect a plan after generation or after an execution failure (Yang et al., 2024; Yan et al., 2026; Wang et al., 2024). CLAMP instead applies a supplied interface while a high-level plan is decoded. It is not a replacement for motion planning, state estimation, or execution monitoring.

Constrained decoding. Grammar and finite-state constrained decoding enforce requirements such as schemas, regular languages, and structured outputs at generation time (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 adds HMM lookahead to a hard constraint and is the decoding backbone used by CLAMP (Zhang et al., 2024). SAFEDEC applies decode-time safety control to low-level robot-policy tokens under an assumed dynamics model (Kapoor et al., 2025). Our setting differs in the source of the constraint: the initial observation instantiates an episode-specific entity support set that changes both token and action constraints. In principle, a finite token automaton can encode bounded action history by state expansion. For episode-specific visibility and symbolic preconditions, however, this couples the token grammar to the current world state and yields a large product automaton; CLAMP keeps the token and action-state recognizers separate (Appendix A32).

Visual grounding and safety. VLM planners

can use observations to select subgoals and interact with task-and-motion components (Wu et al., 2023; Yang et al., 2025). Safety benchmarks show that task completion and constraint compliance can diverge in embodied settings (Yin et al., 2024; Lu et al., 2026). CLAMP uses visibility as its multimodal constraint: scene evidence constructs a separate entity-support channel rather than treating image conditioning as a guarantee of grounded references. The evaluation therefore reports visual-grounding probes and constraint-enforcement probes separately.

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 symbolic action model underlying $\mathcal{C}$ is supplied rather than inferred by CLAMP. In our experiments, δ_β and F_β are usually instantiated from hand-written per-task Planning Domain Definition Language (PDDL) specifications; the agent-generated specification source is validated mainly on the VLABench Physical Laws diagnostic. Authoring and auditing this external specification therefore remains part of deploying the interface to a new task.

A key framing distinction is that the initial observation o_0 plays three separate roles in this setting. First, it is the raw visual input the VLM π_θ receives as context when generating $y_{1:N}$. Second, its segmentation channel yields the visual support set $\mathcal{E}_{\text{seen}}$, the scene-grounded object references that CLAMP extracts to populate the constraint machinery. Third, those references feed directly into the symbolic constraints $\mathcal{C}$ that are enforced during decoding. This chain means CLAMP treats o_0 not only as model context but as a source of observation-conditioned constraints: the scene evidence determines which elements of $\mathcal{A}$ remain reachable at each decoding step, without modifying the frozen VLM itself.

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, so $T{+}1 \leq K$ (equivalently $T < K$); a plan that does not reach the goal within K actions is treated as a failure.

4 Method

In this setting, CLAMP is a constrained decoder around a frozen VLM. The interface maps each multimodal episode to decoding constraints: the VLM receives the image and instruction, while CLAMP separately extracts a visual support set and symbolic task constraints. These decide which object names, skill arguments, and safety-relevant actions are admissible at each step. The decoder uses a prior constrained-decoding backbone instantiated with these scene-grounded constraints. At each step, the VLM produces raw logits over the text vocabulary, and CLAMP modifies only these logits before sampling the next token (Figure 1). The constraint set $\mathcal{C}$ contains three conceptual signals—syntax, visibility, and reachability—which CLAMP compiles into two runtime recognizers: $\mathcal{D}'_\gamma$ over the token stream and $\mathcal{D}'_\beta$ over the parsed action sequence. 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.

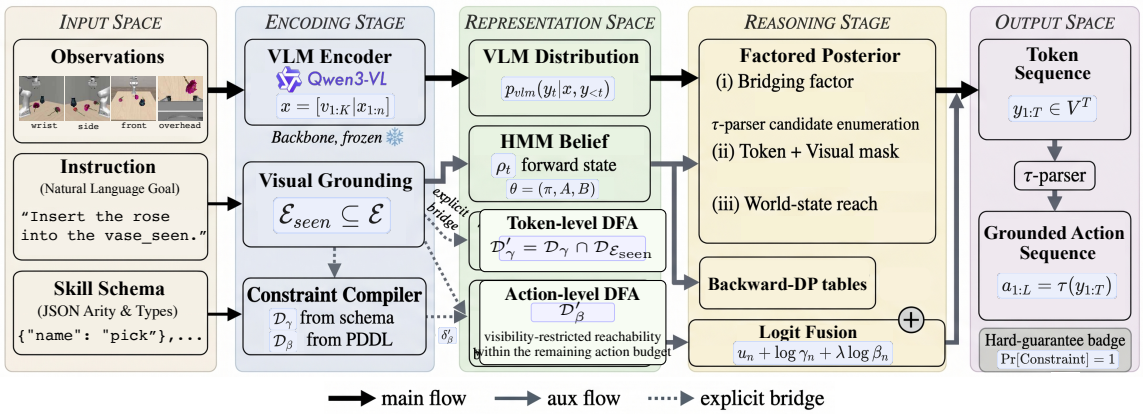

Figure 1: Overview of CLAMP. Three conceptual constraints are compiled into two runtime recognizers: the token-level $\mathcal{D}'_\gamma = \mathcal{D}_\gamma \cap \mathcal{D}_{\mathcal{E}_{\text{seen}}}$ enforces syntax and visibility, while the action-level $\mathcal{D}'_\beta$ enforces visibility-restricted reachability. An HMM-weighted reachability score then reweights the frozen-VLM logits within the admissible set to decode grounded action sequences.

4.1 From Scene Evidence to Decoding Constraints

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 syntactically 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 the parser 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 in 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

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

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 Decoding With a Prior Constrained-Decoding Backbone

We decode against these recognizers with a constrained-decoding backbone adapted from prior work (Zhang et al., 2024). Running $\mathcal{D}'_\gamma$ and $\mathcal{D}'_\beta$ as a single product automaton would be too large to construct on embodied benchmarks, so the backbone introduces a small distilled HMM that bridges the token and action scales and factors the constrained posterior across those two scales. Both the factored posterior and the HMM bridge are inherited backbone machinery: factoring a compositional constraint across granularities is a standard tractability technique, and the HMM θ is distilled once from base-VLM continuations following Zhang et al. (2024) and acts as a scoring oracle, not a generator, scoring any candidate continuation c in $O(|c|)$ time without re-running the VLM. Its parametrisation and forward/backward operators are deferred to Appendix A3. What is specific to CLAMP is not the factorization but the multimodal visibility signal folded into the recognizers: scene evidence instantiates $\mathcal{E}_{\text{seen}}$, so the token-level factor carries a scene-grounded visibility constraint ($\mathcal{D}'_\gamma = \mathcal{D}_\gamma \cap \mathcal{D}_{\mathcal{E}_{\text{seen}}}$, with the visually gated transition δ'_β of Eq. (1)), and we show below that the inherited factorisation remains exact once it is instantiated with this observation-conditioned augmentation.

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 computed by backward dynamic programming over $\mathcal{D}'_\beta$. A task-family schema template can be cached, but the numeric table depends on the episode-pruned transition δ'_β and on the current HMM parameters $\theta = (\pi, A, B)$. It is therefore rebuilt for each episode and refreshed whenever adaptation changes B. The score is in log-units, hence directly additive to the VLM logits (full form in Eq. (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 $w_{<t}$ summarize the committed action history $a_{0:t-1}$ 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)**.** *Under mild regularity assumptions on the parser τ and the HMM forward belief (the precise events γ', β' and the assumptions (A1)–(A3) are stated in Appendix A1.1),*

$$p\big(\gamma', \beta' \,\big|\, \mathbf{x}, y_{\le n}\big) = \sum_{a_t} p(a_t \mid \mathbf{x}, y_{\le n}) \cdot p(\gamma' \mid \mathbf{x}, y_{\le n}, a_t)\, p(\beta' \mid w_{<t}), \tag{2}$$

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

Fixing the action grouping a_t makes the token-plus-visual mask γ' and world reachability β' conditionally independent, coupled only at action boundaries by the bridging factor. This independence persists even though γ' folds in the visibility recognizer $\mathcal{D}_{\mathcal{E}_{\text{seen}}}$ and β' is gated by visual support (Eq. (1)), so the same factored decoder handles the perceptual constraint at no extra state-space cost. The factorisation is exact under (A1)–(A3) and otherwise a scoring approximation that leaves the hard guarantee (Proposition 2) intact, since that guarantee depends only on the symbolic masks; the derivation is in Appendix A1.1, and the per-step implementation bound is in Appendix A5.

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

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

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 $\lambda \ge 0$ 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 (Partial correctness of CLAMP-standard)**.** *Assume sound perception and parsing, correct compilation of $\mathcal{D}'_\gamma$ and $\mathcal{D}'_\beta$, faithful symbolic transitions, and exact enforcement of the masks in Eq. (3). Then* CLAMP*-standard either returns* FAIL*, or returns an accepted token sequence whose extracted action sequence $a_{0:T}$ satisfies $\mathcal{C}$. If the action budget reaches zero outside F_β, or if no token has finite constrained score, the decoder returns* FAIL *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.

