Confidence Scores for Temporal Properties over Sequences of Predictions

Avishree Khare[1], **Hideki Okamoto**[2], **Bardh Hoxha**[2], **Georgios Fainekos**[2], and **Rajeev Alur**[1]

[1]Department of Computer Science, University of Pennsylvania
[2]Toyota Motor North America, R&D
[1]{akhare,alur}@seas.upenn.edu
[2]{hideki.okamoto,bardh.hoxha,georgios.fainekos}@toyota.com

Abstract

Detection models such as YOLO and Grounding DINO can be used to detect local properties such as objects ("car") in individual frames of videos. The likelihood of these detections is indicated by confidence scores in $[0, 1]$. How can these confidence scores be lifted to temporal properties over sequences (e.g., "a car is detected at all times in the video"or "a traffic light is eventually detected")? In this work, we study this problem of assigning Confidence Scores for TempOral Properties (CSTOPs) over sequences, given confidence score predictors for local properties. These scores can be useful for several downstream applications such as query matching, ranking and fine-tuning. We argue that Temporal Logic provides a natural language for expressing diverse temporal properties of interest. We then propose a scoring function called LogCSTOP for defining CSTOPs that can be computed in polynomial time with respect to the lengths of the sequence and temporal property, while existing methods require exponential space and time. We introduce the CSTOP-RealTLV-video benchmark with video sequences from the RealTLV dataset and ground-truth labels for temporal properties from 15 diverse templates. Finally, we show that LogCSTOP outperforms Large Vision Language Models, such as `LongVA-7B`, and an existing neurosymbolic baseline on query matching over videos by more than 16% in terms of balanced accuracy .

1 Introduction

Consider an autonomous vehicle driving in the city, providing real-time perception data as videos to a surveillance system. This system might then be interested in automatically identifying whether the scene satisfies certain properties. Examples of critical *temporal* properties include "the vehicle *always* remains in a given lane" or "no pedestrian is detected on the crossing *until* the traffic light turns red". While neural detection models such as YOLO [20] can detect local properties such as objects ("car") in individual video frames, these detections are accompanied by the models' uncertainty about the predictions in the form of *confidence scores*. Hence, YOLO would detect how likely a car is in a frame with a confidence score between 0 and 1. How can these confidence scores then be lifted to the temporal properties discussed above? In other words,

Given a temporal property and (potentially noisy) confidence score predictors for local properties, what is the confidence score for a data sequence expressing the temporal property?

These Confidence Scores over TempOral Properties (CSTOPs) can be useful for several downstream applications on unstructured data sequences. As in the examples discussed above, a **query matching** system could check if these scores are over a given threshold to decide if a video expresses the

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

Figure 1: LogCSTOPs for three videos from the NuScenes dataset w.r.t the query *"Is there a person in all frames of this video?"*. Video 2 with occluded persons is assigned a lower score than video 1 (where a person is visible in all frames), and higher score than video 3 (where there are frames with no persons). These scores can be used for ranking and query matching with the adaptive threshold. YOLOv8x is used here to detect objects in individual frames of the videos.

temporal property. Furthermore, a **search / ranking** system could compare / rank multiple videos against a temporal query using these scores to provide the top-k most relevant results.

This problem of score assignment was introduced by [23] for detecting temporal events in videos, where the events were represented as properties in Linear Temporal Logic (LTL) [19]. LTL, with temporal operators such as "Always" ($\Box$) and "Until" ($\mathcal{U}$), can be used to represent several useful temporal properties of interest. For example, the property "vehicle always remains in lane" can be written as $\Box car\_in\_lane$ where the local property car_in_lane can be detected using a neural model (e.g., CLIP). Given a video with T frames and local properties C, [23] first constructs a probabilistic automaton on all possible conjunctions of local properties, C, and all frames, resulting in a state space with $T * 2^{|C|}$ states. A probabilistic model checker, STORM [11], is then used to assign a probability to the automaton satisfying an LTL property. Irrespective of the LTL property φ, this requires $\mathcal{O}(2^{|C|} * T)$ space and time. A video matches a query if the score is greater than 0.5.

