# TEMPORAL SPECIFICATION GUIDANCE FOR TEXT-TO-VIDEO GENERATION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Text-to-video models often generate the visual concepts requested in natural language prompts but fail to satisfy their intended ordering, timing, and persistence requirements. We propose representing these requirements as specifications in a formal Temporal Logic, and introduce **Temporal Specification Guidance (TSG)**, a training-free inference-time framework that turns temporal specifications into executable objectives for guiding video generation during sampling. TSG scores traces of visual concept evidence using differentiable quantitative semantics and uses the score's gradients to construct a specification-guided denoising direction. We instantiate this framework with video–text attention traces, efficient differentiable scores for Metric Temporal Logic, and gradient-derived attention biases. This instantiation requires neither an external concept classifier nor backpropagation through the video transformer. Its attention-steered forward pass yields a denoising direction that augments classifier-free guidance (CFG). On a benchmark of 220 prompts spanning 11 categories of temporal constraints, including ordered object entry and state transitions, this instantiation improves specification satisfaction accuracy over CFG by 9.5–20 percentage points across five models, with minimal memory overhead. These results demonstrate the potential of temporal specifications and differentiable satisfaction objectives as a flexible interface for controlling video generation.

## 1 INTRODUCTION

Text-to-Video generation has seen tremendous progress in recent years, with models such as Gemini Omni, Pika, Runway, Wan (Wan et al., 2025), and CogVideoX (Yang et al., 2025) producing videos with improved visual quality and semantic fidelity. Classifier-Free Guidance (Ho & Salimans, 2022) was introduced to improve adherence to natural language prompts in image generation and is now widely adopted by video generation models. However, adherence to temporal constraints remains challenging (Choi et al., 2025; Schiber et al., 2026). Figure 1 shows two examples of such failures with CogVideoX-5B. While the model broadly incorporates the semantic concepts described in the text prompts (e.g., the presence of a traffic light or a drone), it fails to follow the required ordering, persistence, and timing requirements. In these examples, the model misses the intended timed color progression and the eventual appearance of a hiker or a flare signal, respectively.

In this work, we propose expressing temporal requirements as specifications in formal Temporal Logics. Formal temporal specifications can precisely express requirements that are implicitly described in natural language prompts. Frame-level visual concepts, such as the presence of a traffic light or the associated colors, can be represented as atomic propositions. Temporal Logics offer standard Boolean operators (negation, conjunction, and disjunction) and temporal operators such as Always (**G**), Until (**U**), and Eventually (**F**) to express persistence and ordering requirements. In this work, we present a specification language adopted from Metric Temporal Logic (Koymans, 1990) to express temporal requirements with different degrees of timing precision. Representations of temporal requirements from prior work, including timed binary masks (Schiber et al., 2026) and untimed Temporal Logic formulae (Choi et al., 2025), can also be written as specifications in this language.

We propose **Temporal Specification Guidance** (TSG), a training-free inference-time framework for improving adherence to temporal specifications with transformer-based video generation models. Our key insight is that temporal specifications provide an executable objective for guiding video



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.




Figure 1: TSG improves adherence to temporal requirements. **Top:** CFG produces an amber light for the first two-thirds while TSG follows the expected yellow–red–green progression. **Bottom:** CFG omits both requested rescue cues; TSG generates a flame signal while retaining the drone. Videos are generated with CogVideoX-5B.

generation during sampling. TSG makes this practical through differentiable specification scores and attention interventions that avoid backpropagation through the video transformer. Concretely, TSG computes traces of visual concept evidence using video-text attention from the video transformer. For every attention head, TSG computes biases as gradients of a differentiable specification score, steers the attention scores with these biases, and computes the specification-guided direction from the steered forward pass through the video transformer. Our use of attention as a proxy for visual concept presence builds on prior work on attention-based semantic guidance in image and video generation (Chefer et al., 2023; Schiber et al., 2026). The use of video-text attention mitigates the need for an external concept classifier during generation. Because differentiation is restricted to the attention-derived traces and their specification scores, TSG avoids backpropagation through the video transformer, reducing the associated computational and memory demands.

We compute a differentiable specification score over attention traces using a quantitative semantics for Metric Temporal Logic. Inspired by efficient log-domain quantitative semantics for Linear Temporal Logic (Khare et al., 2025), our quantitative semantics aggregates support for the temporal witnesses and alternatives admitted by a specification. Its log-domain form provides differentiable guidance and can be efficiently evaluated in time linear in trace length for a fixed specification. We establish monotonicity in independent literal-support inputs and predictable concept-score gradient signs under consistent polarity.

We evaluate TSG on a benchmark of 220 prompts spanning temporal constraints from 11 categories, including timed state transitions, entity appearance and disappearance, persistence, and choice. Across five models from the CogVideoX (Yang et al., 2025) and Wan (Wan et al., 2025) families, TSG improves specification satisfaction accuracy by 9.5–20 percentage points over CFG-only sampling. We also evaluate TSG with Wan 2.1 1.3B on strictly-timed prompts from the TempoControl object appearance benchmark (Schiber et al., 2026), where it outperforms CFG sampling by up to 22 percentage points in temporal accuracy and matches the state-of-the-art method on that benchmark with approximately 62% lower peak GPU memory. Together, these findings support specification-guided attention steering as an effective and memory efficient approach to temporal control in video generation.

## 2 TEMPORAL CONSTRAINTS AS FORMAL SPECIFICATIONS

Natural Language prompts to video generation models discuss the visual concepts that must be present in the video, along with the order and optionally the timing of their appearance. In this section, we discuss a formal specification language for these constraints.

Let $\mathcal{M}$ represent a Text-to-Video (T2V) generation model that takes a text prompt $p$ as input and outputs a video $v = [v_1, \ldots, v_T]$ with $T$ discrete frames. Let $C$ denote the set of all frame-level visual concepts of interest. Examples of such visual concepts include entity presence (e.g., traffic light), entity states (e.g., the color red), and settings with properties (e.g., sky at dusk). We use $y(v_i, c) \in \{0, 1\}$ to denote the presence or absence of the visual concept $c$ in frame $v_i$.

**Specification language for temporal constraints.** We propose a fragment of Metric Temporal Logic over finite traces ($MTL_f$) as a suitable language for specifying temporal constraints over visual concepts. A specification $\varphi$ in $MTL_f$ takes the following form:

$$\varphi \coloneqq \top \mid \bot \mid c \mid \neg c \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \mathbf{X}_I \varphi \mid \varphi_1 \mathbf{U}_I \varphi_2 \mid \mathbf{G}_I \varphi$$

where $\top$ and $\bot$ represent *True* and *False*, respectively, and $c$ denotes an atomic proposition corresponding to a visual concept $c$. The Boolean operators $\neg$, $\wedge$, and $\vee$ denote negation, conjunction, and disjunction, respectively. The temporal operators $\mathbf{X}_I$, $\mathbf{U}_I$ and $\mathbf{G}_I$ denote *Next*, *Until*, and *Always*, in interval $I \subseteq [0, 1]$, resp. The temporal operator *Eventually* $\mathbf{F}_I$ can be derived as $\mathbf{F}_I \varphi = \top \mathbf{U}_I \varphi$.

This language can be used to specify several interesting constraints on persistence, order, and timing of visual concepts. Time intervals $I = [t_1, t_2]$ allow us to express specifications with varying timing precision: untimed ($t_1 = 0, t_2 = 1$), interval-timed ($t_1 \geq 0, t_2 \leq 1$), and strictly-timed ($0 \leq t_1 = t_2 \leq 1$). Note that the intervals $I = [t_1, t_2]$ are written over time normalized in $[0, 1]$ and the corresponding offsets can be computed as $[\lceil t_1(T-1) \rceil, \lfloor t_2(T-1) \rfloor]$. The constraint from Figure 1 that requires the drone to remain visible *throughout the video* and at least one of the two rescue cues to appear *at some point* in the second half can be written in this language as:

$$\varphi_{\text{rescue}} = \mathbf{G}_{[0,1]}\mathsf{drone} \wedge \mathbf{F}_{[0.5,1]}(\mathsf{flame_signal} \vee \mathsf{stranded_hiker})$$

**Representing temporal constraints from prior work.** This specification language can be used to express temporal constraints from recent works, including untimed Temporal Logic (TL) specifications (Choi et al., 2025) and binary temporal masks that prescribe the presence or absence of a visual concept in each frame (Schiber et al., 2026). Untimed TL specifications can be written with the trivial $[0, 1]$ intervals. A temporal mask can be encoded as a conjunction of *Always* constraints over intervals where the mask is constant, using $c$ for presence and $\neg c$ for absence. Additionally, we can write specifications that correspond to a set of temporal masks, including the choice between visual concepts and the disappearance of an object within an interval. In other words, a temporal mask corresponds to a given valid trace while our specifications can admit several valid traces.

**Deciding whether a video satisfies a specification.** These specifications can be supplied directly with the text prompt, or can be extracted from the prompt using a Large Language Model (LLM). Given the frame-level labels $y(v_i, c)$ for visual concepts, the label $y(v, \varphi, t)$ for a video $v$ satisfying specification $\varphi$ starting from frame $1 \leq t \leq T$ can be computed deterministically using the standard Boolean semantics for $MTL_f$. We present this semantics, along with examples of other specifications, in Appendix B. The examples in Figure 1 thus score $y(v, \varphi, 1) = 0$ with CFG-only sampling, which we discuss next, and $y(v, \varphi, 1) = 1$ with TSG, discussed in Section 4.

## 3 VIDEO GENERATION AND CLASSIFIER FREE GUIDANCE

We focus on latent text-to-video models that generate videos from text prompts through iterative sampling in a compressed latent space, using a transformer to predict a denoising direction at each step. This formulation encompasses model families such as CogVideoX (Yang et al., 2025) and Wan (Wan et al., 2025). The sampling process takes place in a compressed spatio-temporal latent space. A video $v \in \mathbb{R}^{T \times H \times W \times 3}$ corresponds to a latent $z \in \mathbb{R}^{T_z \times H_z \times W_z \times C_z}$, where $T_z$ and $(H_z, W_z)$ denote the latent temporal and spatial resolutions respectively, and $C_z$ denotes the number of latent channels. A 3D Variational Autoencoder (VAE) is trained to decode the video $v = D(z)$.

Given a text prompt $p$, a text encoder produces a sequence of tokens $e_p$. Sampling begins with a Gaussian latent $z_N \sim \mathcal{N}(0, I)$ and proceeds through $N$ steps. At step $k$, the Transformer $\mathcal{T}_\theta$ predicts the update direction $d_k$ as:

$$d_k = T_\theta(z_k, e_p, k), \qquad z_{k-1} = \mathrm{Step}_k(z_k, d_k).$$

We refer to $d_k$ as the update direction or *denoising direction* interchangeably. $\mathrm{Step}_k$ incorporates the sampling rule and update schedule, and can vary across models. The latent generated from the final step ($z_0$) is then de-compressed by the VAE decoder $D$ to the output video $v = D(z_0)$.

**Classifier-free Guidance.** Conditioning on the text prompt alone may not provide sufficient control over the content of the generated video. Classifier-free guidance (CFG) (Ho & Salimans, 2022) was proposed to strengthen this conditioning during sampling, without requiring an external classifier. At each sampling step, CFG combines two update directions using the transformer:

$$d_{k,u} = T_\theta(z_k, e_{\mathrm{ref}}, k), \qquad d_{k,c} = T_\theta(z_k, e_p, k), \qquad d_{k,\mathrm{cfg}} = d_{k,u} + \lambda_{\mathrm{cfg}}(d_{k,c} - d_{k,u}).$$

Here, $e_{\mathrm{ref}}$ represents the embedding of an empty prompt or a negative prompt. The update direction for step $k$, $d_{k,\mathrm{cfg}}$ is computed using the predictions from the unconditional or reference branch $d_{k,u}$ and the prompt conditional branch $d_{k,c}$. The strength of the extrapolation is controlled by the *CFG guidance scale* $\lambda_{\mathrm{cfg}} \in \mathbb{R}$, with different recommendations across models.

## 4 TSG: Specification Guidance for Text-to-Video Generation

We propose Temporal Specification Guidance (TSG), a training-free sampling-time framework that augments CFG with an update direction $d_{k,\mathrm{tsg}}$ guided by the temporal specification $\varphi$. This framework has three components: temporal traces that provide evidence of visual concepts, a differentiable scoring function that evaluates these traces against the specification $\varphi$, and an update mechanism that uses gradients of this score to steer generation. In this work, we compute traces of visual concept evidence using video-text attention within the transformer $T_\theta$, score them against $\varphi$ with a differentiable quantitative semantics for $MTL_f$, and use the gradients as attention logit biases. This attention-steered forward pass produces the specification-guided direction $d_{k,\varphi}$. We present the overall process in Figure 2 and discuss each step in detail now.