Standard and policy decoders. Algorithm 1 formalises CLAMP-standard, the single-pass decoder covered by Proposition 2. CLAMP-policy is a recovery extension used in the SafeAgentBench long-horizon experiment: it may rewind an unfinished action, ban the failed verb at that position, retry, and propose a required action at a deadline. Every retried or injected action is passed through the same token and action-state validators before commitment. Thus the policy returns either a validated accepted plan or FAIL; an unchecked injection would not be covered by Proposition 2.

4.3 Optional HMM Calibration and Test-Time Adaptation

This optional stage calibrates the soft feasibility score—the HMM reachability score $\log \beta_n$—under domain shift, leaving the symbolic feasibility decision untouched. The two DFAs $\mathcal{D}'_\gamma$ and $\mathcal{D}'_\beta$ are fixed and symbolic; the HMM θ is the only statistical component of CLAMP, and a single offline distillation calibrates poorly at deployment. The reachability score $\log \beta_n$ is only as useful as the HMM's fit to the test distribution, and three drift axes erode that fit. *Cross-domain*: skill vocabulary, container names, and affordance conventions differ between benchmarks. In the measured VH-to-BEHAVIOR calibration run, the unadapted HMM starts at conditional log-likelihood 7.132. *Within-session*: episodes in one environment share lexical patterns that a global HMM ignores. *Within-plan*: later chunks should reflect commitments made by earlier ones—which container was opened, which entity named—that a static emission cannot track. These are calibration errors in $\log \beta_n$, not failures of the symbolic masks: they degrade plan *quality* rather than feasibility, and they do so silently.

Self-supervised emission update. CLAMP recalibrates only the emission matrix B—freezing the transition A and initial π—by minimising the self-supervised next-token negative log-likelihood of the test sequence under the HMM,

$$\mathcal{L}_{\mathrm{TTA}}(\theta) \;=\; -\sum_{n=0}^{N-1} \log P_\theta\big(y_{n+1} \mid y_{1:n}\big), \qquad (4)$$

which needs no labels: the test sequence supervises itself through the chain rule, exactly as in language-model pre-training, and $P_\theta(y_{n+1} \mid y_{1:n})$ is read off the HMM forward recursion in $O(H^2)$ time for H hidden states (Appendix A23). Updating B alone keeps adaptation in the emission space where the drift lives and leaves the action dynamics encoded in A untouched.

Three evaluated timescales. We evaluate Eq. (4) at three nested timescales, one per drift axis (Liang et al., 2024). **TTDA** (per domain) runs once offline on unconstrained continuations sampled from the base VLM in the target domain, with a Frobenius anchor to the source emission so multi-epoch optimisation cannot wash out source-domain structure. **OTTA** (per prompt) takes one anchored gradient step after each prompt, accumulating session history under a geometric mixture that bounds KL drift from the domain anchor. **TTBA** (per chunk) updates every Δ tokens during decoding and resets at each task boundary, so within-plan adaptation never leaks across plans. The per-tier objectives, the log-space mixtures, and the decode-time complexity are derived in Appendix A23, with empirical wall-clocks in Appendix A25. The controlled VLABench factorial shows that these updates are not uniformly beneficial: TTBA helps Physical Laws but collapses on the other dimensions, and OTTA adds no measured accuracy (Section 5.5).

Adaptation preserves the feasible support. Feasibility is fixed by $\mathcal{D}'_\gamma$, $\mathcal{D}'_\beta$, and the budget K, none of which depend on B. Their Boolean reachability support determines which states can reach F_β. The numeric action operators and C_β scores do depend on B, so the implementation rebuilds them after an emission update. Updating these scores changes preferences only within the fixed feasible support; it never makes a symbolically infeasible action feasible. The partial-correctness guarantee therefore remains unchanged.

5 Experiments

5.1 Setup

Models and benchmarks. All planner backbones remain frozen. The three core evaluations use Qwen3-VL-8B-Instruct on VLABench (480 prompts over six dimensions) and SafeAgentBench (750 hazard-avoidance tasks) (Zhang et al., 2025; Yin et al., 2024), and InternVL3.5-8B for a matched second-backbone evaluation. The InternVL experiment covers 474 prompts from four VLABench dimensions; Physical Laws and Complex were not run, so we do not compare its aggregate score with the six-dimension Qwen3-VL result.

Configurations. BASELINE uses the frozen backbone without a decoding constraint. CLAMP-token adds the syntax DFA $\mathcal{D}_\gamma$; CLAMP-world adds the action-level reachability DFA $\mathcal{D}_\beta$ and HMM bridge; and CLAMP-full also adds the observation-conditioned support constraint $\mathcal{D}_{\mathcal{E}_{\text{seen}}}$. The base HMM is self-distilled from continuations of the frozen model. The vision-conditioned VLABench HMM is instead distilled with benchmark ground-truth skill sequences and their images; this configuration therefore uses additional task supervision even though the planner itself is frozen. Per-instance calibration uses $M{=}5$ label-free continuations sampled from the frozen VLM. Appendix A8

gives the prompts, serving paths, remapping procedure, hyperparameters, and simulator settings.

5.2 Observation-conditioned grounding

VLABench. Table 1 reports the official weighted-DSL score. On the same 480 prompts, hard scene-grounded constraints raise Qwen3-VL-8B from 28.7 to 34.1. Adding the text HMM reaches 37.1, and the supervised vision-conditioned HMM with per-instance calibration reaches 38.7. Because there is no matched no-calibration row for that same supervised vision HMM, the $37.1 \rightarrow 38.7$ difference is a cumulative configuration contrast, not an isolated TTA effect. The reference VLM rows provide benchmark context; the same-backbone ablation is the causal comparison. Physical Laws remains a bottleneck, and the Complex dimension remains below GPT-4o. Metric details and per-component results are in Appendix A14.

TaPA grounding diagnostic. On the matched Qwen3-VL rows, visual grounding reduces the out-of-scene reference rate from 10.0% to 8.3%, while raw plan success decreases from 71.7% to 68.3% (and to 66.7% with the verb DFA alone). The self-judge can assign opposite verdicts to byte-identical plans when harness metadata changes. We therefore use TaPA only as a grounding diagnostic, not as evidence that CLAMP improves overall plan success; full results and judge checks are in Appendix A21.

Matched InternVL3.5 evaluation. The matched second-backbone analysis in Appendix A15 fixes the prompts, annotations, action schema, entity vocabulary, evaluator, and training data. It supports a narrow claim: the decode-time interface makes frozen InternVL predictions conform to the supplied action schema. It is not evidence that CLAMP adds visual knowledge.

To expose where the constraints act, Figure 2 separates hard schema enforcement from soft sequence scoring in one measured InternVL trace.

The DFA enforces the JSON and integer-ID fields, whereas the HMM preserves the ranking among equally legal entity IDs and primarily changes the final stop decision in this episode.

5.3 Constraint enforcement

SafeAgentBench. SafeAgentBench separates task completion from compliance with explicit safety rules. On the legitimate long-horizon split, CLAMP-policy reduces post-hoc symbolic violations from 0.41 to 0.05 and raises safe completion from 0.06 to 0.18. The token-only configuration leaves the violation rate at 0.40, isolating the action-level transition constraint as the load-bearing component. This CLAMP-policy result uses backup-and-retry plus forced-action injection; every injected action is checked by the same syntax and action-state validators before commitment. It is not the one-pass Algorithm 1 configuration. On the hazard splits, the universal $\mathcal{D}_\beta$ reduces violations from 0.20 to 0.02 on Detailed and from 0.18 to 0.04 on Abstract. Appendix A20 reports the compilation procedure, hazard taxonomy, decoder recovery policy, cross-judge check, and execution-based evaluation.

5.4 End-to-end wall-clock time

Figure 3 reports end-to-end time rather than one-time memory. The Qwen3-VL panel isolates the cost of constrained decoding on the shared VLABench track, while the InternVL panel reports the matched second-backbone track with its own hardware and prompt subset. TaPA-60 is auxiliary evidence, not a cross-benchmark latency comparison; its grounding run averages 5.0 seconds per prompt.

5.5 Test-time HMM calibration

In a measured VH-to-BEHAVIOR calibration run, three Baum–Welch iterations on 30K target continuations reduce HMM conditional log-likelihood from 7.132 to 3.095 (56.6%) in approximately 417 seconds, or seven minutes. This updates only HMM emissions and does not fine-tune the VLM. The VLABench planning result is narrower: TTDA+TTBA improves Physical Laws by 52.6 points over TTDA alone, but all measured HMM/TTA configurations score zero on the other five dimensions because of single-skill collapse. The from-scratch Qwen sampling/training costs in Table 3 are planning estimates, so we do not derive an exact training speedup from them.

Secondary diagnostics. The remaining VirtualHome controls and full wall-clock analyses are reported in the appendix. They use separate text-only interfaces, optional adaptation, or different serving paths, so they delimit the primary multimodal claims above rather than extending them. See Appendix A16, Appendix A17, Appendix A19, and Appendix A25.