In this work, we propose a novel scoring function, called LogCSTOP, that can assign a confidence score to a data sequence satisfying a temporal property in LTL. LogCSTOP is inspired by quantitative semantics for LTL [9] and can assign a score for a video of length T satisfying property φ in $\mathcal{O}(|\varphi| * T)$ time and space. LogCSTOP hence offers a significant improvement in space and time complexity over [23], since it does not need to construct an expensive probablistic automaton. Furthermore, LogCSTOP inherently handles noisy predictions via local smoothing. We also argue that the constant 0.5 threshold for query matching used by [23] is not optimal for long sequences and propose an *adaptive threshold* that depends on the length of the sequence and temporal property. Figure 1 demonstrates how these scores can be used for ranking and query matching.

In order to empirically demonstrate the usefulness of LogCSTOP, we evaluate it on the downstream application of query matching over videos. Since the implementation of [23] is not publicly available, we compare LogCSTOP with NSVS-TL [6]. NSVS-TL [6] also constructs the same automaton as [23] and uses an off-the-shelf model checker for detecting events of interest in videos. However, unlike [23], NSVS-TL uses Probabilistic Computational Tree Logic (PCTL) and only evaluates whether a video matches a query, limiting its use to query matching. Owing to their success on several video-text tasks, we also include Large Vision Language Models (LVLMs), such as `Video-LLava-7B` [14] and `LongVA-7B` [24] as baselines.

To compare these methods, we introduce the CSTOP-RealTLV-video benchmark that compiles video sequences from the Real TLV [6] dataset and provides ground-truth labels for temporal properties

from 15 diverse templates over objects in videos. We find that LogCSTOP with object detection scores from YOLOv8 outperfoms LVLMs and NSVS-TL by $> 16\%$ in terms of balanced accuracy on CSTOP-RealTLV-video. We also evaluate how changes to different components of LogCSTOP affect the performance on query matching and find that local smoothing, and adaptive thresholding improve query matching results.

To summarize, our contributions are as follows:

- We study the problem of assigning scores to temporal properties over sequences, given noisy predictors for local properties C. We propose LogCSTOP, a scoring function inspired by quantitative semantics for Temporal Logic, that can assign a score to a video of length T satisfying a temporal property φ in $\mathcal{O}(|\varphi| * T)$ space and time. This is a significant improvement over existing work that requires $\mathcal{O}(2^{|C|} * T)$ space and time.
- We introduce the CSTOP-RealTLV-video benchmark that compiles video sequences from the Real TLV dataset , with temporal properties from 15 diverse templates over objects and emotions respectively.
- We evaluate LogCSTOP with simple trained predictors on a downstream task, namely query matching on the CSTOP-RealTLV-video benchmark. We find that LogCSTOP with YOLOv8 outperforms LVLMs and a PCTL-based state-of-the-art method for event detection, NSVS-TL, on videos by more than 16% in terms of balanced accuracy.

2 Representing Temporal Properties over Sequences of Predictions

Let $X = [x_0, x_1, \ldots, x_T]$ denote a sequence of data items of length T, where x_t denotes the data item at timestep $0 \leq t \leq T$. Further, let C denote a finite set of local properties of interest. We assume that there exists a true labeling function $y : C \times T \mapsto \{0, 1\}$ such that $y(c, t) = 1$ if the property $c \in C$ is expressed by x_t, and $y(c, t) = 0$ otherwise. In the example of autonomous driving scenarios, X could correspond to a video with each x_t corresponding to the frame at timestep t and C could be objects of interest such as "car" and "pedestrian". In this case, $y(c, t) = 1$ would indicate that object c is present in frame x_t.

In this work, we are interested in temporal compositions of these local properties. For example, given the true labels for the object "car" in individual frames, what is the label for a car being present in all frames or alternatively any frame? Additionally, given the true label for "pedestrian", what is the label for a car being present in all frames until a pedestrian is detected?