**Temporal Specification-guided sampling.** Let $d_{k,u}$ and $d_{k,c}$ denote the reference and prompt-conditional direction predictions from Section 3. Similarly, let $d_{k,\mathrm{cfg}}$ and $\lambda_{k,\mathrm{cfg}}$ denote the CFG-predicted direction and the CFG guidance scale, respectively. At sampling step $k$, TSG computes a *specification-guided* direction $d_{k,\varphi}$ using $\varphi$ and yields the update direction $d_{k,\mathrm{tsg}}$ as:

$$d_{k,\mathrm{tsg}} = d_{k,\mathrm{cfg}} + \lambda_{k,\mathrm{tsg}}(d_{k,\varphi} - d_{k,c}), \qquad z_{k-1} = \mathrm{Step}_k(z_k, d_{k,\mathrm{tsg}}).$$

As with CFG, the strength of TSG is controlled by the *TSG guidance scale* $\lambda_{k,\mathrm{tsg}} \geq 0$. As the CFG guidance scale $\lambda_{k,\mathrm{cfg}}$ usually takes values recommended by model providers and can dynamically vary across sampling steps, we compute $\lambda_{\mathrm{tsg}}$ using the ratio between the TSG contribution and the $\lambda_{\mathrm{cfg}}$-scaled CFG contribution as:

$$\lambda_{k,\mathrm{tsg}} = \rho_{k,\mathrm{tsg}} \cdot \frac{\overbrace{\|\lambda_{k,\mathrm{cfg}}(d_{k,\mathrm{c}} - d_{k,u})\|_2}^{\text{Scaled CFG contribution}}}{\underbrace{\|d_{k,\varphi} - d_{k,c}\|_2}_{\text{Unscaled TSG contribution}}}. \tag{1}$$

The parameter $\rho_{k,\mathrm{tsg}}$ denotes the relative strength of TSG with respect to CFG. With $\rho_{k,\mathrm{tsg}} = 1$, TSG's contribution is scaled to match that of CFG. We apply TSG for the first $N_{\mathrm{tsg}}$ update steps, with CFG applied during the remaining steps. We next describe how the direction $d_{k,\varphi}$ is computed.

**Attention-derived temporal traces for visual concepts.** To mitigate the need for an external concept classifier and inspired by recent work (Schiber et al., 2026), we use video-text attention to compute temporal traces of visual concepts. The text prompt $p$ and video latent $z$ provide tokens to serve as text keys and video queries, respectively, for every attention head $h$ in layer $\ell$ of transformer $\mathcal{T}_\theta$. We map every concept proposition $c$ in specification $\varphi$ to the set of token indices $P_c$ in prompt $p$. Let $Q_t$ be the set of spatial video queries for latent frame $z_t$ and $A[q, j]$ represent the post-softmax attention

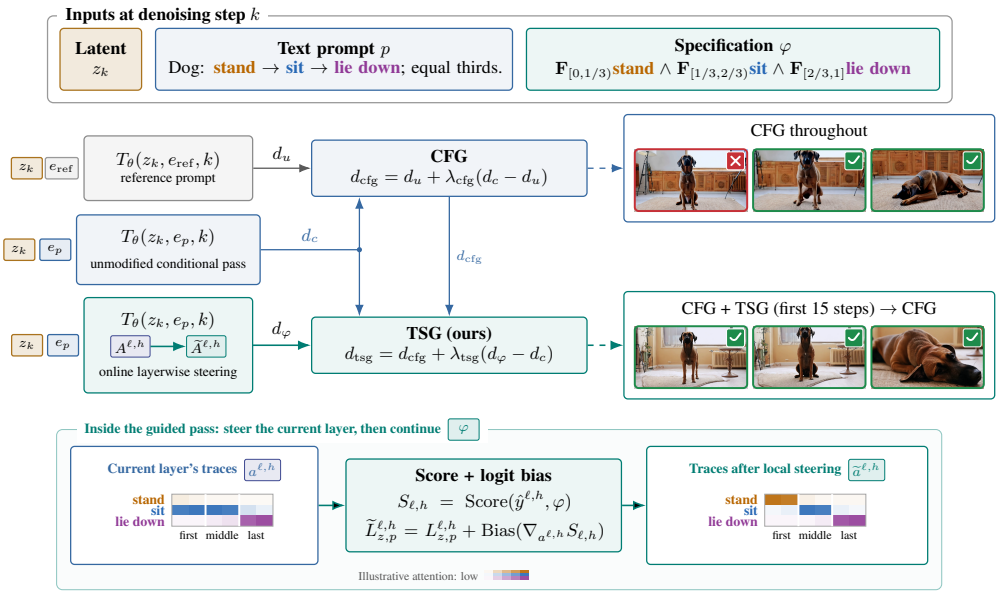


Figure 2: *Temporal Specification Guidance (TSG).* During an additional conditional pass, TSG scores each selected head's current attention traces against the specification and applies gradient-derived logit biases. The resulting prediction augments CFG. Transformer weights are shared and receive no gradients. Traces are illustrative; videos use Wan2.2 5B.

score for video token $q \in Q_t$ and text token index $j$. We construct the raw attention trace for concept $c$ at layer $\ell$ and head $h$, $a_{t,c}$ and the corresponding concept score trace, $\hat{y}(z_t, c)$ as:

$$a_{t,c} = \frac{1}{|\mathcal{Q}_t|} \sum_{q \in \mathcal{Q}_t} \max_{j \in \mathcal{P}_c} A[q, j], \qquad \hat{y}(z_t, c) = \sigma\left(\frac{a_{t,c} - \mu_c}{\tau s_c + \epsilon}\right). \tag{2}$$

The score $a_{t,c}$ corresponds to the maximum attention to any token corresponding to concept $c$, averaged over all spatial positions for latent frame $z_t$. We compute concept presence scores $\hat{y}(z_t, c)$ by scaling attention scores over latent frames using the sigmoid function with temporal mean $\mu_c$ and population standard deviation $s_c$. $\tau > 0$ is a temperature, and $\epsilon > 0$ stabilizes the denominator.

Note that the latent frame $z_t$ does not correspond to video frame $v_t$ because it is defined at the latent temporal resolution $T_z < T$. We limit steering to this latent resolution and convert normalized specification intervals to offsets on this grid using the convention in Section 2, with $T$ replaced by $T_z$.

**Differentiable Specification Scores with LogSTOP.** Inspired by efficient quantitative semantics for Linear Temporal Logic specifications over videos (Khare et al., 2025), we define an efficient interval-aware sum–product quantitative semantics in log space for $MTL_f$ specifications. Given an MTL specification $\varphi$ and concept score traces $\hat{y}(z_t, c)$ at attention layer $\ell$ and head $h$, we can compute the score $\hat{y}(z, \varphi)$ recursively using Algorithm 1 in Appendix E. Intuitively, conjunction and Always multiply support for their requirements, while disjunction, Eventually, and Until sum support for admissible alternatives or temporal witnesses.

The score for any specification $\varphi$ can be computed using $O(|\varphi|T_z)$ arithmetic operations (Proposition E.2). The score is non-decreasing in independent literal-support inputs, as proved in Proposition E.3. This yields predictable gradient signs for concept–time inputs with consistent polarity (Corollary E.4). The direction is determined by the surrounding trace when a concept appears with both polarities. Appendix E provides further details on this monotonicity property, the algorithm's complexity, and examples of the resulting gradients.

**Online attention steering.** For each layer $\ell$, head $h$, and concept $c \in \varphi$, we differentiate $\hat{y}(z, \varphi)$ with respect to the raw trace $a_{t,c}$:

$$g_{\cdot,c} = \nabla_{a_{\cdot,c}} \hat{y}(z_t, \varphi), \qquad b_{\cdot,c} = \eta\, g_{\cdot,c} / \|g_{\cdot,c}\|_\infty, \qquad B[q, j] = \sum_{c:\, j \in \mathcal{P}_c} b_{t(q),c}.$$

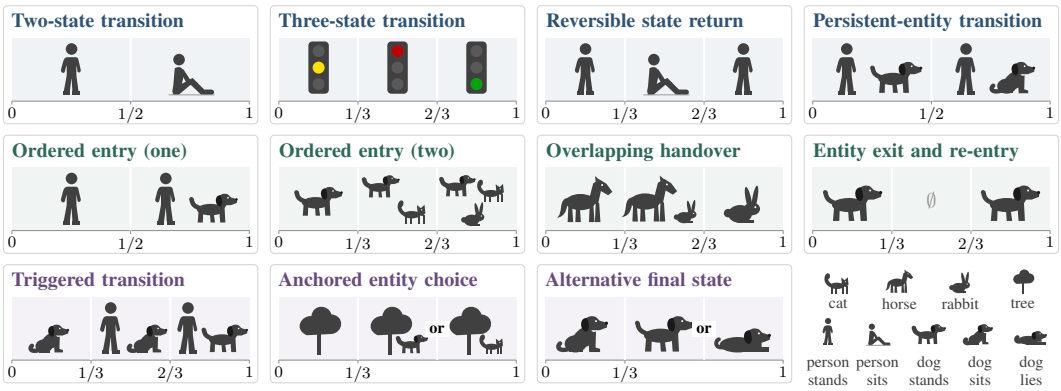


Figure 3: **TSG benchmark with interval-timed temporal requirements from 11 categories.** Shaded intervals indicate permitted event timing within the video duration; $\emptyset$ denotes absence, and "or" denotes mutually exclusive outcomes. Pictograms are illustrative.

where $b_{t,c}$ is the normalized *attention bias* for temporal position $t$ and concept $c$. We set $b_{\cdot,c} = 0$ for negligible gradients. The step size $\eta$ controls the bias magnitude. We broadcast temporal bias $b_{t,c}$ to every spatial query $q \in Q_t$ at position $t(q)$ and every text token $j$ associated with the concept $c$ as $B[q, j]$. For spatial query $q$, we further center the biases by the mean bias over all text keys and clip them in a given range $[b_{\min}, b_{\max}]$.

We apply the resulting bias to the logits for layer $\ell$ and head $h$ as:

$$\widetilde{L}^{\ell,h}_{z,p} = L^{\ell,h}_{z,p} + B^{\ell,h} \tag{3}$$

The biased attention is used to compute the layer's output, and the forward pass continues with the updated representations. Biases are computed anew at each selected layer and sampling step, separately for each head. The completed guided pass yields the specification-guided direction $d_{k,\varphi}$.

## 5 Experimental Evaluation

### 5.1 Setup and Benchmarks

**The TSG Benchmark with interval-timed temporal constraints.** Existing benchmarks primarily focus on untimed constraints, such as state transitions in TC-Bench (Feng et al., 2024) and NeuS-V (Sharan et al., 2025), and strictly-timed constraints such as TempoControl (Schiber et al., 2026). To evaluate TSG on interval-timed constraints, we construct a benchmark with 220 text prompts from 11 categories of temporal constraints. These categories are presented in Figure 3 and include variations in the requested timing, ordering, persistence, and choice among visual concepts.

We use `GPT-5.6-Sol` to propose 50 entities spanning objects from VBench (Huang et al., 2024) and background settings, with associated states (e.g., pose of an object, or time of day for a setting). We construct sets of 20 prompts for each temporal constraint category in Figure 3. We restrict the intervals to halves and thirds within 5–6-second videos, allowing transitions to occur at any point within each interval. Finally, we manually assign temporal specification templates for each category and instruct `GPT-5.6-Sol` to instantiate the specifications with atomic propositions mapped to respective visual concepts in each prompt. Appendix C provides further details on examples of prompts for each category along with the list of temporal specification templates.

**Baselines and Models.** We evaluate text-to-video generation models with CFG sampling as our main baseline. We include five models from two model families: CogVideoX-2B and CogVideoX-5B (Yang et al., 2025), and Wan2.1 T2V-1.3B, Wan2.2 TI2V-5B, and Wan2.2 T2V A14B (Wan et al., 2025). These models cover both joint video–text attention, used by CogVideoX, and dedicated video-to-text cross-attention, used by Wan. We use the CFG settings recommended by the model providers. On the strictly-timed prompts from the TempoControl benchmark, we also compare TSG against the TempoControl method. We use TSG during the first 15 sampling steps with $\rho_{k,\text{tsg}} = 1$. Appendices D.1 and D.2 describe the model settings and TSG configuration; Appendices D.3 and D.5 detail the TempoControl comparison and compute measurements.

Table 1: **TSG improves satisfaction accuracy on a benchmark of 220 prompts.** Latency and Memory measurements are on B200 for all models, except for Wan2.2 A14B which was run on H200.

| Model | Method | Sat Acc. (%) | Img. quality | Latency (min) | Peak GPU mem. |
|---|---|---|---|---|---|
| CogVideoX 2B | Base | 10.5% (+0.0%) | 59.80 (+0.0) | 0.75 ± 0.03 (1.00×) | 26.30 GiB |
| | TSG | 30.5% (+20.0%) | 56.85 (-3.0) | 2.62 ± 0.38 (3.49×) | 26.33 GiB |
| CogVideoX 5B | Base | 8.6% (+0.0%) | 63.72 (+0.0) | 1.82 ± 0.08 (1.00×) | 31.28 GiB |
| | TSG | 26.8% (+18.2%) | 62.18 (-1.5) | 6.06 ± 0.86 (3.33×) | 31.31 GiB |
| Wan 2.1 1.3B | Base | 15.9% (+0.0%) | 69.25 (+0.0) | 1.39 ± 0.02 (1.00×) | 30.48 GiB |
| | TSG | 31.4% (+15.5%) | 67.39 (-1.9) | 2.35 ± 0.17 (1.69×) | 30.52 GiB |
| Wan 2.2 5B | Base | 25.0% (+0.0%) | 70.38 (+0.0) | 2.43 ± 0.03 (1.00×) | 46.83 GiB |
| | TSG | 34.5% (+9.5%) | 69.16 (-1.2) | 4.56 ± 0.41 (1.87×) | 45.15 GiB |
| Wan 2.2 A14B | Base | 35.9% (+0.0%) | 65.42 (+0.0) | 24.28 ± 0.21 (1.00×) | 109.33 GiB |
| | TSG | 46.8% (+10.9%) | 68.63 (+3.2) | 35.18 ± 1.93 (1.45×) | 107.95 GiB |

**Metrics.** We uniformly sample 21 decoded frames from each video, and evaluate the frame-level presence of a visual concept using `Qwen2.5-VL-7B-Instruct`. We use the ground-truth semantics of MTL described in Appendix B to compute the specification satisfaction accuracy (Sat Acc.). We also evaluate visual quality of the generated videos using VBench's MUSIQ-SPAQ imaging quality scorer (Huang et al., 2024), along with the time taken for generation (latency) and the peak GPU memory allocated to the process. Appendix D.6 provides further details.