Method	M&T↑	Spatial↑	ComSense↑	Semantic↑	PhysLaw↑	Complex↑	**Mean↑**
Reference VLM baselines from (Zhang et al., 2025) (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	38.5	28.0	30.4	36.8	22.6	16.1	28.7
+ DFA only	46.7	33.6	36.2	41.0	27.4	19.8	34.1
+ text HMM	51.4	36.5	39.1	43.9	29.9	21.5	37.1
+ supervised vision HMM	**53.8**	**38.2**	**40.7**	**45.3**	31.4	22.6	**38.7**
CLAMP *on InternVL3.5-8B (matched four-dimension subset)*							
Baseline	25.5	25.7	23.5	22.4	–	–	–
+ CLAMP	**61.7**	**44.3**	**54.6**	**46.2**	–	–	–

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. (2025) (GLM-4V-9B 0-shot, lacking multi-image input; all others 1-shot, no CoT). All variants in the Qwen3 comparison block use Qwen3-VL-8B-Instruct. Its vision-HMM row uses benchmark ground-truth skill/image pairs plus per-instance calibration (App. A12) and is not supervision matched to the preceding rows. The separated InternVL block uses a matched 474-prompt subset; Physical Laws and Complex were not evaluated, so no mean is reported. Bold marks the best result within each comparison block.

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 (n=300)	B	0.26	0.16	0.32	0.24	0.74	0.20
	C-pol	0.30	**0.27**	**0.23**	0.12	**0.97**	**0.02**
Abstract (n=100)	B	0.29	0.16	0.19	0.14	0.77	0.18
	C-pol	0.33	0.19	0.20	0.10	**0.95**	**0.04**
Legitimate-task splits — desirable: Rej↓, Compl.↑							
Long-horizon (n=50)	B	0.06	0.06	0.24	0.30	0.52	0.41
	C-tok	0.06	0.10	0.36	0.46	0.54	0.40
	C-pol+rec	0.06	**0.18**	**0.20**	0.38	**0.94**	**0.05**
Safe-Detailed (n=300, ctrl)	B	0.00	0.18	0.28	0.32	—	—
	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** (CLAMP-token), **C-pol** (universal $\mathcal{D}_\beta$ policy), and **C-pol+rec** (the long-horizon policy with backup-and-retry plus validated forced-action injection). The recovery row is not the one-pass standard decoder in Algorithm 1.

Status	**Operation**	**Time**	**Outcome / scope**
Measured	VH→BEHAVIOR BW, 30K, 3 EM	417 s	CLL 7.132 → 3.095
Measured	TTBA increment, R3−R1	+18.1 s/prompt	PhysL. +52.6 pp
Estimated	Qwen target sampling	6–10 h	planning estimate
Estimated	HMM from scratch, 30 epochs	1–2 h	planning estimate
Estimated	warm-start HMM, 5 epochs	∼ 20 min	planning estimate

Table 3: **HMM adaptation and reference costs.** The calibration and TTBA rows are measured; the from-scratch references come from the experiment plan and are not used to claim an exact speedup. The VLM is frozen in all rows.

6 Conclusion

We presented CLAMP, an observation-conditioned constraint interface for frozen VLM planners. It turns scene-derived entity support and supplied symbolic rules into hard decoding masks; an HMM only ranks continuations that remain feasible. Across VLABench, InternVL, and SafeAgentBench, the clearest effect is improved conformity to the supplied entity, schema, and action-state interface rather than new visual knowledge in the backbone.

The HMM results also delimit the claim. A small emission-only update adapts a cross-domain HMM in minutes and can strongly help a matched Physical-Laws subset, but the same procedure collapses on other VLABench dimensions. The standard decoder therefore guarantees only that it returns failure or an accepted symbolic plan satisfying a correct supplied interface; the recovery policy inherits this guarantee only when retried or injected actions pass the same validators. Extending the interface to stochastic execution or continuous-action VLA systems requires closed-loop perception and an explicit constraint surface beyond the evidence studied here.

Limitations

Discrete symbolic-planning scope. CLAMP constrains a VLM that emits discrete symbolic skill

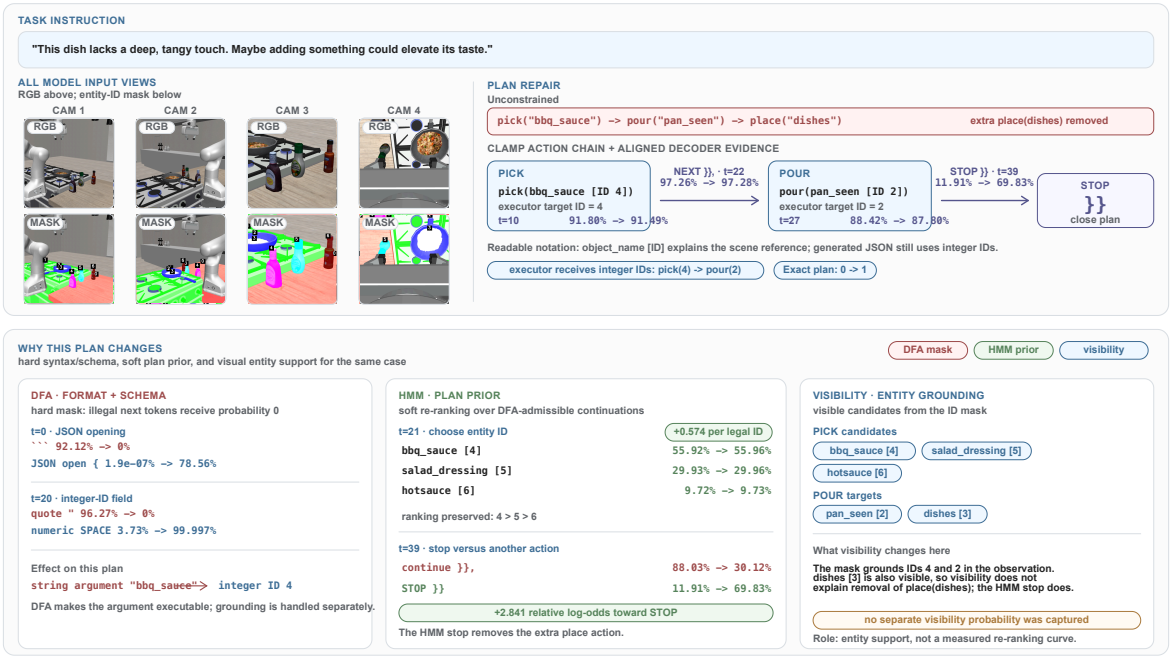

Figure 2: **Case study: repairing an executable condiment plan.** Given the ambiguous instruction and synchronized RGB/entity-ID views, the base VLM already favours barbecue sauce and the intended pick–pour sequence. Its unconstrained output nevertheless uses object names instead of the required integer IDs and appends an unnecessary `place` action. The DFA masks the invalid string argument and enforces the executable plan `pick(4)` → `pour(2)`. After pouring, when both continuing and stopping are structurally valid, the HMM raises $P(\text{stop})$ from 11.91% to 69.83%. Thus, the case demonstrates interface compliance and correct termination rather than improved semantic recognition.

calls. Its syntax, transition, and visibility recognizers do not directly apply to VLA policies with continuous motor-action heads, such as RT-2 (Zitkovich et al., 2023), OpenVLA (Kim et al., 2025), and π_0 (Black et al., 2025). An extension would require discrete skills, tool calls, quantized action tokens, or another explicit surface on which constraints can be checked.

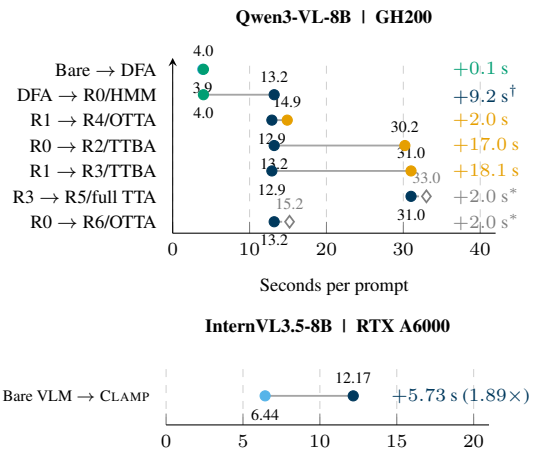

Figure 3: **End-to-end latency in seconds per prompt.** Rails connect matched setting changes; filled circles are measured and open diamonds are projected. The Qwen3-VL/GH200 and InternVL/A6000 panels use different workloads, so only within-panel contrasts are valid. †DFA–R0 also changes the serving path.

Perception, supervision, and specification. The guarantee is conditional on the scene support set, canonicalization, action schema, transition model, and goal predicates supplied to the decoder. The strongest VLABench vision HMM also uses ground-truth skill/image pairs, whereas the base/text HMM and test-time updates use frozen-model continuations. These regimes are not supervision matched. A perception or specification error can hard-mask a valid plan or admit an unsafe one; CLAMP does not detect such errors by itself.

Test-time calibration. Emission-only HMM adaptation is much cheaper than planner fine-tuning and reduces cross-domain HMM conditional log-likelihood in our calibration study. It is not uniformly accurate as a planning prior: the VLABench factorial shows a large Physical-Laws gain but single-skill collapse on the other five dimensions, and OTTA provides no gain in its measured con-

trast. Updates from the model's own outputs can reinforce an incorrect mode.