We find that Linear Temporal Logic (LTL), introduced and widely used for formal specification and verification of reactive systems [19], provides a suitable *language* for expressing such temporal properties. Formally, a temporal property φ over local properties C can be expressed in LTL as follows:

$$\varphi \coloneqq \top \mid \bot \mid c \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \bigcirc\varphi \mid \Box\varphi \mid \varphi_1 \,\mathcal{U}\, \varphi_2$$

where, $c \in C$ is a local property and φ_1, φ_2 are temporal properties. $\neg, \wedge, \vee$ are the logical *not*, *and*, and *or* operators respectively. $\bigcirc$, $\Box$ and $\mathcal{U}$ are temporal operators *Next*, *Always*, and *Until* respectively. Other temporal operators such as "Eventually φ" ($\Diamond\varphi$) can then be derived as $\neg\Box\neg\varphi$.

This language can now be used to represent properties from the previous examples. For instance, the temporal properties for a car being present in all frames and any frame in a video can be represented as $\varphi = \Box\, car$ and $\varphi = \Diamond\, car$ respectively. Furthermore, the property that a car is present in all frames until a pedestrian is present can be represented as $\varphi = car\, \mathcal{U}\, pedestrian$.

The ground truth labeling function $y(c, t)$ for local properties can be lifted to $y(\varphi, t)$, meaning that the sequence X expresses temporal property φ starting at timestep t, using the standard semantics for LTL over finite sequences [7] as follows: for $0 \leq t \leq T$,

- $y(\top, t) = 1$ and $y(\bot, t) = 0$
- $y(c, t) = 1$ iff c is expressed by x_t
- $y(\neg\varphi, t) = 1$ iff $y(\varphi, t) = 0$
- $y(\varphi_1 \wedge \varphi_2, t) = 1$ iff $y(\varphi_1, t) = 1$ and $y(\varphi_2, t) = 1$
- $y(\varphi_1 \vee \varphi_2, t) = 1$ iff $y(\varphi_1, t) = 1$ or $y(\varphi_2, t) = 1$

- $y(\bigcirc\varphi, t) = 1$ iff $t < T$ and $y(\varphi, t+1) = 1$
- $y(\Box\varphi, t) = 1$ iff $y(\varphi, t') = 1$ for all $t \leq t' \leq T$
- $y(\varphi_1 \,\mathcal{U}\, \varphi_2, t) = 1$ iff there exists a $t \leq t' \leq T$ such that $y(\varphi_2, t') = 1$ and $y(\varphi_1, t'') = 1$ for all $t \leq t'' < t'$

Given the true labeling functions for local properties $y(c, \cdot)$, the standard semantics can perfectly determine if a sequence X expresses a temporal property φ (if and only if $y(\varphi, 0) = 1$). In practice, however, we do not have access to these true labeling functions. We assume that noisy predictors can be used instead to provide *confidence scores* $\hat{y} : C \times T \mapsto [a, b]$ for the label being 1 . The estimate for true label $y(c, t)$ can then be computed as $\tilde{y}(c, t) = \hat{y}(c, t) > \tau$ for some threshold τ. Most neural models, including object detection models such as YOLO, provide confidence scores for predictions in the range $[0, 1]$. The object c with score $\hat{y}(c, t)$ is said to be detected, i.e., $\tilde{y}(c, t) = 1$ if $\hat{y}(c, t) > \tau$, where τ usually is 0.5. Note that it is easy to normalize scores in any range $[a, b]$: for example, scores in the $[0, 1]$ range can be shifted to the $[-\infty, 0]$ range by using the $\log$ operator (and vice versa, using the e operator), the threshold τ will also have to be scaled accordingly. The accuracy of these predictors then just measures how well $\tilde{y}(c, \cdot)$ estimates $y(c, \cdot)$.