### 5.2 RQ1. Does TSG improve adherence to temporal specifications?

**Result 1: TSG consistently outperforms CFG-only on interval-timed and strictly-timed temporal constraints.** Table 1 presents the results comparing TSG against CFG-only sampling on the TSG benchmark. TSG improves satisfaction accuracy by 9.5–20 percentage points across models, with Wan 2.2 A14B reporting the highest satisfaction accuracy of 46.8% with TSG. Interestingly, Wan2.1 1.3B reports a 31.4% accuracy with TSG compared to Wan2.2 5B's 25% with CFG-only sampling, despite having fewer parameters. We see this pattern more broadly: *smaller models with TSG report higher or comparable satisfaction accuracies to larger models with CFG only*. On the strictly-timed prompts from the TempoControl benchmark, TSG improves temporal accuracy over CFG-only by 17.6–22.0 percentage points across splits, and exceeds full TempoControl by 2.97 and 0.58 percentage points on the one-object and two-object splits, respectively (Table 2).

### 5.3 RQ2. What are the computational costs of TSG?

**Result 2: TSG incurs little to no memory overhead but increases time taken for generation.** TSG matches the CFG-only sampling in terms of the peak GPU memory allocated to the model. This is expected as attention steering detaches the computational graph and hence does not need to store gradients with respect to model weights. TSG uses 30.52 GiB versus 79.43 GiB for full TempoControl, a reduction of approximately 62%, as TempoControl optimizes video latents requiring a gradient computation graph through the entire video transformer (Table 2).

The additional specification-guided forward pass through the transformer incurs a latency cost with TSG. This scales with the size of the model as expected, with TSG taking around 1 extra minute with Wan2.1 1.3B to 11 extra minutes with Wan2.2 A14B (Table 1). Full TempoControl takes approximately 4.9× and 3.6× as long as TSG on the one-object and two-object splits, respectively (Table 2). Limiting TempoControl to one optimization round brings its latency close to TSG's but retains approximately 79.4 GiB peak memory; TSG improves temporal accuracy over this variant by 20.1% and 27.5% relative to its one-object and two-object scores.

**Result 3: TSG modestly impacts the visual quality of the generated videos.** TSG lowers imaging quality by up to 3.0 points on four models and improves it by 3.2 points on Wan2.2 A14B (Table 1).

Table 2: **TempoControl benchmark results with Wan2.1 1.3B.** Parentheses show accuracy changes in percentage points and computational-cost ratios relative to CFG. Latency reports mean ± standard deviation; memory reports peak allocation. We present setup details in Appendix D.3.

| Method | Temporal Acc ↑ | Latency (min/video) ↓ | Peak GPU mem (GiB) ↓ |
|---|---|---|---|
| **One object** (N=80) | | | |
| Base | 69.40% (0.0%) | 1.29 ± 0.01 (1.00×) | 30.48 (1.00×) |
| TempoControl (one round) | 72.44% (+3.0%) | 2.23 ± 0.01 (1.72×) | 79.41 (2.61×) |
| TempoControl (full) | 84.05% (+14.6%) | 9.87 ± 2.10 (7.63×) | 79.43 (2.61×) |
| TSG | 87.02% (+17.6%) | 2.02 ± 0.01 (1.56×) | 30.52 (1.00×) |
| **Two object** (N=82) | | | |
| Base | 36.12% (0.0%) | 1.30 ± 0.01 (1.00×) | 30.48 (1.00×) |
| TempoControl (one round) | 45.53% (+9.4%) | 2.28 ± 0.10 (1.75×) | 79.42 (2.61×) |
| TempoControl (full) | 57.49% (+21.4%) | 8.11 ± 3.41 (6.24×) | 79.43 (2.61×) |
| TSG | 58.07% (+22.0%) | 2.27 ± 0.01 (1.74×) | 30.52 (1.00×) |

## 6 Analysis and Discussion

**Category-wise results and qualitative examples.** Figure 4 (left) shows category-wise satisfaction accuracies, averaged across five models, with whiskers indicating the minimum and maximum. TSG improves mean accuracy across all 11 categories by approximately 2–43 percentage points. The largest gains occur for alternative final states (23% to 66%), two-state transitions (35% to 71%), and reversible state returns (21% to 43%). Performance nevertheless varies substantially across models: with TSG, accuracy ranges from 20% to 75% for reversible state returns and from 10% to 80% for transitions with a persistent entity. Handover with overlap, triggered responses, and three ordered entries remain challenging, with mean accuracies of approximately 9%, 17%, and 12%, respectively, and gains of only about 2 percentage points. These results show that TSG is particularly effective for state-transition requirements, while coordinating multiple entity appearances, disappearances, and dependencies remains challenging. In Appendix A Figure 5, we show examples of temporal requirements where TSG improves adherence over CFG-only sampling.

**Varying the TSG Guidance strength.** We vary the TSG strength $\rho \in \{0, 1, 2, 3, 4, 5\}$ on a fixed 25-prompt subset per model (top-right in Figure 4), with $\rho = 0$ corresponding to CFG-only sampling. Both models attain their highest satisfaction accuracy at $\rho = 1$: 32% for Wan2.1 1.3B and 36% for Wan2.2 5B, improving over CFG-only by 16 and 12 percentage points, respectively. The accuracy is non-increasing beyond $\rho = 1$, and at $\rho = 5$ it matches or falls below the respective CFG-only baseline. These results support $\rho = 1$ as the default among the tested strengths on this subset and show that increasing specification guidance beyond this level does not improve adherence. Appendix D.4 provides the sampling configurations and ablation protocol.

**TSG-CFG interaction.** To evaluate whether the gains from TSG are an artifact of a higher guidance strength alone, we compare it against CFG with twice the guidance strength (bottom-right in Figure 4). We further evaluate the need for the additional prompt-conditional pass by comparing our hybrid CFG + TSG setup against TSG only, and TSG only for the first 15 denoising steps followed by CFG. Combined CFG + TSG reports the highest specification satisfaction accuracies of 32% and 36% on Wan2.1 1.3B and Wan2.2 5B, respectively. Doubling the CFG strength yields 24% and 20%, respectively, falling short of combined guidance on both models. The effects of replacing CFG with TSG are mixed across models. These results support our design choice of combining CFG with TSG.

## 7 Related Work

**Test-time control for video generation.** Methods for controllable video generation differ primarily in terms of the *control surface* and the *time of control* (during or post sampling). Control surfaces include spatiotemporal scene layouts (Lian et al., 2024; Qu et al., 2026), and motion transfer using reference videos (Ling et al., 2024), which are complementary to TSG's focus on temporal constraints over frame-level visual concepts. Closest to our work, TempoControl (Schiber et al., 2026) aligns concept attention traces with prescribed binary temporal masks by repeatedly optimizing video latents

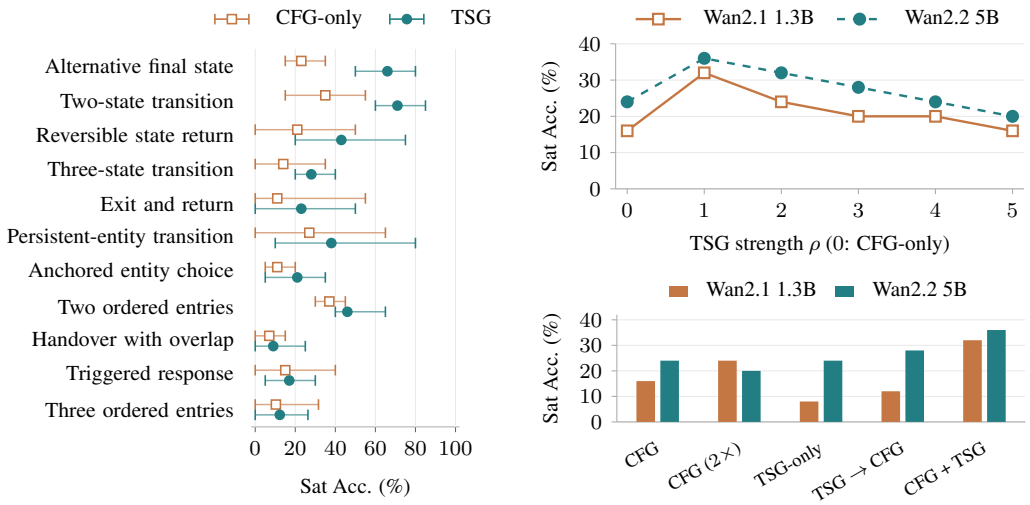


Figure 4: **Category-wise results and ablations.** Left: mean satisfaction accuracy across five models, with min–max whiskers. Right: guidance-strength (top) and CFG–TSG (bottom) ablations.

during sampling. TSG offers control over more expressive temporal constraints and outperforms TempoControl on strict-timed traces with lower computational overhead (Table 2). Post-sampling temporal control methods include NeuS-E (Choi et al., 2025) that uses Vision Language Models (VLM) to detect visual concepts and feedback from a Temporal Logic Model Checker to edit faulty video frames. TSG and NeuS-E offer tradeoffs as TSG does not require access to a VLM or Image Editing model, while NeuS-E can be applied to black-box models. MinT (Wu et al., 2025) learns temporal control by fine-tuning a video diffusion transformer on temporally grounded data. TSG operates at test-time with frozen parameters and no additional training data.

**Attention steering for text, image, and video generation.** Attention provides both a signal of concept grounding and a mechanism for controlling generation of text (Zhang et al., 2024; Guardieiro et al., 2025), and images (Hertz et al., 2022; Chefer et al., 2023). VideoTetris (Tian et al., 2024) composes attention maps across space and time to control spatial positions of concepts. TSG supports richer temporal compositions beyond transitions but does not directly support control of spatial positions.

**Benchmarks for video generation.** VBench (Huang et al., 2024) assesses visual quality, temporal consistency, and semantic alignment, while T2V-CompBench (Sun et al., 2025) evaluates compositional capabilities such as attribute binding, spatial relationships, actions, and motion. These broader evaluations remain complementary to specification satisfaction. TC-Bench (Feng et al., 2024) and NeuS-V (Sharan et al., 2025) evaluate untimed state transitions and temporal relationships, while TempoControl (Schiber et al., 2026) tests exact timing of object appearance. These temporal constraints complement the *interval-timed* constraints in our benchmark. Our benchmark additionally evaluates alternate outcomes through choice, not supported by any prior benchmark.

## 8 Conclusion and Future Work

We present Temporal Specification Guidance (TSG), a training-free framework that turns temporal specifications into executable objectives for guiding video generation during sampling. We instantiate TSG with attention-derived concept traces, differentiable Metric Temporal Logic scores, and attention interventions that avoid backpropagation through the video transformer. This instantiation improves specification adherence across five models on a 220-prompt benchmark spanning 11 categories and achieves comparable temporal accuracy to TempoControl at lower computational cost.

Future work should explore traces of concept evidence beyond video-text attention maps, along with other scoring functions. TSG is limited to the latent temporal resolution of models and extending it to finer-grained temporal constraints is an interesting direction, along with identifying richer specification languages to express temporal requirements.

### AI USE STATEMENT

In this work, we used generative AI tools to assist benchmark construction, including proposing entities and states and drafting prompts and specification instances (Appendix C), and to assist manuscript editing and LaTeX preparation. We have reviewed all AI-assisted work and take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

### REPRODUCIBILITY STATEMENT

We present details on benchmark construction, along with specification templates and example prompts, in Appendix C. We further document settings used with different video generation models, along with detailed evaluation protocol in Appendix D. Finally, we present pseudocode for the TSG algorithm in Appendix F and the quantitative semantics in Appendix E.

## REFERENCES

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.

Hila Chefer, Yuval Alaluf, Yael Vinker, Lior Wolf, and Daniel Cohen-Or. Attend-and-excite: Attention-based semantic guidance for text-to-image diffusion models. *ACM transactions on Graphics (TOG)*, 42(4):1–10, 2023.

Minkyu Choi, SP Sharan, Harsh Goel, Sahil Shah, and Sandeep Chinchali. We'll fix it in post: Improving text-to-video generation with neuro-symbolic feedback. *arXiv preprint arXiv:2504.17180*, 2025.