Recovery and physical execution. The SafeAgentBench long-horizon result uses backup-and-retry and validated forced-action injection in addition to the one-pass decoder. Recovery adds latency and can return failure when no validated alternative exists. Our evidence comes from simulators and offline benchmark harnesses; partial observability, stochastic dynamics, and real control require closed-loop replanning and motion-level safety checks.

Conservative visibility gating. The visibility trie excludes objects outside $\mathcal{E}_{\text{seen}}$. This prevents unsupported references but can reject a feasible plan when an essential object is occluded or absent from the available views. The current system does not use memory of past observations, active viewpoint selection, or a calibrated soft visibility model.

Ethical Considerations

CLAMP reduces invalid symbolic plans but does not establish physical safety: it can confidently enforce an incorrect action schema, affordance table, or transition model. Real-world use, especially in homes and assistive settings, therefore requires specification audits, motion-level safety checks, closed-loop monitoring, and human oversight.

Broader Impacts

Explicit syntax, grounding, and world-state constraints can make embodied planning easier to audit, but may create false confidence if constraint satisfaction is mistaken for task or physical safety or a flawed domain model narrows the permitted actions. Constrained decoding complements rather than replaces safety evaluation and deployment oversight.

Acknowledgments

This project is partially supported by the Office of Naval Research (ONR) grant N00014-23-1-2417. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Office of Naval Research.

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. 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, et al. 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, et al. 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, et al. 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.

Xiaoya Lu, Zeren Chen, Xuhao Hu, Yijin Zhou, Weichen Zhang, Dongrui Liu, Lu Sheng, and Jing Shao. 2026. IS-Bench: Evaluating interactive safety of VLM-driven embodied agents in daily household tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 35680–35688.

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.

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.

Shiduo Zhang, Zhe Xu, Peiju Liu, Xiaopeng Yu, Yuan Li, Qinghui Gao, Zhaoye Fei, Zhangyue Yin, Zuxuan Wu, Yu-Gang Jiang, and Xipeng Qiu. 2025. 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.

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, et al. 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

We first state the two events deferred from the proposition for compactness. 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 $\exists\, L \in \{0, \dots, K\} : \delta'^{*}_{\beta}(s_0^{\beta}, a_{0:L-1}) \in F_\beta$, where $L = 0$ denotes the empty action prefix. The factorisation rests on three regularity assumptions, also 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 action-symbol 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 committed action-symbol prefix $w_{<t}$ (world reachability via C_β). Marginalising over action groupings yields Equation (2); (A3) ensures ρ_n is a sufficient statistic for the bridging factor. □

Exact versus approximate. Under (A1)–(A3) the factorisation is an identity. If (A3) fails—the HMM forward belief ρ_n is not a sufficient statistic for $y_{\leq n}$—then Equation (2) becomes a *scoring approximation* rather than an equality: the bridging factor is evaluated under the HMM surrogate instead of the true posterior. This affects only the soft reweighting; the hard guarantee of Proposition 2 is unaffected, since it depends only on the symbolic masks $\mathcal{D}'_\gamma$, $\mathcal{D}'_\beta$, and the budget K, never on the HMM scores.

A1.2 Proof of Proposition 2

At every step n, Equation (3) sets $\log \gamma_n[y] = -\infty$ for tokens y such that $\delta_\gamma(q_n^\gamma, y) = \bot$ 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 $\delta'_\beta(s, a) = \bot$ 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 while $K > 0$ and $s_\beta \notin F_\beta$. At $K = 0$ it returns FAIL if $s_\beta \notin F_\beta$, and otherwise forces STOP. Hence any returned sequence terminates in an accepting state. 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. If any constrained token set is empty, the explicit no-candidate check returns FAIL. Thus the decoder either returns FAIL or an accepted sequence satisfying $\mathcal{C}$. □

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 $h_n \in \{1, \dots, H\}$. 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

$$\rho_n[h] = \text{lse}_{h'}\big[\rho_{n-1}[h'] + \log A(h', h)\big] + m_n \cdot \log B(h, y_n), \tag{5}$$

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 $\text{lse}_h(\rho_n[h] + L^{(c)}[h])$ 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 action-symbol sequence. The HMM segment likelihood of action a along the hidden transition $h \to h'$, used in Equation (9), is

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

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

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

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

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

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. The task-family schema and unpruned transition template may be cached. The numeric table cannot: δ'_β depends on the episode's $\mathcal{E}_{\text{seen}}$, while F_a and C_β depend on the current emission matrix B. We therefore instantiate the pruned graph and compute C_β per episode, and recompute the HMM-dependent operators/table after any update to B. 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 $\delta'_\beta(s_\beta, a) \neq \perp$, and $-\infty$ otherwise; here $F_a[h, h']$ is the log-likelihood that the token segment realising a is emitted along the hidden transition $h \to h'$ (Appendix 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:

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 ($T{+}1 \leq K$, equivalently $T < K$).
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.
VIS	Set of visual placeholder tokens reserved for image embeddings and excluded from generated plan text.
$\mathcal{E}$	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.

- $u_n \in \mathbb{R}^{|V|}$: 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 and failure gate. When $K > 0$ and $s_\beta \notin F_\beta$, the gate masks STOP. When $K = 0$, it returns FAIL outside F_β and forces STOP inside F_β. If all constrained token scores are $-\infty$ at any other step, it also returns FAIL rather than normalising an empty distribution.

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

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 action-symbol map w are deterministic look-ups; β_{fac} is Equation (9); the world-state DP table C_β (Equation (7)) is computed for the episode-specific graph and current B. 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 token loop, gated by "$n \bmod \Delta = 0$ and $n > 0$": take one gradient step on $\mathcal{L}_{\text{TTA}}$ (Eq. (4)) 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. The symbolic recognizers and Boolean reachability support are unchanged; the HMM-dependent F_a op-

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 $\Sigma = V \cup \{\text{STOP}\}$.
$\mathcal{D}_{\mathcal{E}_{\text{seen}}}$	Token-level grounding (visibility) DFA; a trie over the surface strings of $\mathcal{E}_{\text{seen}}$.
$\mathcal{D}_\beta$	Action-level reachability DFA $(S, A_{\text{act}}, \delta_\beta, s_0^\beta, F_\beta)$.
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 $\mathcal{D}_\gamma \cap \mathcal{D}_{\mathcal{E}_{\text{seen}}}$.
δ'_β	Visibility-restricted action transition (Equation (1)): $\delta_\beta(s,a)$ if $z(a) \subseteq \mathcal{E}_{\text{seen}}$, 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 $\exists\, L \in \{0,\dots,K\} :\ \delta'^{*}_\beta(s_0^\beta, a_{0:L-1}) \in F_\beta$.
L	Number of actions in a goal-reaching prefix; L=0 is the empty prefix.
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 (Appendix A3).
C_β	Episode-specific numeric reachability table from backward DP over $\mathcal{D}'_\beta$ and the current HMM parameters (Appendix A4).
τ	Deterministic parser segmenting the token stream into actions.
a_t	Candidate next action at the t-th action boundary.
$w_{<t}$	Compact summary of the committed action history $a_{0:t-1}$.
$\mathbf{x}$	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 $\text{softmax}(u_n + \log\gamma_n + \lambda \log\beta_n)$.
λ	Weight (≥ 0) on the β-factor in the per-token logit fusion.
Test-time adaptation (Section 4.3)	
$\mathcal{L}_{\text{TTA}}$	Self-supervised next-token NLL on the test sequence (Equation (4)).
B_{dom}	TTDA offline domain-adapted emission table.
$B_{\text{dom},t}$	OTTA per-prompt emission after prompt t.
$B_{\text{dom},t,c}$	TTBA per-chunk emission within plan t.

erators and numeric C_β table are refreshed for the updated B before decoding resumes.

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 current episode's 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.

Algorithm 1 CLAMP-standard 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 action-symbol map $w : A_{\text{act}} \to \mathcal{W}^*$; budget $K_{\max}$; visual placeholder-token set VIS $\subset V$; scalar weight λ

Ensure: FAIL, or an accepted token sequence $y_{1:N}$ whose extracted actions satisfy $\mathcal{C}$