Omer Dahary, Or Patashnik, Kfir Aberman, and Daniel Cohen-Or. Be yourself: Bounded attention for multi-subject text-to-image generation. In *European Conference on Computer Vision*, pp. 432–448. Springer, 2024.

Weixi Feng, Jiachen Li, Michael Saxon, Tsu-jui Fu, Wenhu Chen, and William Yang Wang. Tc-bench: Benchmarking temporal compositionality in text-to-video and image-to-video generation. *arXiv preprint arXiv:2406.08656*, 2024.

Weixi Feng, Chao Liu, Sifei Liu, William Yang Wang, Arash Vahdat, and Weili Nie. Blobgen-vid: Compositional text-to-video generation with blob video representations. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12989–12998. IEEE, 2025.

Vitoria Guardieiro, Avishree Khare, Adam Stein, and Eric Wong. Instruction following by principled boosting attention of large language models. *arXiv preprint arXiv:2506.13734*, 2025.

Hao He, Yinghao Xu, Yuwei Guo, Gordon Wetzstein, Bo Dai, Hongsheng Li, and Ceyuan Yang. Cameractrl: Enabling camera control for text-to-video generation. *arXiv preprint arXiv:2404.02101*, 2024.

Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control. *arXiv preprint arXiv:2208.01626*, 2022.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, et al. Vbench: Comprehensive benchmark suite for video generative models. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21807–21818. IEEE, 2024.

Avishree Khare, Hideki Okamoto, Bardh Hoxha, Georgios Fainekos, and Rajeev Alur. Logstop: Temporal scores over prediction sequences for matching and retrieval. *arXiv preprint arXiv:2510.06512*, 2025.

Ron Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Syst.*, 2(4): 255–299, October 1990. ISSN 0922-6443. doi: 10.1007/BF01995674. URL https://doi.org/10.1007/BF01995674.

Long Lian, Baifeng Shi, Adam Yala, Boyi Li, et al. Llm-grounded video diffusion models. In *International Conference on Learning Representations*, volume 2024, pp. 50207–50227, 2024.

Pengyang Ling, Jiazi Bu, Pan Zhang, Xiaoyi Dong, Yuhang Zang, Tong Wu, Huaian Chen, Jiaqi Wang, and Yi Jin. Motionclone: Training-free motion cloning for controllable video generation. *arXiv preprint arXiv:2406.05338*, 2024.

Leigang Qu, Ziyang Wang, Na Zheng, Wenjie Wang, Liqiang Nie, and Tat-Seng Chua. Ttom: Test-time optimization and memorization for compositional video generation. In *International Conference on Learning Representations*, volume 2026, pp. 70225–70247, 2026.

Shira Schiber, Ofir Lindenbaum, and Idan Schwartz. Tempocontrol: Temporal attention guidance for text-to-video models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 36670–36679, 2026.

SP Sharan, Minkyu Choi, Sahil Shah, Harsh Goel, Mohammad Omama, and Sandeep Chinchali. Neuro-symbolic evaluation of text-to-video models using formal verification. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8395–8405. IEEE, 2025.

Ariel Shaulov, Itay Hazan, Lior Wolf, and Hila Chefer. Flowmo: Variance-based flow guidance for coherent motion in video generation. *Advances in Neural Information Processing Systems*, 38: 82725–82754, 2026.

Kaiyue Sun, Kaiyi Huang, Xian Liu, Yue Wu, Zihan Xu, Zhenguo Li, and Xihui Liu. T2v-compbench: A comprehensive benchmark for compositional text-to-video generation. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8406–8416. IEEE, 2025.

Ye Tian, Ling Yang, Haotian Yang, Yuan Gao, Yufan Deng, Jingmin Chen, Xintao Wang, Zhaochen Yu, Xin Tao, Pengfei Wan, et al. Videotetris: Towards compositional text-to-video generation. *Advances in Neural Information Processing Systems*, 37:29489–29513, 2024.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Ao Wang, Hui Chen, Lihao Liu, Kai Chen, Zijia Lin, Jungong Han, and Guiguang Ding. Yolov10: Real-time end-to-end object detection. *Advances in neural information processing systems*, 37: 107984–108011, 2024.

Ziyi Wu, Aliaksandr Siarohin, Willi Menapace, Ivan Skorokhodov, Yuwei Fang, Varnith Chordia, Igor Gilitschenski, and Sergey Tulyakov. Mind the time: Temporally-controlled multi-event video generation. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 23989–24000. IEEE, 2025.

Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, et al. Cogvideox: Text-to-video diffusion models with an expert transformer. In *International Conference on Learning Representations*, volume 2025, pp. 83048–83077, 2025.

Qingru Zhang, Chandan Singh, Liyuan Liu, Xiaodong Liu, Bin Yu, Jianfeng Gao, and Tuo Zhao. Tell your model where to attend: Post-hoc attention steering for llms. In *International Conference on Learning Representations*, volume 2024, pp. 42411–42430, 2024.

## A EXAMPLES WITH TSG

Figure 5 presents three examples of improved adherence with TSG.

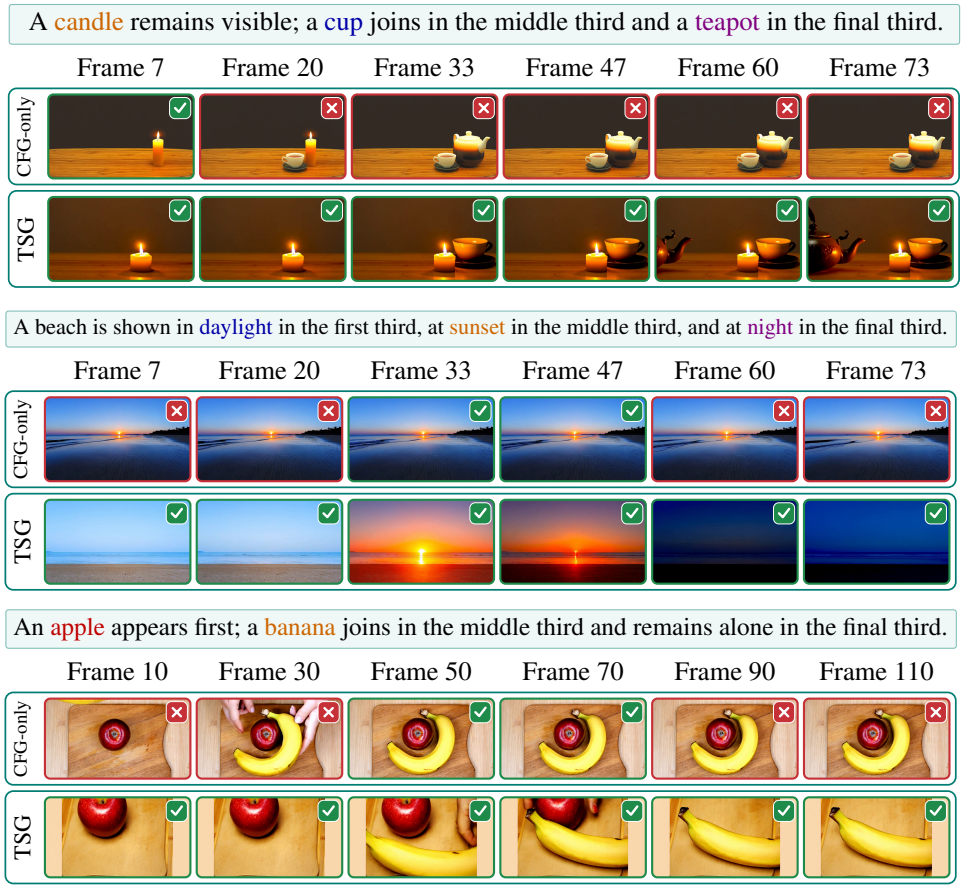


Figure 5: Representative examples comparing CFG-only sampling (top row of each panel) with TSG (bottom row) on ordered entry, three-state transition, and overlapping handover respectively.

## B MTL SEMANTICS AND EXAMPLES

Given the frame-level labels $y(v_i, c)$ for visual concepts, the label $y(v, \varphi, t)$ for a video $v$ satisfying specification $\varphi$ starting from frame $1 \leq t \leq T$ can be computed deterministically using the standard Boolean semantics for $MTL_f$ as follows:

- $y(v, \top, t) = 1$ and $y(v, \bot, t) = 0$.
- $y(v, c, t) = y(v_t, c)$.
- $y(v, \neg c, t) = 1$ iff $y(v, c, t) = 0$.
- $y(v, \varphi_1 \wedge \varphi_2, t) = 1$ iff $y(v, \varphi_1, t) = 1$ and $y(v, \varphi_2, t) = 1$.
- $y(v, \varphi_1 \vee \varphi_2, t) = 1$ iff $y(v, \varphi_1, t) = 1$ or $y(v, \varphi_2, t) = 1$.
- $y(v, \mathbf{X}_I \varphi, t) = 1$ iff $t < T$, $a_I \leq 1 \leq b_I$, and $y(v, \varphi, t + 1) = 1$
- $y(v, \mathbf{G}_I \varphi, t) = 1$ iff $y(v, \varphi, t') = 1$ for every integer $t'$ satisfying $t + a_I \leq t' \leq \min\{t + b_I, T\}$.
- $y(v, \varphi_1 \mathbf{U}_I \varphi_2, t) = 1$ iff there exists an integer $t'$ satisfying $t + a_I \leq t' \leq \min\{t + b_I, T\}$ such that $y(v, \varphi_2, t') = 1$ and $y(v, \varphi_1, t'') = 1$ for every integer $t''$ satisfying $t \leq t'' < t'$.

where $[a_I, b_I]$ is the integer-offset interval corresponding to $I = [t_1, t_2]$, with $a_I = \lceil t_1(T-1) \rceil$ and $b_I = \lfloor t_2(T-1) \rfloor$.

**Example: departure with different timing constraints.** For instance, we can represent "a dog leaves the scene" in a 5-s long video, with decreasing flexibility in timing as:

$$\underbrace{\mathsf{dog}\,\mathbf{U}\,(\mathbf{G}\neg\mathsf{dog})}_{\text{Leaves at an unspecified time}} \qquad \underbrace{\mathsf{dog}\,\mathbf{U}_{[0.5,1]}\,(\mathbf{G}\neg\mathsf{dog})}_{\text{Leaves at some point during the second half}} \qquad \underbrace{\mathsf{dog}\,\mathbf{U}_{[0.6,0.6]}\,(\mathbf{G}\neg\mathsf{dog})}_{\text{Leaves at the 3s mark}}.$$

## C THE TSG BENCHMARK

Table 3: The 11 temporal constraint categories in the main benchmark, each containing 20 examples. Descriptions summarize the temporal structure; individual specifications also encode timing, absence, and persistence requirements.

| Category | Temporal behavior |
|---|---|
| Two-state transition | A subject or setting changes from state $A$ to $B$. |
| Three-state transition | A subject or setting progresses through $A$, $B$, then $C$. |
| Ordered entry of two entities | One entity is initially present; a second joins it and both remain. |
| Ordered entry of three entities | Two further entities enter in sequence, accumulating in the scene. |
| Entity handover with overlap | $A$ alone is followed by $A$ and $B$ together, then $B$ alone. |
| Entity exit and re-entry | An entity is present, becomes absent, then returns. |
| Reversible state return | A subject changes from $A$ to $B$ and returns to $A$. |
| Anchored entity choice | An anchor persists while one of two alternative entities appears. |
| Transition with a persistent entity or setting | An anchor remains present while another concept changes state. |
| Alternative final state | An initial state gives way to either of two permitted final states. |
| Trigger-conditioned state transition | A trigger entity appears before the target changes state. |

**Visual concepts as states of entities.** We first use `GPT-5.6-Sol` to propose 50 entities spanning object and setting classes, together with associated states or attributes, such as color and time of day. This collection includes object classes from VBench (Huang et al., 2024), augmented with states. One author manually reviews the collection to exclude unfamiliar entities and implausible entity–state combinations. Visual concepts represent entity presence or an entity assuming a particular state. The final inventory contains 50 entries: 36 objects and 14 scenes or settings. Examples of object entities with states are presented in Table 4 and of setting entities with states in Table 5.

**Benchmark categories.** We manually define 11 categories of temporal constraints covering state transitions, entity appearance and disappearance, and choice. Table 3 illustrates each category. For each category, one author collaborates with `GPT-5.6-Sol` to construct 20 prompts, reviewing them for relevance, diversity of entities and states, and realism. We specify temporal intervals using halves and thirds of the video duration, allowing meaningful time for each event in a five-second video.

**Specification templates for different categories.** We then instantiate manually designed specification templates to encode each prompt's temporal requirements. These interval-timed specifications allow the requested transitions to occur anywhere within the designated intervals, subject to any ordering and persistence requirements. Thus, as in the examples in Section 2, they do not prescribe a frame-by-frame schedule for each concept.