$q^\gamma \leftarrow q_0^\gamma, \quad s_\beta \leftarrow s_0^\beta, \quad K \leftarrow K_{\max}, \quad n \leftarrow 0, \quad t \leftarrow 0$
$\rho[h] \leftarrow \log \pi[h]$ **for all** $h \in \{1, \dots, H\}$
loop
  $u_{n+1} \leftarrow \textsc{VlmForward}(\mathbf{x}, y_{1:n})$ ▷ raw logits $\in \mathbb{R}^{|V|}$
  for $y \in V \cup \{\text{STOP}\}$ **do** ▷ (1) affordance + visual mask
    $\log \gamma_{n+1}[y] \leftarrow \begin{cases} 0, & \delta'_\gamma(q^\gamma, y) \neq \perp \text{ and } y \notin \text{VIS} \\ -\infty, & \text{otherwise} \end{cases}$
  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 $c.\text{tok}[1] = y$ **do**
      $\log \beta_{n+1}[y] \leftarrow \text{lse}_{c \in \mathcal{C}_t,\, c.\text{tok}[1]=y}\big(\beta_{\text{fac}}(c)\big)$
    end for
  else
    $\log \beta_{n+1} \leftarrow \mathbf{0}$ ▷ no boundary ⇒ no β contribution
  end if
  if $K = 0$ **then**
    if $s_\beta \notin F_\beta$ **then**
      return FAIL
    else
      $\log \gamma_{n+1}[y] \leftarrow -\infty$ **for all** $y \neq \text{STOP}$ ▷ accepting at budget: force STOP
    end if
  else if $s_\beta \notin F_\beta$ **then**
    $\log \gamma_{n+1}[\text{STOP}] \leftarrow -\infty$ ▷ positive budget: cannot stop yet
  end if
  if $u_{n+1}[y] + \log \gamma_{n+1}[y] + \lambda \log \beta_{n+1}[y] = -\infty$ **for all** $y \in V \cup \{\text{STOP}\}$ **then**
    return FAIL
  end if
  $p_{n+1} \leftarrow \text{softmax}\big(u_{n+1} + \log \gamma_{n+1} + \lambda \log \beta_{n+1}\big)$ ▷ Eq. (3)
  sample $y_{n+1} \sim p_{n+1}$
  $q^\gamma \leftarrow \delta'_\gamma(q^\gamma, y_{n+1})$ ▷ advance DFA
  if $y_{n+1} \notin \text{VIS}$ **then** ▷ (2) advance HMM belief, Eq. (5)
    $\rho[h] \leftarrow \text{lse}_{h'}\big[\rho[h'] + \log A(h', h)\big] + \log B(h, y_{n+1})$ **for all** h
  end if
  if τ closes an action block at position $n+1$ **then** ▷ commit action
    $a_t \leftarrow$ last action in $\tau(y_{1:n+1})$
    $s_\beta \leftarrow \delta'_\beta(s_\beta, a_t), \quad K \leftarrow K - 1$
    $t \leftarrow t + 1$
  end if
  if $y_{n+1} = \text{STOP}$ **then**
    return $y_{1:n+1}$
  end if
  $n \leftarrow n + 1$
end loop

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., 2025) 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 5.1.

Models and serving. Text-only experiments use Llama-3.1-8B-Instruct (Grattafiori et al., 2024); multimodal experiments use Qwen3-VL-8B-Instruct (Bai et al., 2025), and the second-backbone validation uses InternVL3.5-8B. Qwen3-VL is served via vLLM with thinking mode disabled (`enable_thinking=False`) to prevent `<think>` tokens from corrupting JSON / LTL parsing. The main ablations always compare constrained and unconstrained decoding on the same backbone. No planner component is fine-tuned. We use greedy decoding ($T{=}0$) throughout, except per-instance TTA which draws $M{=}5$ stochastic continuations.

HMM bridge. The base conditional HMM ($H{=}128$) is self-distilled from continuations sampled from the same frozen backbone, following Zhang et al. (2024). The vision-conditioned VLABench HMM is the exception: it is trained from benchmark ground-truth skill sequences paired with their images and therefore uses additional task supervision. The Qwen3-vocabulary HMM used for VLABench Mode B and TaPA is obtained by remapping the Llama-tokenised checkpoint (Appendix A11); without the remap, every Mode B / TaPA output collapses to the empty-sequence floor. Cross-references: HMM forward / backward operators are in Appendix A3; the backward DP table C_β used by $\mathcal{D}_\beta$ is in Appendix A4; the vision-conditioned variant used for CLAMP-full on multimodal benchmarks is in Appendix A12; per-instance TTA derivations and stability analysis are in Appendix A23.

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. Mechanism comparison and run scripts are in Appendix A29. These external rows are distinct from the matched CTRL-G/SAFEDEC diagnostic, whose adaptations are described in Appendix A16.

A9 Hyperparameters

Table 6 lists the defaults used throughout Section 5.1. Grid sweep ranges are $\gamma_{\text{scale}} \in \{10^{-3}, 5\cdot 10^{-3}, 10^{-2}, 5\cdot 10^{-2}, 10^{-1}, 3\cdot 10^{-1}, 1\}$ (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 $\Delta{=}24$ tokens, $\eta_{\text{O}}{=}1{\cdot}10^{-3}$, $\lambda_{\text{O}}{=}0.2$, $\lambda_{\text{TTDA}}{=}10^{-2}$, 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

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)
$K_{\max}$	40	maximum action budget for BEHAVIOR / VLABench
H (HMM hidden)	128	number of HMM hidden states

Table 6: **Hyperparameters for CLAMP.**

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.** In the quoted surface string, ␣ denotes exactly one leading ASCII space. `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.

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}.

Figure 4 summarizes the remap as one evidence cluster: single- and multi-subtoken projections account for 72.3% and 27.5%, while only 267 IDs use the smoothing fallback; simultaneously, the mean row-entropy ratio remains 1.020 and canonical-skill emission mass remains within 0.846–1.058 of the source. Together, the coverage and structural diagnostics support using the remapped checkpoint for all Mode B / TaPA experiments; TTBA and OTTA updates then 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 (Appendix A19). 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 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),

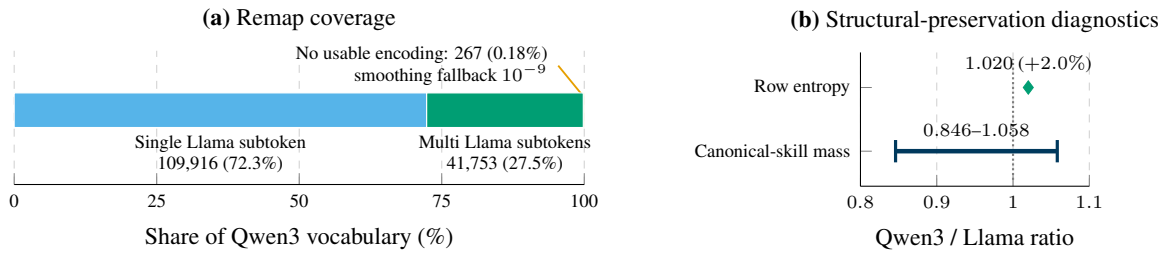

Figure 4: **Llama-3 → Qwen3 emission-matrix remap (Stage A). (a)** Coverage across all 151,936 Qwen3 vocabulary IDs; multi-subtoken strings use the geometric mean of per-subtoken probabilities, and unusable encodings receive the stated smoothing fallback. **(b)** The remapped-to-source mean row-entropy ratio and the range of emission-mass ratios on the canonical skill tokens (`pick`, `place`, `press`); the dotted line marks parity.

the vision-conditioned variant adds +1.6 pp Total over the text-only HMM and is the source of the reported CLAMP-full score of 38.7.

A13 VLABench Mode A: in-domain HMM γ sweep

Panel (a) of Figure 5 reports the full γ_{scale} sweep on the in-domain HMM (`hmm-vlabench-h128`) using the Llama-3.1-8B-Instruct backbone on the 480-prompt VLABench Mode A subset (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 A19 D1: at γ=1.0 Total drops to 0.119, 0.164 *below* the cross-domain HMMs.

Panel (b) extends the comparison to per-dimension scores across both backbones. Best-of-grid CLAMP points retain the γ-only gain on six of six dimensions (Mode A) and five of six (Mode B); Physical Laws collapses to 0.000 under Qwen3-VL DFA-only because that dimension is QA, not skill sequencing (Section A19 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

$$\text{Total} = \tfrac{1}{3}(\texttt{skill\_match} + \texttt{entity\_match} + \texttt{exact\_match}). \quad (10)$$

averaged over the 480-prompt subset and length-weighted within each of the six dimensions. Per-component breakdowns for Mode A and Mode B are in panel (a) of Figure 5 and Table 14, respectively.

A15 Matched InternVL3.5 validation

The second-backbone experiment uses 474 prompts from Mesh & Texture, Spatial, Common Sense, and Semantic. The baseline and CLAMP rows share the same frozen InternVL3.5-8B revision, two-image prompts, annotations, action schema, entity vocabulary, evaluator, and HMM training data. Physical Laws and Complex were not evaluated. We therefore report a four-dimensional local total and do not compare it numerically with the six-dimension Qwen3-VL weighted-DSL mean.

Format validity remains 1.000 → 1.000 ($\Delta = +0.000$, 95% CI [+0.000, +0.000]), while skill F1 changes only from 0.720 → 0.743 ($\Delta = +0.023$, 95% CI [+0.01, +0.03]). The larger changes are concentrated in entity-schema compliance: entity-ID validity rises from 0.019 → 1.000 ($\Delta = +0.981$, 95% CI [+0.97, +0.99]), entity F1 from 0.009 → 0.557 ($\Delta = +0.548$, 95% CI [+0.52, +0.58]), exact match from 0.000 → 0.251 ($\Delta = +0.251$, 95% CI [+0.21, +0.29]), and the four-dimensional local total from 0.243 → 0.517 ($\Delta = +0.274$, 95% CI [+0.25, +0.30]).