Let $T$ denote video duration. We use half intervals $H_1 = [0, T/2)$ and $H_2 = [T/2, T]$, or thirds $I_1 = [0, T/3)$, $I_2 = [T/3, 2T/3)$, and $I_3 = [2T/3, T]$. Thus, adjacent phases do not share a boundary frame. For a phase with positive propositions $P^+$ and negative propositions $P^-$, its predicate is

$$p = \bigwedge_{a \in P^+} a \;\wedge\; \bigwedge_{b \in P^-} \neg b.$$

| Entity | Candidate states or attributes |
|---|---|
| airplane | configuration: landed, in flight |
| apple | condition: whole, sliced |
| balloon | inflation: visibly deflated, visibly inflated |
| bicycle | orientation: upright, lying on its side |
| bird | configuration: perched, in flight |
| boat | orientation: upright, capsized |
| book | position: closed, open |
| butterfly | configuration: perched, in flight |
| campfire | flame visibility: no visible flame, visible flames |
| candle | flame: lit, unlit |
| car | orientation: upright, overturned; headlight appearance: headlights visibly on, headlights visibly off |
| cat | pose: standing, sitting, lying down |
| cup | fill level: visibly empty, filled to the rim |
| dog | pose: standing, sitting, lying down |
| door | position: closed, open |
| egg | shell: intact, cracked |
| faucet | water visibility: no visible water stream, visible water stream |
| flag | position: fully lowered, fully raised |

Table 4: Examples of object entities with corresponding states.

| Entity | Candidate states or attributes |
|---|---|
| beach | illumination: bright daylight, orange horizon, dark night |
| city street | illumination: bright daylight, dark night; precipitation: no visible precipitation, visible falling rain, visible falling snow |
| farm field | ground cover: green crops, bare soil, snow-covered; precipitation: no visible precipitation, visible falling rain, visible falling snow |
| forest | dominant foliage: green foliage, autumn-colored foliage, completely bare branches, snow-covered trees; precipitation: no visible precipitation, visible falling rain, visible falling snow |
| kitchen | artificial lighting: lights visibly illuminated, no lights visibly illuminated |
| lake | surface: visible open water, completely ice-covered; visibility: far shore clearly visible, far shore obscured by fog; precipitation: no visible precipitation, visible falling rain, visible falling snow |
| mountain landscape | illumination: bright daylight, orange horizon, dark night; visibility: mountains clearly visible, mountains obscured by fog |
| ocean | illumination: bright daylight, orange horizon, dark night; sea appearance: calm surface, large breaking waves |
| park | illumination: bright daylight, dark night; precipitation: no visible precipitation, visible falling rain, visible falling snow |
| road | surface: visibly dry, visibly wet, snow-covered |
| room | artificial lighting: lights visibly illuminated, no lights visibly illuminated |
| sky | illumination: bright daylight, orange horizon, dark night; cloud appearance: mostly clear sky, overcast sky, dark storm clouds |
| stadium | occupancy: no visible spectators, spectators visible; floodlight appearance: floodlights visibly illuminated, floodlights visibly unlit |
| waterfall | water appearance: visible flowing water, completely frozen; surroundings: green vegetation, snow-covered ground |

Table 5: Examples of scene and setting entities with corresponding states.

Only the listed propositions are constrained; unrelated scene content is not prohibited. A state proposition includes its entity (e.g., *sitting dog*), and persistent-entity transitions retain these compound propositions rather than factoring them into separate entity and state atoms.

For compactness, write $W_J(p)$ for an eventually–always requirement: there must be a consecutive run satisfying $p$ wholly within $J$, with the steering persistence parameter set to $\delta = 0.5$ seconds. In

continuous notation this is $\mathbf{F}_{\{t\in J:[t,t+\delta]\subseteq J\}}\mathbf{G}_{[0,\delta]}p$; the discrete implementation is specified below. Define

$$S_2(p_1,p_2)=W_{H_1}(p_1)\wedge W_{H_2}(p_2),\qquad S_3(p_1,p_2,p_3)=\bigwedge_{j=1}^{3}W_{I_j}(p_j).$$

The steering templates are:

1. *Two-state transition (2ST):* $S_2(A\wedge\neg B, B\wedge\neg A)$.
2. *Three-state transition (TST):* $S_3(A\wedge\neg B\wedge\neg C, B\wedge\neg A\wedge\neg C, C\wedge\neg A\wedge\neg B)$.
3. *Two ordered entries (OE2):* $S_2(A\wedge\neg B, A\wedge B)$.
4. *Three ordered entries (OE3):* $S_3(A\wedge\neg B\wedge\neg C, A\wedge B\wedge\neg C, A\wedge B\wedge C)$.
5. *Handover with overlap (HWO):* $S_3(A\wedge\neg B, A\wedge B, B\wedge\neg A)$.
6. *Exit and return (EXR):* $S_3(A,\neg A,A)$.
7. *Reversible state return (RSR):* $S_3(A\wedge\neg B, B\wedge\neg A, A\wedge\neg B)$.
8. *Persistent-entity transition (PST):* $\mathbf{G}_{[0,T]}E\wedge S_2(A\wedge\neg B, B\wedge\neg A)$.
9. *Triggered response (TRG):* $S_3(A\wedge\neg B\wedge\neg E, A\wedge E\wedge\neg B, B\wedge E\wedge\neg A)$. Here $E$ is the entering trigger and $A, B$ are the original and response states. This specifies an observed temporal sequence, not a causal intervention.
10. *Alternative final state (AFS):* $\mathbf{G}_{I_1}(A\wedge\neg B\wedge\neg C)\wedge[\mathbf{G}_{I_3}(B\wedge\neg A\wedge\neg C)\vee\mathbf{G}_{I_3}(C\wedge\neg A\wedge\neg B)]$.
11. *Anchored entity choice (AEC):* $\mathbf{G}_{[0,T]}E\wedge\mathbf{G}_{I_1}(\neg A\wedge\neg B)\wedge[\mathbf{G}_{I_3}(A\wedge\neg B)\vee\mathbf{G}_{I_3}(B\wedge\neg A)]$.

In the state-transition templates, $A, B, C$ refer to states of the same subject or setting; in the entry, handover, and entity-choice templates they denote entity presence. $E$ denotes an anchor or trigger as indicated. Every category has 20 examples. Choice formulas use an ordinary disjunction between *exclusive branch predicates*: either branch is acceptable, but both alternatives cannot satisfy the same required frame. Placing the disjunction outside the final-region always operators selects one fixed branch throughout that region. The middle third is unconstrained by AFS steering and, apart from anchor persistence, by AEC steering; these formulas do not prohibit transient alternative states in that gap.

**Concrete examples of prompts.** The following are exact released prompts, with proposition assignments that instantiate the corresponding templates above.

1. *Two-state transition* (`2st-181_person_stand_to_sit`). $A=$ standing person; $B=$ sitting person.
   "A continuous full-body studio shot follows one person against a simple background. The person begins standing and holds that pose for roughly the first half of the video. Around the midpoint, they sit down on the floor and remain seated for the rest of the video. The same person remains in frame throughout, with no cuts, duplicate people, or camera movement."
2. *Three-state transition* (`tst-001_traffic_light_yellow_red_green`). $A=$ yellow traffic light; $B=$ red traffic light; $C=$ green traffic light.
   "A cinematic, photorealistic 4K close-up follows one street traffic light at dusk from a fixed tripod camera. The signal is solid yellow for roughly the first third of the video. About one-third of the way through, that same signal changes cleanly to solid red. Around two-thirds of the way through, it changes to solid green and remains green through the end. The softly blurred urban background stays unchanged, with no camera movement, scene cuts, duplicate traffic lights, or extra signal flicker."
3. *Ordered entry of two entities* (`oe2-201_person_then_dog`). $A=$ person; $B=$ dog.
   "One person waits alone in a quiet park clearing for roughly the first half of a continuous medium-wide shot. Around the midpoint, a dog runs into the clearing and stops nearby. The same person and dog remain fully visible together through the end, with no cuts, duplicate subjects, or camera movement."
4. *Ordered entry of three entities* (`oe3-121_dog_cat_rabbit`). $A=$ dog; $B=$ cat; $C=$ rabbit.

"One dog waits alone on a broad living-room rug for roughly the first third of a fixed shot. A cat then walks into the room and settles within the same view while the dog stays. Around two-thirds of the way through, a rabbit hops in through the open doorway. The dog, cat, and rabbit all remain visible together until the end, with no duplicate animals or camera movement."

5. *Entity handover with overlap* (`hwo-021_horse_to_rabbit`). $A =$ horse; $B =$ rabbit.
   "A horse stands alone in a grassy paddock at the beginning of a steady wide shot. About one-third of the way through, a rabbit hops into the foreground while the horse remains nearby. Around the two-thirds point, the horse walks calmly out through an open gate, leaving the rabbit in view until the end. The camera does not cut or change position."
6. *Entity exit and re-entry* (`exr-061_dog_exit_return`). $A =$ dog.
   "A fixed living-room shot begins with one dog standing in clear view. About one-third of the way through, the dog walks out through the left side until it is completely absent. Near the two-thirds point, the same dog walks back in from the left and remains in the room until the end. There are no cuts or other dogs."
7. *Reversible state return* (`rsr-081_person_stand_sit_stand`). $A =$ standing person; $B =$ sitting person.
   "A continuous full-body studio shot begins with one person standing against a simple background. About one-third of the way through, they lower themselves and sit on the floor. Near the two-thirds point, the same person rises to stand again and remains standing through the end. The camera stays fixed without cuts or other people."
8. *State transition with a persistent entity or setting* (`pst-101_person_dog_stand_sit`). $E =$ person; $A =$ standing dog; $B =$ sitting dog.
   "A person and one standing dog share a clear full-body studio shot. The person stays in the same spot throughout the video. Around halfway through, the dog lowers into a seated pose and remains sitting beside the person until the end. The camera does not move and no additional subjects enter."
9. *Trigger-conditioned state transition* (`trg-041_person_triggers_dog_standing`). $E =$ person; $A =$ sitting dog; $B =$ standing dog.
   "A continuous full-body shot begins with one dog sitting alone in a quiet room. The dog remains seated for roughly the first third of the video. About one-third of the way through, a person walks into the frame and approaches while the dog remains seated. Around two-thirds of the way through, the dog responds to the person by standing up, and the person remains nearby through the end. The same dog and person stay in frame, with no cuts, duplicate subjects, or camera movement."
10. *Alternative final state* (`afs-161_dog_sit_to_stand_or_lie`). $A =$ sitting dog; $B =$ standing dog; $C =$ lying dog.
    "One dog sits upright in a clear full-body studio view for the opening third. During the middle third, it either rises onto all four paws or lowers into a lying pose, but it does not perform both actions. By the start of the final third, the dog is either standing or lying and remains in that selected pose through the end."
11. *Anchored entity choice* (`aec-141_tree_dog_or_cat`). $E =$ tree; $A =$ dog; $B =$ cat.
    "A single tree remains centered and alone throughout the opening third of a fixed garden view. During the middle third, either one dog may walk in or one cat may walk in, but not both. Whichever animal arrives stays near the tree throughout the final third without any duplicate animals."

**Evaluation specifications.** We assign each sampled frame its timestamp and use half-open phase intervals, except that the final phase includes the video endpoint. We evaluate temporal adherence using a phase-based criterion. Let $t_i = i\Delta$ denote the sampled timestamps and let $r = \lceil \delta/\Delta \rceil$, where $\delta = 0.5$ seconds. A phase predicate $p$ satisfies the persistence requirement within interval $J$ if

$$W_J(p) = 1 \quad \Longleftrightarrow \quad \exists i : \{t_i, \ldots, t_{i+r}\} \subseteq J \ \wedge \ \bigwedge_{k=0}^{r} p(v_{i+k}),$$

| Model | Resolution | Frames/FPS | Latent frames | Steps | CFG |
|---|---|---|---|---|---|
| Wan 2.1 1.3B | $480 \times 832$ | $81/16$ | 21 | 50 | 6 |
| Wan 2.2 TI2V 5B | $704 \times 1280$ | $121/24$ | 31 | 50 | 5 |
| Wan 2.2 T2V A14B | $704 \times 1280$ | $81/16$ | 21 | 40 | 4 / 3 |
| CogVideoX 2B | $480 \times 720$ | $49/8$ | 13 | 50 | 6 |
| CogVideoX 5B | $480 \times 720$ | $49/8$ | 13 | 50 | dynamic (6) |

Table 6: Generation settings for the TSG benchmark. Wan 2.2 A14B uses CFG scales 4 and 3 for its high- and low-noise experts, respectively. CogVideoX 5B uses its dynamic CFG schedule with scale parameter 6. All methods compared within a model share these settings. The TempoControl comparison uses the Wan 2.1 row.

where the witness must use valid sampled-frame indices. Thus, $p$ must hold at $r + 1$ consecutive sampled frames whose first-to-last timestamp span is at least $\delta$, with the entire run contained in $J$.

The predicate $p$ jointly encodes the required presence and absence of visual concepts. Explicit persistence requirements are evaluated over their stated intervals rather than reduced to a $0.5$-second witness. We use the same duration threshold across the evaluated 5–6-second videos to require sustained evidence beyond an isolated frame while allowing flexibility in transition timing.