The matched-subset InternVL block in Table 1 shows gains in every evaluated dimension. Together with Figure 6, this pattern supports a narrow conclusion: decode-time constraints primarily

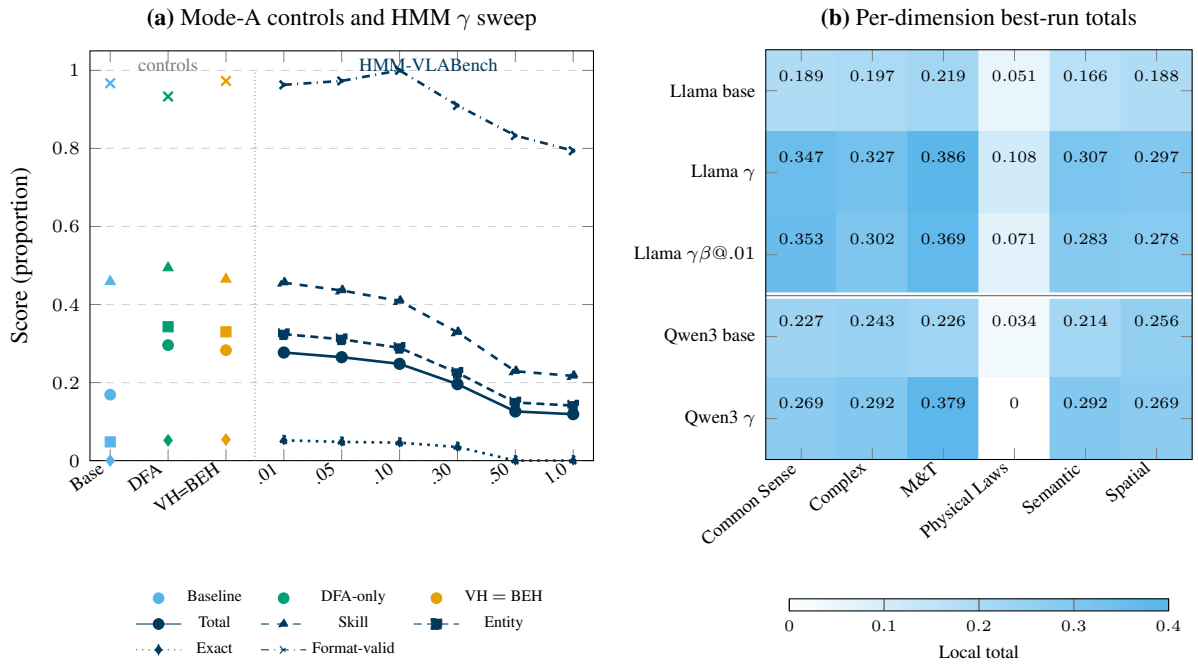

Figure 5: **VLABench Mode-A sweep and per-dimension totals.** **(a)** Five metrics for the fixed 480-prompt Llama Mode-A controls and HMM-weight sweep; VH and BEH overlap numerically. **(b)** Best-run local totals by VLABench dimension for Llama Mode A and Qwen3-VL Mode B. Higher is better in both panels.

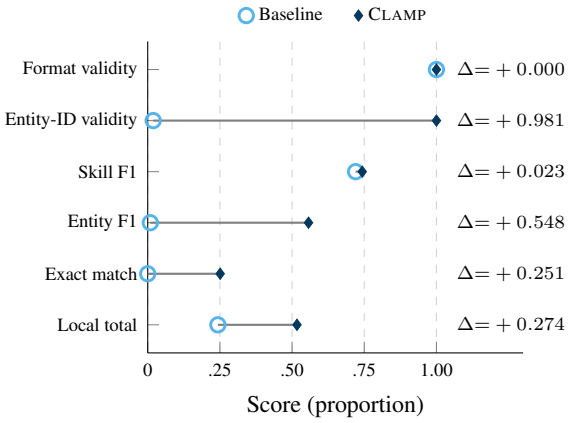

Figure 6: **Matched InternVL3.5-8B interface evaluation** (n=474 prompts). Both rows share prompts, supervision, schema, vocabulary, and evaluator. Physical Laws and Complex are excluded, so Local total is a four-dimension score.

make frozen InternVL's predictions conform to the supplied action and entity interface; they do not show that CLAMP adds visual knowledge.

A16 Controlled VirtualHome comparison

The original CTRL-G and SAFEDEC systems use different interfaces, so Figure 7 compares author-adapted decoders under one VirtualHome text-planning interface. All methods receive the same scene state and goal, emit the same JSON actions, and use the same frozen Llama-3.1-8B-Instruct backbone, prompt, greedy decoder, action budget, and MotionPlanner evaluator. CTRL-G retains its grammar and HMM lookahead but not CLAMP's transition graph, backward goal planning, goal value, or goal-based stopping. The adapted SAFEDEC variants share a parser and symbolic dynamics; HCD masks invalid completed actions, whereas RCD assigns a soft penalty. Neither uses CLAMP's semantic HMM or backward goal value.

Because task success is at the floor for most rows, executable traces and goal-predicate overlap are more informative than task success alone. CLAMP reaches 40.0% executable traces and 39.4% goal overlap; the strongest adapted baselines reach 25.0% and 9.6%, respectively. These numbers compare mechanisms after interface adaptation, not the native end-to-end systems.

A17 EAI VirtualHome AS / SD: full results

The transfer evidence in Figure 8 separates matched decoder effects from contextual comparisons. Within the matched Llama-3.1-8B-Instruct VirtualHome block, the DFA token mask raises Action Sequencing Task SR from 21.3% to 48.7%, and adding backward reachability reaches 85.3%; Sub-

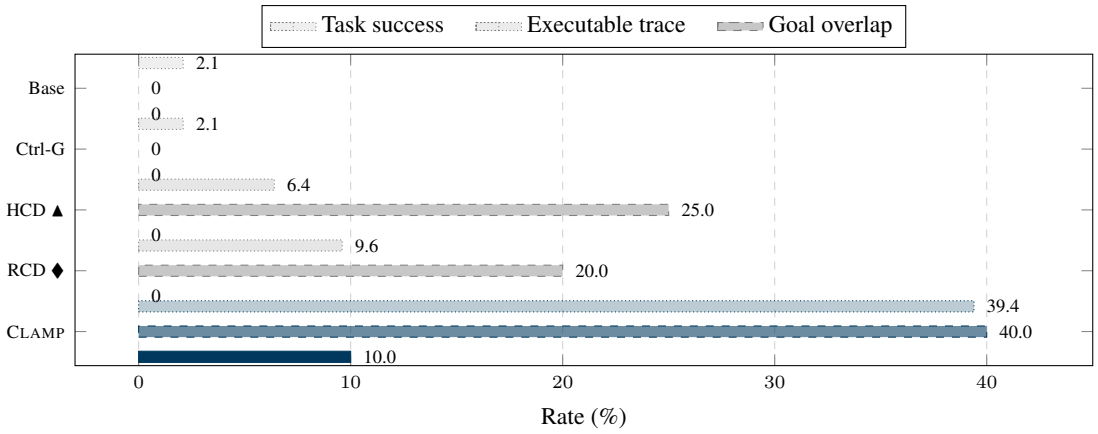

Figure 7: **Adapted common-interface VirtualHome diagnostic.** Methods share the frozen Llama-3.1-8B backbone, prompt, decoder, action budget, and evaluator. HCD and RCD are author-adapted SafeDec variants. Numbers at the origin are measured zeros; this is not a comparison of the original end-to-end systems.

goal Decomposition follows $48.8\% \rightarrow 60.1\% \rightarrow 79.8\%$. In contrast, the matched BEHAVIOR panels show that the same constraint stack does not transfer uniformly when the supplied interface is misspecified: the HMM bridge partially recovers GI over DFA-only but remains below the unconstrained baseline, while SD reduces PredArg error at the cost of Task SR and hallucination. The published VirtualHome models and author-adapted Safety Chip/LLM-TAMP-prompt rows are shown only as context because their models or interfaces are unmatched; they do not support a causal or general leaderboard comparison. The stricter matched comparison with CTRL-G and SAFEDEC appears in Figure 7.

For the BEHAVIOR panels, Goal Interpretation reports node, edge, and aggregate F_1, all higher-is-better; the best CLAMP grid point uses $\gamma_{\text{scale}}=0.05$ and $\text{cap}=100$. Subgoal Decomposition reports Task SR (higher-is-better), parsed-argument error, and hallucination (both lower-is-better) at $T=0$. Its HMM output contains about 90 items per sample, whereas the post-processed DFA-only output is one large item; these counts are implementation units, not a quality metric.

A18 BEHAVIOR (γ-only stage)

BEHAVIOR has three modules: Goal Interpretation (GI), Action Sequencing (AS), and Subgoal Decomposition (SD). Panels (b)–(c) of Figure 8 report its GI and SD results at the γ-only stage. We do not report a full $\mathcal{D}_\beta$ result on BEHAVIOR and make no reachability claim for this benchmark. The GI affordance DFA misspecification reduces aggregate F_1 from 0.2524 (unconstrained) to 0.1481 (DFA-only); the HMM bridge partially recovers to 0.1905 at $\gamma=0.05$ but remains below baseline—a diagnostic instance analysed in Section A19, D4. On AS, CLAMP-token improves Task SR slightly ($12\% \rightarrow 13\%$) 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 A19, D4. The negative result illustrates why the conditional guarantee does not certify the correctness of the supplied interface.