*Interpretation of the criterion.* We evaluate whether each required phase occurs within its designated interval and persists for at least 0.5 seconds. We intentionally leave the intervening transitions unconstrained: intermediate configurations may not map unambiguously to the discrete propositions. For example, a partially open door may be labeled either open or closed while it moves between states. Requiring a sustained witness for each phase establishes that the requested states occur in the specified temporal order without prescribing the duration or frame-by-frame labeling of the transition between them. This criterion does not require each phase to occupy its entire interval or restrict behavior outside the witness runs, except where the specification imposes explicit persistence or absence constraints.

# D Experimental setup and evaluation

## D.1 Models and sampling settings

We evaluate TSG on the 220 prompt–specification pairs described in Appendix C. Table 6 summarizes the generation settings. Resolutions are given as height $\times$ width. We define duration as the time between the first and last decoded frames, $T = (F - 1)/f$, where $F$ is the frame count and $f$ is the frame rate. Thus the Wan clips span five seconds and the CogVideoX clips span six seconds. Fractional specification intervals are compiled for each model's actual duration and temporal grid.

We use the diffusers library to obtain model implementations. The checkpoints are `Wan-AI/Wan2.1-T2V-1.3B-Diffusers`, `Wan-AI/Wan2.2-TI2V-5B-Diffusers`, `Wan-AI/Wan2.2-T2V-A14B-Diffusers`, `THUDM/CogVideoX-2b`, and `THUDM/CogVideoX-5b`. The TI2V checkpoint is used in text-to-video mode. Wan uses `UniPCMultistepScheduler`, with flow shift 3 for Wan 2.1 and 5 for both Wan 2.2 models. The fused prediction is passed to the scheduler once per denoising step, preserving its multistep state. CogVideoX 2B uses `CogVideoXDDIMScheduler`; CogVideoX 5B uses `CogVideoXDPMScheduler`. Both use trailing timestep spacing. For CogVideoX 5B, the implemented dynamic scale is

$$g(t) = 1 + 6\,\frac{1 - \cos\Big(\pi\,((N - t)/N)^5\Big)}{2},$$

where $N = 50$ and $t$ is the scheduler timestep value, not the zero-based sampling-step index. Other CFG scales remain constant, except for the expert-dependent A14B scales.

For each model and method, we generate one video per prompt. The TSG benchmark seed is $5000 + j$, where $j$ is the numeric component of the example identifier. Each base run saves its initial latent tensor; the paired TSG run reloads that tensor, with shape, dtype, and SHA-256 checks. CogVideoX 5B additionally records and replays the sampling-generator state used by its stochastic DPM sampler.

Prompts and negative prompts are identical within each paired comparison. The Wan negative prompt on this benchmark is: "bright tones, overexposed, static, blurred details, subtitles, text, watermark, low quality, worst quality, jpeg artifacts, deformed". CogVideoX uses no negative prompt.

### D.2 TSG IMPLEMENTATION AND HYPERPARAMETERS

The main TSG benchmark uses online attention steering with norm ratio $\rho = 1$, applied during the first 15 denoising steps (indices 0–14). Ordinary CFG remains active throughout sampling. We set the additional weight to zero when the residual norm is at most $10^{-12}$. We use the z-score–sigmoid calibration with temperature $\tau = 1.0$ and denominator stabilizer $\epsilon = 10^{-6}$, holding the temporal mean and standard deviation fixed during differentiation. After broadcasting and centering the attention-logit biases, we clip them to $[b_{\min}, b_{\max}] = [-1.0, 1.0]$. We use bias step size $\eta = 1$. We steer all attention heads in all transformer layers: 30 layers for Wan 2.1, Wan 2.2 TI2V, and CogVideoX 2B; 40 per active expert for Wan 2.2 A14B; and 42 for CogVideoX 5B.

### D.3 TEMPOCONTROL COMPARISON

We use all 80 one-object and 82 two-object prompts from the official TempoControl benchmark repository. One-object prompts request an initially absent object that appears during the second, third, fourth, or last second and remains visible, with 20 prompts for each onset. Two-object prompts require a static object throughout, with another object appearing in the second half. Binary masks are derived from the requested timing and the actual video frame rate and duration, rather than reusing the original 21-bin masks at a different duration. For our five-second Wan clips, the four one-object onsets occur at sampled indices 4, 8, 12, and 16, respectively; the object must be absent before its onset and present from that index through index 20. Two-object masks require absence of the entering object at indices 0–9 and presence at indices 10–20, together with the static object throughout.

To obtain TSG specifications, we convert each maximal constant segment of a binary mask into an always requirement for the proposition or its negation and conjoin the segment requirements. The static-object requirement is a global always formula. The runtime steering grid with Wan2.1 1.3B has 21 latent frames. All four comparison methods, i.e., CFG-only, TempoControl, one-round TempoControl, and TSG, use Wan 2.1 at $480 \times 832$, 81 frames, 16 FPS, 50 steps, CFG 6, and flow shift 3. We use seed 42 and replay the initial latent tensor saved by the full TempoControl run for each prompt.

Our TempoControl port preserves the released optimization order using the Diffusers backbone: latent optimization follows the scheduler update and queries attention at the preceding timestep. It optimizes the latent with AdamW (weight decay 0.01) at steps 0–10, with learning rate $5 \times 10^{-4}$ for one-object and $10^{-3}$ for two-object prompts. The objective uses energy weight 0.3 and entropy weight 10, with query chunks of 512 positions and activation checkpointing for the attention forward pass. The full variant allows up to ten optimization updates per guided step, following official recommendations in the repository. Early stopping checks whether the first controlled concept's Pearson correlation with its target mask exceeds 0.9, before applying the next update. The one-round variant changes only the maximum update count to one; it retains this early-stopping check and the same 11 guided steps.

### D.4 ABLATION PROTOCOL AND PARAMETER SELECTION

The guidance-strength ablations in Section 6 compare Wan 2.1 and Wan 2.2 TI2V on the same 25 samples. Every configuration replays its model's saved base latent tensor for each example. We retain the model-specific settings in Table 6, all attention layers and heads, the first 15 guided steps, step size 1, and the calibration and clipping settings above. The additional strengths are $\rho \in \{2, 3, 4, 5\}$, compared with the existing base and $\rho = 1$ runs. Only $\rho$ changes within this sweep.

The CFG interaction ablations also use the same base settings but modify the sampling as required. For the double strength CFG runs, we double the CFG strengths in Table 6 for Wan2.1 and Wan2.2 5B. For TSG-only, we run TSG for all 50 steps and replace the conditional directions in CFG's update with the TSG predicted specification-guided direction (CFG's guidance scale is used).

### D.5 Compute and measurement protocol

Wan 2.1, Wan 2.2 TI2V, both CogVideoX models, and the TempoControl comparison are run on a cluster with NVIDIA B200 GPUs. The Wan 2.2 A14B model is run on a cluster with NVIDIA H200s. The models use bfloat16 except CogVideoX 2B, which uses float16; the VAE uses float32. The recorded runtime is PyTorch 2.9.1 with CUDA 12.8, Diffusers 0.36.0, and Transformers 4.57.3. CogVideoX enables VAE tiling and slicing. The attention implementation uses PyTorch scaled-dot-product attention for ordinary computation and explicit attention-score computation where needed for capture and steering; the exact kernel selected by the runtime is not recorded in the per-video profiles.

Timing uses a monotonic clock with CUDA synchronization at measurement boundaries. Before each measured video, the profiler empties the CUDA allocator cache and resets peak-memory statistics after model loading. There is no separate discarded warm-up video in this measurement procedure. Generation latency includes initial-noise preparation or replay, prompt encoding, denoising and guidance, and VAE decoding. It excludes model loading, video export, evaluation, and separately timed diagnostic artifacts. Diagnostic memory allocations can still contribute to the recorded peak. When reported as mean $\pm$ standard deviation, the standard deviation is the sample standard deviation across videos, not a confidence interval or variation over repeated seeds. Peak memory is the maximum of the per-video PyTorch allocated-memory peaks, including resident model weights, converted to GiB; it is not device-wide memory usage or allocator-reserved memory. Hardware differs for A14B, so cross-model latency comparisons also reflect that difference.

### D.6 Evaluation metrics

**Frame sampling and specification accuracy.** We evaluate 21 uniformly spaced decoded frames for every model, including the first and last frame. For a video with $F$ frames, sampled index $i$ is $\lfloor i(F-1)/20 \rfloor$, for $i = 0, \ldots, 20$. We extract binary concept traces using `Qwen/Qwen2.5-VL-7B-Instruct`. *Specification satisfaction accuracy (Sat Acc.)* evaluates the prompt-derived specification under the Boolean semantics from Appendix B.

**Visual quality.** We use the MUSIQ-SPAQ model underlying VBench's imaging-quality dimension (Huang et al., 2024). It scores the same sampled frames used for specification evaluation, resizing the longer side to at most 512 pixels according to the VBench preprocessing. We average over frames within each video and then over videos. The table reports raw MUSIQ units, with larger values indicating better perceptual frame quality. Latency and peak GPU memory follow the measurement definitions above.

**Evaluator configuration.** The Qwen evaluator makes one request per sampled frame, asking for all relevant propositions together. The frame is converted to a uint8 PIL image and passed through the model's image processor and chat template. The application sends one user message containing the image and the verification prompt. Decoding is greedy, with at most 256 new tokens and one frame per request. The evaluator sees proposition definitions but not the requested temporal schedule or neighboring frames.

The fixed user-prompt preamble is reproduced below; the schema-specific field definitions are then appended from each example's proposition contexts.

> Inspect only the supplied frame. Report what is visibly supported, not what ought to happen. The scene may be stylized or synthetic. Do not infer missing objects or a desired temporal phase. Return only one JSON object with exactly the keys described below, without explanations.
>
> Object-presence fields: "present" means any recognizable visible part, even at the edge, small, or partially occluded. "absent" means confidently not visible. "unclear" means the evidence cannot establish presence or absence; do not equate uncertainty with absence.
>
> State fields: return a JSON list of the exact state names visibly supported. Normally one state applies. If multiple states visibly coexist, list all of them; do not force a valid phase. Return [] only when you can confidently establish that none applies. Return ["unclear"] if the state cannot be determined, the transition is ambiguous, or the relevant entity is not recognizable. Do not combine "unclear" with other labels. Use the same visible entity when distinguishing its states. A closed door is shut; an open or ajar door is not closed. If the opening cannot be seen clearly, use unclear. Standing and sitting refer to the actual pose; an

> ambiguous intermediate pose is unclear. For traffic lights, judge the illuminated signal, not an unlit colored lens.

For example, a pose-transition request appends a `state` field listing the allowed compound state labels and their definitions. Entity-presence fields instead allow `present`, `absent`, or `unclear`. City-lighting alternatives additionally receive explicit daytime, sunset, and nighttime definitions.

Parsing accepts a JSON object, optionally within a JSON code fence, and normalizes case, whitespace, underscores, and Unicode dash separators in labels. Multiple supported states may be returned simultaneously; we do not force the answer to match a valid temporal phase. Explicit unclear labels and omitted fields map to false in the binary trace. Unknown fields are ignored; invalid known fields, unknown state labels, and inconsistent aliases raise protocol errors. The current evaluator allows two schema-correction retries on the same frame and preserves the invalid responses; exhausted retries leave an error record rather than a fabricated verdict. Earlier saved evaluations without retries remain identified by their raw-response records and source snapshots.

Across all five models, Qwen returned "unclear" labels for at least one proposition in 433 of 46,200 individual frames (0.94%). We assessed sensitivity to these labels using conservative and optimistic treatments across all five models. Conservatively, unclear labels support neither positive nor negative literals; *no currently passing video fails under this treatment*, leaving reported accuracies unchanged. Optimistically, unclear labels may take Boolean values that satisfy the specification. With models weighted equally, this increases CFG accuracy by $0.73$ percentage points and TSG accuracy by $1.36$ percentage points.

**TempoControl object and action metrics.** Object evaluation uses the `YOLOv10x` detector with Ultralytics 8.4.152 on 21 sampled frames. The detector is invoked with its prediction defaults. Detected class names and requested labels are compared literally after lowercasing.

For one-object clips, a sampled frame is correct if detected presence matches the target mask. For two-object clips, correctness additionally requires the static object in both phases: before entry, the static object must be detected and the entering object absent; after entry, both must be detected. Temporal accuracy is the mean per-video fraction of correct frames. This frame-wise metric differs from the whole-video specification accuracy used on the TSG benchmark.

# E Interval-aware specification scores

We define a differentiable sum–product score for the fragment of $MTL_f$ in Section 2. Inspired by the log-domain computation of LogSTOP (Khare et al., 2025), this construction aggregates support for the witnesses permitted by a specification. The score need not lie in $[0, 1]$ and is not interpreted as a probability. In particular, disjunction and existential temporal operators sum support for alternatives, rather than combining them through complements of composite scores.

## E.1 Score definition and evaluation

Fix a sampling step, attention layer, and head, and write $p_{t,c} = \hat{y}(z_t, c) \in [0, 1]$ for the calibrated concept scores at latent frames $t = 1, \ldots, N$, where $N = T_z$. Let $Q_\psi[t] \geq 0$ denote the score of subformula $\psi$ starting at frame $t$, and let $S_\psi[t] = \log Q_\psi[t]$, with $\log 0 = -\infty$. The scalar objective for steering is $\hat{y}(z, \varphi) = S_\varphi[1]$.

For a normalized interval $I = [\ell_I, u_I]$, define

$$\begin{aligned} a_I &= \lceil \ell_I (N-1) \rceil, \qquad b_I = \lfloor u_I (N-1) \rfloor, \\ J_I(t) &= \{ j \in \mathbb{Z} : t + a_I \leq j \leq \min(t + b_I, N) \}. \end{aligned} \tag{4}$$

An omitted interval denotes $[0, 1]$. Every interval is relative to the evaluation frame of its operator; the conversion to offsets always uses the full trace length $N$, including for nested operators. The set $J_I(t)$ can be empty.

The literal scores are $Q_c[t] = p_{t,c}$ and $Q_{\neg c}[t] = 1 - p_{t,c}$, with $Q_\top[t] = 1$ and $Q_\bot[t] = 0$. Conjunction and Always multiply their operand scores, whereas disjunction and Eventually sum them. Next selects the score at $t + 1$ if this frame exists and $1 \in [a_I, b_I]$, and is zero otherwise. Until is defined

by

$$Q_{\alpha \mathbf{U}_I \beta}[t] = \sum_{j \in J_I(t)} Q_\beta[j] \prod_{r=t}^{j-1} Q_\alpha[r]. \tag{5}$$

Each summand represents a witness at $j$: the left operand is supported at every preceding frame, and the right operand is supported at $j$. There is no requirement that the right operand fail before $j$. In particular, $Q_{\mathbf{F}_I \beta}[t] = \sum_{j \in J_I(t)} Q_\beta[j] = Q_{\top \mathbf{U}_I \beta}[t]$.

For integer offsets $0 \le a \le b$, write $U_{a,b}(t) = Q_{\alpha \mathbf{U}_{[a,b]} \beta}[t]$. Equation 5 also gives the recurrence

$$U_{a,b}(t) = \begin{cases} Q_\alpha[t] U_{a-1,b-1}(t+1), & a > 0, \\ Q_\beta[t] + Q_\alpha[t] U_{0,b-1}(t+1), & a = 0, \end{cases} \tag{6}$$

where a remainder with no valid witness has score zero. Thus the recurrence inside the permitted window is $B + AR$, where $A$, $B$, and $R$ are the left-operand, right-operand, and remaining Until scores, respectively.

### E.2 Linear-time evaluation

Evaluating Equation 5 independently at every starting frame would repeat work across overlapping windows. Instead, we compute all windows using associative summaries. For fixed child-score arrays $A_t = Q_\alpha[t]$ and $B_t = Q_\beta[t]$, summarize a consecutive block $[i, j]$ by

$$P_{i:j} = \prod_{r=i}^{j} A_r, \qquad R_{i:j} = \sum_{k=i}^{j} B_k \prod_{r=i}^{k-1} A_r. \tag{7}$$

The first component is the left-operand product, and the second is the weighted witness sum starting at the block's first frame. An empty block has summary $(1, 0)$.

**Lemma E.1 (Composition of consecutive blocks)** *Summaries of consecutive blocks combine in temporal order under the associative operation*

$$(P_1, R_1) \circ (P_2, R_2) = (P_1 P_2,\ R_1 + P_1 R_2), \tag{8}$$

*with identity* $(1, 0)$.

*Proof.* The left-operand product over the concatenation is $P_1 P_2$. A witness either lies in the first block, contributing $R_1$, or lies in the second and requires the left operand throughout the first, contributing $P_1 R_2$. Both parenthesizations of three summaries give $(P_1 P_2 P_3, R_1 + P_1 R_2 + P_1 P_2 R_3)$, establishing associativity. The identity follows by substitution. The argument allows zero scores and does not require division. □

For a nonempty witness window, the Until score factors as

$$Q_{\alpha \mathbf{U}_I \beta}[t] = \left( \prod_{r=t}^{t+a_I-1} A_r \right) R_{t+a_I : \min(t+b_I, N)}. \tag{9}$$

The prefix maintains the left operand before the window; the block summary handles all witnesses inside it. Both factors can be obtained for all starting frames by sliding windows. An empty witness window still has score zero.

We compute block summaries in log space. For $(u, v) = (\log P, \log R)$, Equation 8 becomes

$$(u, v) \star (u', v') = \big(u + u', \operatorname{logaddexp}(v, u + v')\big), \tag{10}$$

with identity $(0, -\infty)$. A single frame has summary $(S_\alpha[t], S_\beta[t])$. This avoids direct products of scores and differences of potentially infinite prefix sums.

**Sliding-window aggregation.** Algorithm 2 computes all window aggregates for an associative operation with a constant-size summary. Its queue is implemented with a front stack and a back stack. Every stack entry stores its value and a cached aggregate. The front aggregate $G_F$ runs from the top to the bottom of the front stack; the back aggregate $G_B$ runs from the bottom to the top of the back stack. An empty stack has aggregate $e$. The operations are:

**Algorithm 1** Linear-time interval-aware sum–product scoring in log space

**Input:** Concept scores $p_{t,c} = \hat{y}(z_t, c) \in [0,1]$, $t = 1, \dots, N = T_z$; specification $\varphi$
**Output:** Log score $S_\varphi[1] = \hat{y}(z, \varphi)$ for a fixed sampling step, layer, and head

Convert each normalized interval to $(a_I, b_I)$ as in Equation 4.
WINDOWFOLD$(x, a, b, \circ, e)$ returns all window aggregates (Algorithm 2).
$\star$ is the log-domain block operation in Equation 10.
**for** each distinct subformula $\psi$ in postorder (children before parents) **do**
  **if** $\psi = \mathbf{G}_I \alpha$ **then**
    $S_\psi \leftarrow$ WINDOWFOLD$(S_\alpha, a_I, b_I, +, 0)$
  **else if** $\psi = \mathbf{F}_I \alpha$ **then**
    $S_\psi \leftarrow$ WINDOWFOLD$(S_\alpha, a_I, b_I, \mathrm{logaddexp}, -\infty)$
  **else if** $\psi = \alpha \, \mathbf{U}_I \, \beta$ **then**
    $H \leftarrow$ WINDOWFOLD$(S_\alpha, 0, a_I - 1, +, 0)$ ▷ Left operand before the window
    $x_j \leftarrow (S_\alpha[j], S_\beta[j])$ for $j = 1, \dots, N$
    $W \leftarrow$ WINDOWFOLD$(x, a_I, b_I, \star, (0, -\infty))$
    $S_\psi[t] \leftarrow H[t] + W[t]_2$ for $t = 1, \dots, N$
  **else**
    **for** $t = 1, \dots, N$ **do**

$$S_\psi[t] \leftarrow \begin{cases} 0 & \psi = \top, \\ -\infty & \psi = \bot, \\ \log p_{t,c} & \psi = c, \\ \log(1 - p_{t,c}) & \psi = \neg c, \\ S_\alpha[t] + S_\beta[t] & \psi = \alpha \wedge \beta, \\ \mathrm{logaddexp}(S_\alpha[t], S_\beta[t]) & \psi = \alpha \vee \beta, \\ S_\alpha[t+1] & \psi = \mathbf{X}_I \alpha, \ t < N, \ a_I \le 1 \le b_I, \\ -\infty & \psi = \mathbf{X}_I \alpha \text{ otherwise.} \end{cases}$$

    **end for**
  **end if**
  Cache $S_\psi[1:N]$.
**end for**
**return** $S_\varphi[1]$
$W[t]_2$ denotes the second component; each $S_\psi$, $H$, and $W$ is a length-$N$ array.
$\log 0 = -\infty$; empty windows return the supplied identity $e$.
$\mathrm{logaddexp}(x, y) = \log(e^x + e^y)$, evaluated stably.

- $\mathrm{PushBack}(x)$: push $(x, G_B \circ x)$ onto the back stack.
- $\mathrm{PopFront}()$: if the front stack is empty, transfer all values from the back stack, popping each value $x$ and pushing $(x, x \circ G_F)$ onto the front stack. Then pop the front stack.
- $\mathrm{Aggregate}()$: return $G_F \circ G_B$.

The order of combination is essential because the Until operation is not commutative. Each value enters the back stack once, transfers to the front stack at most once, and leaves the queue at most once. Push and aggregate operations have constant cost; removal has amortized constant cost. No inverse operation is needed, so zero child scores and empty windows are handled directly.

**Proposition E.2 (Correctness and complexity of evaluation)** *Algorithm 1 computes the stated sum–product log score using $O(|\varphi|N)$ arithmetic operations and $O(|\varphi|N)$ storage, including for nested temporal operators. These bounds count operations on scalar scores as constant-time operations.*

*Proof.* For fixed offsets $a \le b$, both endpoints of the clipped window advance monotonically with $t$. Algorithm 2 therefore inserts each sequence element once and removes it at most once. The cached stack aggregates preserve the sequence order by associativity, so each returned value is the requested window aggregate. Including transfers between stacks, the total work for all $N$ windows is $O(N)$. If $a > b$, the algorithm directly returns the identity at every frame in $O(N)$ time.

Always uses addition of log scores with identity zero; Eventually uses log-add-exp with identity $-\infty$. Until uses one window computation for the prefix log product and one for the log-domain block

summaries. Lemma E.1 and Equation 9 show that their combination equals Equation 5, including empty windows. Constants, literals, Boolean operators, and Next take $O(N)$ work by pointwise evaluation. Processing subformulas in postorder proves correctness by induction and evaluates each distinct subformula once, giving $O(|\varphi|N)$ total work. Cached score arrays require $O(|\varphi|N)$ storage, and the working arrays and aggregate queues require $O(N)$ additional space per subformula. The argument depends only on the number of subformulas, not on their nesting depth. □

The computation also has $O(|\varphi|N)$ scalar intermediate operations. At differentiable inputs, reverse-mode differentiation can therefore use the same asymptotic time and storage bounds, with structurally zero branches handled as constants. This is an arithmetic complexity statement; it does not establish a wall-clock speedup over a vectorized quadratic implementation on a GPU.

**Numerical evaluation.** The mathematical definition admits exact zeros and ones. For numerical gradients, literal inputs can be clipped to $[\epsilon_{\mathrm{LS}}, 1-\epsilon_{\mathrm{LS}}]$, separately from the calibration in Equation 2. Composite scores are not clipped to one, and negation is applied only to atoms. Products are computed as sums of logarithms, and sums as log-sum-exp reductions. Empty products are one and empty sums are zero: Always on an empty window has log score zero, while Eventually and Until have log score $-\infty$. On the domain where $Q_\varphi[1] > 0$, the log score is differentiable for literal inputs strictly between zero and one. Structurally zero branches, including log-add-exp with two constant $-\infty$ inputs, should be handled explicitly when differentiating. If the entire score is identically zero, for example because a required witness window is empty, its logarithm provides no finite gradient; the specification must be checked for feasibility on the chosen grid, and that objective should not produce a steering update.

### E.3 Monotonicity and its scope

To distinguish support for a literal from the underlying concept score, introduce independent non-negative inputs $x^+_{t,c}$ and $x^-_{t,c}$ for $c$ and $\neg c$, respectively. Write $\mathcal{Q}_\varphi(x)$ for the score obtained from the same rules with these literal inputs. The score on a concept trace is recovered by substituting $x^+_{t,c} = p_{t,c}$ and $x^-_{t,c} = 1 - p_{t,c}$.

**Algorithm 2** All sliding-window aggregates in linear time

**Input:** Sequence $x_1, \ldots, x_N$; integer offsets $a \geq 0, b$; associative operation $\circ$ with identity $e$
**Output:** $W[t] = x_{t+a} \circ \cdots \circ x_{\min(t+b,N)}$, or $e$ for an empty window

**function** WindowFold($x, a, b, \circ, e$)
  **if** $a > b$ **then**
    **return** $[e, \ldots, e]$ ▷ Includes the empty prefix when $a = 0, b = -1$
  **end if**
  Initialize an empty aggregate queue $D$ with operation $\circ$ and identity $e$.
  $l \leftarrow 1$; $r \leftarrow 0$
  **for** $t = 1, \ldots, N$ **do**
    $L \leftarrow \min(t + a, N + 1)$; $R \leftarrow \min(t + b, N)$
    **while** $r < R$ **do**
      $r \leftarrow r + 1$; $D.\mathrm{PushBack}(x_r)$
    **end while**
    **while** $l < L$ **do**
      $D.\mathrm{PopFront}()$; $l \leftarrow l + 1$
    **end while**
    $W[t] \leftarrow D.\mathrm{Aggregate}()$
  **end for**
  **return** $W$
**end function**

The queue uses two stacks with cached aggregates, as defined in Appendix E.2.
Each element is inserted once and removed at most once; aggregation preserves sequence order.