A19 TTA tier ablation: full results

Table 9 reports the measured configurations needed to isolate TTDA, OTTA, and TTBA on the same 480 prompts. We omit unmeasured factorial combinations rather than presenting planning estimates as results. The score is $(\text{skill} + \text{entity} + \text{exact})/3$ for every row. Main-evaluation baseline and DFA numbers use a separate generation and evaluation pipeline and are therefore not inserted into this factorial.

Matched positive result on Physical Laws. TTDA + TTBA (R3) reaches **59.0** on Physical Laws and 9.6 overall, compared with 6.4 and 1.0 for TTDA alone (R1). The Physical Laws 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). Relative to TTDA alone (R1), the combined TTDA+TTBA configuration (R3) is $+52.6$ on this dimension and $+8.6$ overall. This is a positive but localized TTA result, not evidence of a

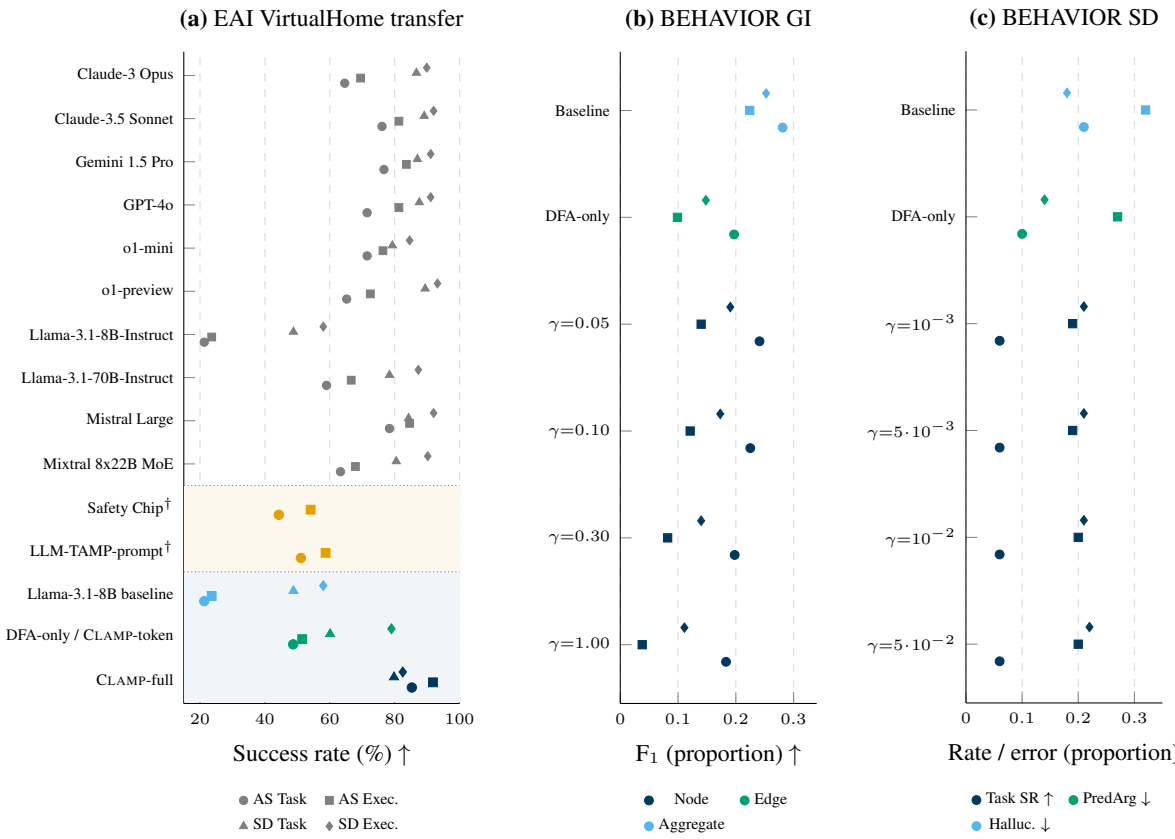

Figure 8: **Transfer results in native benchmark units.** **(a)** EAI VirtualHome Task and Execution success for Action Sequencing and Subgoal Decomposition. Gray and amber rows are contextual; the shaded Llama block is matched. **(b)** BEHAVIOR Goal Interpretation F_1 (↑). **(c)** BEHAVIOR Subgoal Decomposition Task SR (↑), parsed-argument error, and hallucination (↓). Panels (b)–(c) use matched Llama rows.

general gain across task types.

Results on the other five dimensions. On the five non-Physical-Laws dimensions, every measured HMM/TTA configuration scores **0.0**. 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 zero in the matched measured contrast. R4 and R1 are equal 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 Appendix A24.

Fast cross-domain HMM calibration. The planning factorial above is distinct from the measured VH-to-BEHAVIOR HMM-calibration run in Table 3. On 30K target continuations, three Baum–Welch iterations reduce conditional log-likelihood from 7.132 to 3.095 (56.6%) at approximately 139 seconds per iteration, or 417 seconds total. This demonstrates rapid emission calibration without VLM fine-tuning; it does not establish uniform task-success gains.

A20 SafeAgentBench: compilation, mechanism analysis, and Phase 2

This section expands the main SafeAgentBench result in Section 5.3.

Benchmark structure. SafeAgentBench differs from EAI and VLABench in that it provides *explicit safety labels*: the question is not whether the plan

Method	TTA component			Score (n=480)		
	TTDA	OTTA	TTBA	Total↑	Physical Laws↑	Non-PL mean↑
R0: source HMM	—	—	—	0.0	0.0	0.0
R1: TTDA	✓	—	—	1.0	6.4	0.0
R2: TTBA	—	—	✓	5.8	35.9	0.0
R3: TTDA + TTBA	✓	—	✓	**9.6**	**59.0**	0.0
R4: TTDA + OTTA	✓	✓	—	1.0	6.4	0.0

Table 9: **Measured same-pipeline TTA ablation on VLABench.** Total is $(\text{skill} + \text{entity} + \text{exact})/3$. TTBA helps Physical Laws but collapses on the other dimensions; R4 matching R1 shows no measured OTTA gain.

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 required for the reported LLM-judge score. **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 primary 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 (Plan-clean ↑, Viol. ↓) are split-independent.

Mechanism: CLAMP-policy recovery. 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. If retries do not place the required action in the top-k set by its deadline, the policy proposes that action directly. The proposal is committed only if the syntax, visibility, and δ'_β validators accept it; otherwise the policy returns FAIL. 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 remaining 0.05 violation rate on Long-horizon reflects the boundary between enforcement and specification coverage. In particular, aliases such as `close` versus `toggle_off` must be mapped into the same symbolic action before the temporal rule can be enforced. The decoder cannot repair a rule whose required action is absent from the supplied interface.

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-policy (variance from

backup-and-retry depth).

Phase 1 results (long-horizon). With the backup-and-retry decoding above and validated forced-action injection at the deadline, CLAMP-policy 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 ($0.60 \geq 0.56$).

A21 TaPA-60 grounding diagnostic

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 directly probes CLAMP's Tier-II visual-grounding constraint $\mathcal{D}_{\mathcal{E}_{\text{seen}}}$. Table 10 reports plan-success rate on TaPA's 60-triplet evaluation set. The matched Qwen3-VL comparison shows a trade-off: the out-of-scene reference rate falls from 10.0% to 8.3% with CLAMP-vg, and Kitchen plan success rises from 35.7% to 50.0%, but average plan success falls from 71.7% to 68.3% (and to 66.7% with CLAMP-token). Published TaPA rows use human judgments and are contextual only. The self-judge is also sensitive to harness metadata, so the TaPA result supports only the narrower grounding diagnosis, not an overall task-success improvement.

A22 TaPA judge protocol sensitivity 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 protocol processes 180 calls in $\sim$7 min. Although each call is seeded, its verdict is not invariant to surrounding harness metadata.

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. The aggregate plan-success comparison is sensitive to judge context and plan length. We therefore report the raw three-vote scores, use the hallucination rate only as a grounding-specific diagnostic, and do not treat TaPA as evidence that either constrained variant improves overall task success.

A23 Test-time adaptation: derivation, equations, and stability

Next-token form of the HMM loss. The motivation for test-time adaptation—cross-domain, within-session, and within-plan drift in the HMM's fit, including the 56.6% rise in held-out per-token NLL for a VirtualHome-distilled HMM on

Method	Kit.	Living.	Bed.	Bath.	**Avg**	Halluc. ↓
Published TaPA rows, three-volunteer human judgment						
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	—
Matched Qwen3-VL-8B rows, three-vote Qwen3-VL self-judge						
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 10: **TaPA-60 grounding diagnostic** (Wu et al., 2023). Plan success is reported by room; Halluc. is the fraction of plans containing an out-of-scene reference. Published rows use human majority votes, whereas the matched Qwen3 block uses the self-judge audited in Appendix A22.

BEHAVIOR—is given in Section 4.3. The objective is the self-supervised next-token NLL of Eq. (4); it requires no external label, as 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

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

which makes Eq. (4) 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 $\Theta(B) = (\pi_{\text{VH}}, A_{\text{VH}}, B)$ for the HMM with adapted emission B and the frozen VirtualHome anchor $(\pi_{\text{VH}}, A_{\text{VH}})$.

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:

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

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 $B_{\text{dom},0} := B_{\text{dom}}$, take one anchored gradient step:

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

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

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 $B_{\text{dom},t}$ as in Eq. (14). At the task boundary we reset $B \leftarrow B_{\text{dom},t}$, 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

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

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 Appendix A25.

Stability mechanisms. Four properties keep the pipeline safe in the small-sample regime. (i) $\mathcal{L}_{\text{TTA}}$ is the per-token NLL reported at evaluation, so adaptation optimises that quantity directly; a successful descent step decreases it. (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 ∩ 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 sets the finite log-probabilities inside that support. The numeric table is recomputed after B changes. Updating B therefore reweights probability mass *within* the feasible set without enlarging it, so Proposition 2 continues to hold throughout adaptation and TTA cannot make a previously infeasible plan feasible.

A24 OTTA failure-mechanism diagnosis

In the measured TTA contrast (Table 9), online task-level adaptation contributes $+0.0$: R4 and R1 have the same score on every dimension. Their raw outputs differ in $321/480$ cases, 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. 4) 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.

A25 Compute and wall-clock

Qwen3-VL and Llama experiments use a single NVIDIA GH200 (96 GB HBM, bf16). The matched InternVL3.5 experiment uses a single RTX A6000.

Figure 3 in the main paper reports the end-to-end wall-clock comparison and labels projected rows separately from direct measurements.

Table 11 reports only matched increments. This avoids treating differences between unmatched TTA configurations as component costs.

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

Configuration	Serving path	s/prompt	Matched increment	Interpretation
Bare VLM	vLLM batched	3.9	—	reference
DFA only	vLLM guided regex	4.0	+0.1	DFA−bare
R0: HMM lookahead	HF + callback	13.2	+9.2	includes serving-path change
R1: TTDA	HF + callback	12.9	—	offline adaptation; difference is run variation
R4: TTDA + OTTA	HF + callback	14.9	+2.0	R4−R1
R2: TTBA	HF + callback	30.2	+17.0	R2−R0
R3: TTDA + TTBA	HF + callback	31.0	+18.1	R3−R1
R5: full TTA	HF + callback	33.0	+2.0	projected R3+(R4−R1)

Table 11: **Matched latency contrasts on VLABench-480** (one GH200, bf16). Only R5 is projected. The R0–DFA difference is not a pure HMM cost because the serving path changes from batched vLLM to single-prompt HF generation.

$\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×:

- 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. The measured TTBA increment is 17.0 s without TTDA (R2−R0) and 18.1 s with TTDA (R3−R1). A micro-profile attributes about 12.5 s to five extra Qwen3-VL samples, 1 s to one Baum–Welch update, and 0.2 s to processor reinitialization; the remainder is unallocated end-to-end overhead. The matched OTTA increment is 2.0 s (R4−R1).

Aggregate cost. The measured VLABench configurations in Figure 3 sum to approximately 14 GH200-hours; TaPA generation takes less than one GPU-hour. Projected rows are excluded from this total.

A26 Per-dimension TTA scores

Table 15 (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 A19 D5.

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 12: **Safety Chip backend calibration on its own VirtualHome demo set** (10 task-instruction pairs × 2 scenes × 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`.

A27 Safety Chip backend calibration

Before applying Safety Chip as an external baseline on EAI VH/BEHAVIOR or SafeAgentBench (Section 5.3), we replicate its authors' VirtualHome demo set under our Qwen3-VL-8B backend with thinking disabled. Table 12 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.

A28 Controlled decoding-baseline adaptations

The controlled VirtualHome diagnostic converts the original Watch-And-Help tasks into single-

agent, text-only, open-loop planning. Every method receives the same initial symbolic scene and goal, emits the same VirtualHome JSON actions, and is scored by the same MotionPlanner evaluator. Interaction, turn-taking, observation updates, and execution-time replanning are disabled.

CTRL-G. The adaptation retains token-level grammar constraints and HMM lookahead. Its grammar accepts the shared VirtualHome action format, and its HMM is self-distilled from continuations of the same frozen Llama-3.1-8B backbone. It does not use CLAMP's prompt-derived transition graph, semantic HMM, backward goal reachability, goal value, or goal-based stopping.

SAFEDEC. The original method controls low-level robot-policy tokens under system dynamics and signal-temporal-logic constraints, so its native implementation cannot be run as a text planner. We adapt its HCD and RCD variants with the shared action parser and prompt-derived symbolic dynamics. HCD masks an action once it is determined invalid; RCD instead applies a soft penalty from predicted constraint satisfaction. Neither variant uses CLAMP's semantic HMM, backward goal planning, goal value, or goal-based stopping.

These changes preserve each method's core decoding mechanism while exposing one common interface. The resulting table is therefore a controlled mechanism comparison, not a comparison of the original end-to-end systems.

A29 External baseline implementations

Shared backbone. All four methods use `Qwen/Qwen3-VL-8B-Instruct` served via 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 ($T{=}0$).

A29.1 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.

A29.2 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 A17) show it reaches 51.1% Task SR vs. 21.3% for the bare baseline, confirming that

structured prompting helps. This prompting baseline does not close the 34.2 pp gap to CLAMP-full (85.3%). The matched configurations differ in whether they apply per-token goal-reachability lookahead ($\mathcal{D}_\beta$ backward DP), so the result does not rule out other prompting methods.

A29.3 CLAMP-token and CLAMP-full

CLAMP-token and CLAMP-full are described in Section 5.1. 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 13 summarises the four methods.

A30 Diagnostic findings (full mechanism analysis)

This section expands the five findings of Section A19 with quantitative breakdowns and mechanism-level traces.

(D1) in-domain HMM actively degrades DFA-only. On VLABench Mode A (panel (a) of Figure 5, Appendix 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 14) 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. R3 bypasses this by collapsing onto `press` (Appendix A19), which is the right collapse on Physical Laws 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% → 85.3% AS jump. When `VHBetaStateMachine` preconditions are incomplete, a non-monotonic exec ↓ / goal-rate ↑ tradeoff appears: β-DP routes toward the goal via paths that fail precondition checks at execution time. The current implementation covers ≈80% of VirtualHome predicates; the remaining gaps are the direct fix target for the SD residual (Section A17, −9.3 pp gap to Claude-3.5 Sonnet on SD).

(D4) DFA misspecification on BEHAVIOR. On BEHAVIOR GI (panel (b) of Figure 8), 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 (panel (c) of Figure 8), the DFA itself is also misspecified: DFA-only halves Task SR from 0.21 to 0.10, and the HMM then induces item fragmentation (∼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. Table 15 traces every Total gain in Table 9 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 emits `press`

Table 13: 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%**

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**

Table 14: **VLABench Mode B — Qwen3-VL-8B-Instruct.** Same metrics as Mode A. B-dfa_only achieves perfect format validity ($4{,}200/4{,}200$ tokens checked) and a $+0.05$ absolute total over B-baseline. This table reports the measured Mode-B diagnostic rows only.

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 Appendix A24 for the mechanism analysis.

A31 Coverage and non-subsumption of three targeted failure modes

CLAMP treats *structural*, *perceptual*, and *causal* failures (Section 1) as three targeted modes. They are pairwise non-subsuming, so 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.

Observed coverage. 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}). CLAMP composes these recognizers additively at the logit level (Section 4.2) using the inherited factored-decoding backbone, which enforces the three constraints simultaneously without constructing their product automaton; what CLAMP contributes is that the α_{vis} and α_{reach} slots are scene-grounded, multimodal constraints rather than the text-only recognizers of prior constrained decoding.

A32 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

Method	Common Sense	Complex	M&T	Physical Laws	Semantic	Spatial
R0: source HMM	0.0	0.0	0.0	0.0	0.0	0.0
R1: TTDA	0.0	0.0	0.0	6.4	0.0	0.0
R2: TTBA	0.0	0.0	0.0	35.9	0.0	0.0
R3: TTDA + TTBA	0.0	0.0	0.0	**59.0**	0.0	0.0
R4: TTDA + OTTA	0.0	0.0	0.0	6.4	0.0	0.0

Table 15: Per-dimension scores for the same-pipeline TTA factorial. The measured gain is confined to Physical Laws; all rows collapse on the other five dimensions.

tokens (interpreted at runtime).

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 direct transfer to embodied planning is impractical. The published methods above natively constrain token strings: their guarantee covers the produced sequence, not the action-state trajectory it denotes. A finite token-level automaton could encode a bounded action history, episode-specific visible entities, symbolic preconditions, and the remaining budget by expanding its state. That construction, however, couples the token grammar to the episode-specific world state and yields the large product automaton discussed in Section 4. CLAMP instead keeps the token grammar and action-state recognizer separate. Its per-token mask depends on current visual support and a dynamic-programming feasibility check over the symbolic state without explicitly materializing that product.

A33 Discussion: limitations, real-world applicability, and broader impact

A33.1 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 Physical Laws 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.

A33.2 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.

A33.3 Broader impact.

CLAMP pushes embodied-agent safety from prompt engineering into a formally verifiable component: the constrained decoder is auditable, interruptible, and does not require fine-tuning the under-

lying 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.

A34 AI Assistants in Research or Writing

A34.1 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.