**Proposition E.3 (Monotonicity in literal support)** *Fix a finite temporal grid and the intervals of a specification $\varphi$ in the stated fragment. The function $\mathcal{Q}_\varphi$ is a polynomial with nonnegative coefficients in the independent literal-support inputs. Consequently, for any componentwise ordered inputs $0 \le x \le x'$,*

$$\mathcal{Q}_\varphi(x) \le \mathcal{Q}_\varphi(x').$$

*The same inequality holds for their logarithms wherever both scores are positive. The claim allows arbitrary nesting of the supported operators and is non-strict.*

*Proof.* We proceed by structural induction on the formula, simultaneously for every starting frame. Constants are nonnegative constant polynomials, and literal scores are individual input variables. Conjunction and Always take finite products of child scores; disjunction and Eventually take finite sums. These operations preserve polynomials with nonnegative coefficients, including their empty-product and empty-sum conventions. Next selects a child score or the constant zero. Finally, each Until score in Equation 5 is a finite sum of products of child scores, and therefore has the same property. This completes the induction. Every monomial with a nonnegative coefficient is nondecreasing in nonnegative inputs, so their sum is nondecreasing as well. The logarithm is increasing on the positive real numbers. □

**Corollary E.4 (Monotonicity in a concept–time coordinate)** *Expand the score at a fixed starting frame into literal-support inputs. If a concept–time pair $(t, c)$ occurs only positively, the specification score is nondecreasing in $p_{t,c}$; if it occurs only negatively, the specification score is nonincreasing in $p_{t,c}$, with all other concept scores fixed. These directions also hold for its logarithm wherever the specification score is positive.*

*Proof.* In the first case, increasing $p_{t,c}$ increases the only relevant literal input $x^+_{t,c}$. In the second, it decreases the only relevant input $x^-_{t,c} = 1 - p_{t,c}$. Apply Proposition E.3. □

Consistency of polarity is required only for the concept–time coordinate in this sufficient condition, not for an entire concept throughout the formula. For example, presence in an early interval and absence in a disjoint later interval have consistent directions at each frame. Both polarities are allowed at the same frame, but then

$$\frac{\partial Q_\varphi[1]}{\partial p_{t,c}} = \frac{\partial \mathcal{Q}_\varphi}{\partial x^+_{t,c}} - \frac{\partial \mathcal{Q}_\varphi}{\partial x^-_{t,c}}, \tag{11}$$

evaluated at the corresponding literal inputs. Each term on the right is nonnegative; their difference need not have a fixed sign. A formula with mixed polarity can nevertheless be monotone after simplification, as the untimed Until example below demonstrates. No probabilistic independence assumption is needed for these algebraic claims.

**Agreement with Boolean satisfaction.** For exact Boolean concept inputs, $Q_\varphi[t] > 0$ if and only if the trace satisfies $\varphi$ at $t$ under Section 2. This follows by structural induction: a product of nonnegative scores is positive exactly when all its factors are positive, and a sum is positive exactly when at least one summand is positive. Applied to Equation 5, these conditions are precisely an admissible right-operand witness and satisfaction of the left operand at all preceding frames. The constants, literals, Next, and empty-window cases follow their Boolean definitions. This statement concerns the unclipped score; it does not prescribe a satisfaction threshold for soft inputs.

Different satisfying traces can have different scores, and logically equivalent formulas need not have identical soft scores. Summing alternatives rewards multiple supported witnesses, and repeating a disjunct changes its weight. We therefore retain the input formula's structure rather than treating Boolean rewrites as score-preserving transformations.

### E.4 Examples and implications for gradients

For the following examples, let $p_t = \hat{y}(z_t, \mathsf{dog}) \in (0, 1)$, and let $u_t = \mathrm{logit}(p_t)$ be its calibrated trace logit. This $u_t$ is distinct from a transformer attention logit. We write $Q = Q_\varphi[1]$ and $S = \log Q$, assuming $Q > 0$.

**Presence and absence in prescribed intervals.** For $\varphi = \mathbf{G}_{I_+}\mathsf{dog} \wedge \mathbf{G}_{I_-}\neg\mathsf{dog}$, let $E = J_{I_+}(1)$ and $A = J_{I_-}(1)$ be disjoint. Then

$$S = \sum_{t\in E} \log p_t + \sum_{t\in A} \log(1-p_t), \qquad \frac{\partial S}{\partial u_t} = \mathbf{1}_{\{t\in E\}}(1-p_t) - \mathbf{1}_{\{t\in A\}}p_t. \tag{12}$$

Gradient ascent thus increases presence evidence in $E$, decreases it in $A$, and leaves unconstrained frames unaffected. A strictly timed witness is likewise obtained by restricting an existential operator to a singleton window, provided the requested time lies on the grid.

**Untimed departure without persistence.** For $\varphi = \mathsf{dog}\,\mathbf{U}\,\neg\mathsf{dog}$, the witness sum telescopes:

$$Q = \sum_{j=1}^{N}(1-p_j)\prod_{r<j} p_r = 1 - \prod_{t=1}^{N} p_t, \qquad \frac{\partial Q}{\partial p_t} = -\prod_{r\neq t} p_r \le 0. \tag{13}$$

Increasing absence evidence at any frame cannot decrease this score, despite its mixed polarity. Under Boolean semantics, the formula is equivalent to $\mathbf{F}\neg\mathsf{dog}$: choose the first absent frame as the Until witness, and all earlier frames necessarily contain the dog. An immediate witness is permitted, and no constraint is imposed after the witness. Nevertheless, our score for $\mathbf{F}\neg\mathsf{dog}$ is $\sum_t(1-p_t)$, illustrating that Boolean equivalence does not imply equality of soft scores.

**Departure followed by persistent absence.** For $\varphi = \mathsf{dog}\,\mathbf{U}_I\,(\mathbf{G}\neg\mathsf{dog})$, define

$$w_j = \underbrace{\prod_{r<j} p_r}_{\text{presence before } j} \underbrace{\prod_{r\ge j}(1-p_r)}_{\text{absence from } j}, \qquad Q = \sum_{j\in J_I(1)} w_j, \qquad \pi_j = \frac{w_j}{Q}. \tag{14}$$

Each term corresponds to one complete departure pattern. Differentiating $\log Q$ gives a weighted combination of the gradients of these patterns:

$$\frac{\partial S}{\partial u_t} = \sum_{j\in J_I(1)} \pi_j \frac{\partial \log w_j}{\partial u_t} = \underbrace{\sum_{\substack{j\in J_I(1)\\ j>t}} \pi_j}_{\text{support for being present at } t} - p_t. \tag{15}$$

Before the earliest admissible departure this derivative is $1-p_t$; at or after the latest admissible departure it is $-p_t$. Within the window its sign depends on the relative support for departure times on either side of $t$. Thus the objective encourages a coherent departure, rather than absence independently at every permitted frame.

An unconditional claim that more absence always helps would contradict the required persistence. For untimed Until, the Boolean trace $[1,1,1,0]$ has score one, whereas changing its second frame to obtain $[1,0,1,0]$ gives score zero: the dog now returns after an earlier disappearance. Also, the formula as written permits absence throughout. To require initial presence, one must explicitly conjoin $\mathsf{dog}$.

**Nested departure and return.** For $\varphi = \mathsf{dog}\,\mathbf{U}_{I_1}(\neg\mathsf{dog}\,\mathbf{U}_{I_2}\mathsf{dog})$, recursive evaluation yields

$$Q = \sum_{j\in J_{I_1}(1)} \sum_{k\in J_{I_2}(j)} \underbrace{\prod_{r<j} p_r}_{\text{before departure}} \underbrace{\prod_{r=j}^{k-1}(1-p_r)}_{\text{absent until return}} \underbrace{p_k}_{\text{return}}. \tag{16}$$

Each admissible pair $(j,k)$ contributes gradients that favor presence before $j$, absence from $j$ through $k-1$, and presence at $k$; it imposes no condition after $k$. The log-score gradient is their weighted combination, with weights proportional to the corresponding summands. Replacing the final $\mathsf{dog}$ by $\mathbf{G}\mathsf{dog}$ replaces $p_k$ by $\prod_{r=k}^{N} p_r$ and requires persistence after return.

The second interval is measured from $j$, not from the start of the video. A lower bound $a_{I_2} \ge 1$ requires at least one absent frame between departure and return. In particular, for $N \ge 2$ and $I_1 = I_2 = [0.5, 1]$, departure must occur at the midpoint and return at the final frame. This pair exists only if the midpoint lies on the sampled grid: writing $d = \lceil (N-1)/2 \rceil$, two successive delays

require $2d \leq N - 1$. Otherwise the witness set is empty. Shorter lower bounds permit multiple departure–return pairs.

**Alternative outcomes.** For $\varphi = \alpha \vee \beta$ with positive operand scores,

$$\nabla S = \frac{Q_\alpha}{Q_\alpha + Q_\beta} \nabla \log Q_\alpha + \frac{Q_\beta}{Q_\alpha + Q_\beta} \nabla \log Q_\beta. \tag{17}$$

The update combines the directions for both alternatives according to their current support. A disjunction alone does not impose exclusivity or select a single branch; when exclusivity is intended, it must be encoded in the specification. Similarly, summing temporal witnesses can favor support at several admissible times rather than only one.

**Transferring gradients to attention biases.** For any differentiable input representation and $Q > 0$, $\nabla S = \nabla Q / Q$. Thus the nonzero gradients of $S$ and $Q$ have the same direction after the infinity-norm normalization used by TSG, before numerical thresholds and clipping. The score need not be bounded by one for this normalization to apply. If the calibration statistics are held fixed while differentiating Equation 2, then $u_{t,c} = (a_{t,c} - \mu_c)/(\tau s_c + \epsilon)$ gives

$$\frac{\partial S}{\partial a_{t,c}} = \frac{1}{\tau s_c + \epsilon} \frac{\partial S}{\partial u_{t,c}}.$$

The positive factor preserves the gradient signs described above. These results concern the scoring function and its input traces. Broadcasting, centering, and clipping these gradients to form attention-logit biases, and propagating the resulting intervention through the transformer, do not inherit a guarantee of score ascent or improved Boolean satisfaction in the decoded video.

## F THE TSG ALGORITHM

Algorithm 3 presents TSG with online attention steering. The guided steps are the first $N_{\text{tsg}}$ sampling updates, i.e., $\mathcal{K} = \{N - N_{\text{tsg}} + 1, \ldots, N\}$. The loop over selected attention layers takes place within a single additional forward pass: normal transformer operations proceed between interventions, and later layers use the representations produced by earlier steered layers. Differentiation is enabled only for the detached traces, their calibration, and the specification score.

**Algorithm 3** Temporal Specification Guidance with online attention steering

**Input:** Prompt embeddings $e_p, e_{\text{ref}}$, specification $\varphi$, transformer $T_\theta$, decoder $D$
**Input:** Sampling steps $N$, guided steps $\mathcal{K}$, selected layer–head pairs $\mathcal{H}$
**Input:** CFG scale $\lambda_{\text{cfg}}$, TSG norm ratio $\rho$, bias step size $\eta$, bounds $[b_{\min}, b_{\max}]$
**Input:** Calibration temperature $\tau$ and stabilizer $\epsilon$
**Output:** Generated video $v$

Map concepts in $\varphi$ to prompt-token sets $\mathcal{P}_c$
Initialize $z_N \sim \mathcal{N}(0, I)$; convert intervals in $\varphi$ to the latent temporal grid
**for** $k = N, N-1, \ldots, 1$ **do**
    $d_{k,c} \leftarrow T_\theta(z_k, e_p, k)$ without steering
    $d_{k,u} \leftarrow T_\theta(z_k, e_{\text{ref}}, k)$ without steering
    $d_{k,\text{cfg}} \leftarrow d_{k,u} + \lambda_{\text{cfg}}(d_{k,c} - d_{k,u})$
    $d_k \leftarrow d_{k,\text{cfg}}$
    **if** $k \in \mathcal{K}$ **then**
        Begin an additional conditional evaluation of $T_\theta(z_k, e_p, k)$ without a transformer gradient graph
        **for** selected attention layers $\ell$, in forward order **do**
            Advance this guided pass to layer $\ell$, using outputs of the preceding layers
            **for all** heads $h$ with $(\ell, h) \in \mathcal{H}$ **do**
                Compute current attention $A^{\ell,h}$ before the local intervention
                Form raw traces $a^{\ell,h}$ using Equation 2; detach and enable trace gradients
                Calibrate to $\hat{y}^{\ell,h}$ using Equation 2, holding statistics fixed
                $S_{\ell,h} \leftarrow \text{Score}(\hat{y}^{\ell,h}, \varphi)$
                **for all** concepts $c$ in $\varphi$ **do**
                    $g_{\cdot,c} \leftarrow \nabla_{a_{\cdot,c}} S_{\ell,h}$
                    $b_{\cdot,c} \leftarrow \eta g_{\cdot,c} / \|g_{\cdot,c}\|_\infty$; use zero for negligible gradients
                **end for**
                Broadcast $b$ to query–text pairs; center across text keys and clip to obtain $B^{\ell,h}$
                Detach $B^{\ell,h}$ and apply it at this head using Equation 3
            **end for**
            Complete layer $\ell$ with the biased attention; release local scoring buffers
        **end for**
        $d_{k,\varphi} \leftarrow$ output of the completed guided forward pass
        $r_k \leftarrow d_{k,\varphi} - d_{k,c}$; compute $\lambda_{k,\text{tsg}}$ using Equation 1
        $d_k \leftarrow d_{k,\text{cfg}} + \lambda_{k,\text{tsg}} r_k$
    **end if**
    $z_{k-1} \leftarrow \text{Step}_k(z_k, d_k)$
**end for**
**return** $D(z_0)$