We formally present the problem of assigning Confidence Scores for TempOral Properties (CSTOP) as follows: Given confidence score predictors for local properties $\{\hat{y}(c, t) : c \in C, 0 \leq t \leq T\}$ and a temporal property φ defined over the local properties in C, how can the confidence score for φ at the time step $0 \leq t' \leq T$, denoted $\hat{y}(\varphi, t')$, be assigned?

3 LogCSTOP: An Algorithm for Computing CSTOP

While the standard semantics of LTL assigns a Boolean truth value to each temporal property, quantitative extensions have been proposed to assign numerical values to temporal properties that are intended to capture how well a sequence satisfies the property [8]. Informed by this literature on quantitative semantics, we propose a scoring function *LogCSTOP* $\hat{y} : \varphi \times T \times W \mapsto [-\infty, 0]$ in Algorithm 1 that recursively computes a confidence score for temporal property φ in time polynomial in T and length of the temporal property $|\varphi|$. $\hat{y}(\varphi, \cdot)$ assumes that the scores for local properties are also given in range $[-\infty, 0]$, i.e., $\hat{y}(c, \cdot) \in [-\infty, 0]$. Whenever needed, we normalize the confidence scores for local properties $\hat{y}(c, \cdot)$ to be in the $[0, 1]$ range using $e^{\hat{y}(c,\cdot)}$. $1 \leq w \leq T$ denotes a window used for downsampling and smoothing predictions and we discuss this later.

Algorithm 1 LogCSTOP $\hat{y}(\varphi, t, w)$

Require: Temporal property φ, current timestep t, dowsampling-smoothing window $1 \leq w \leq T$
Ensure: $\hat{y}(\varphi, t, w)$

function $S(\varphi, t, w)$
  if $t > T$ **then return** $-\infty$
  else if $\varphi = \top$ **then**
    return 0
  else if $\varphi = \bot$ **then**
    return $-\infty$
  else if $\varphi = c$ **then**
    return $\log(average_{t' \in [t, t+w)} e^{\hat{y}(c, t'))})$ ▷ Smooth scores in window $[t, t+w]$
  else if $\varphi = \neg\varphi'$ **then**
    return $\log(1 - e^{\hat{y}(\varphi, t, w)})$
  else if $\varphi = \varphi_1 \wedge \varphi_2$ **then**
    return $\hat{y}(\varphi_1, t, w) + \hat{y}(\varphi_2, t, w)$
  else if $\varphi = \varphi_1 \vee \varphi_2$ **then**
    return $\hat{y}(\neg(\neg\varphi_1 \wedge \neg\varphi_2), t, w)$
  else if $\varphi = \bigcirc \varphi'$ **then**
    return $\hat{y}(\varphi', t + w, w)$ ▷ Shift the current timestep from t to $t + w$
  else if $\varphi = \Box \varphi'$ **then**
    return $\hat{y}(\varphi' \wedge \bigcirc \Box \varphi', t, w)$
  else if $\varphi = \varphi_1 \,\mathcal{U}\, \varphi_2$ **then**
    return $\hat{y}(\varphi_2 \vee (\varphi_1 \wedge \neg\varphi_2 \wedge \bigcirc (\varphi_1 \,\mathcal{U}\, \varphi_2)), t, w)$
  end if
end function

Downsampling and average smoothing of confidence scores of local properties. Firstly, LogCSTOP downsamples the sequence of length T to contiguous blocks of length w. The confidence scores for any local property in each block starting at t, $\hat{y}(c, t' \in [t, t+w])$ are first averaged after being normalized to the $[0, 1]$ range. The confidence score for each block is then the $\log$ of this averaged $[0, 1]$-normalized score. This series of downsampling, normalizing and smoothing operations starting at timestep t for a window w can be seen on line 8. For example, given a sequence of log scores for object "car",

$$\hat{y}(car, t \in [0, 5]) = [\log(0.9), \log(0.1), \log(0.9), \log(0.9), \log(0.9), \log(0.9)]$$

and $w = 3$, the LogCSTOP first downsamples to 2 blocks,

$$[[\log(0.9), \log(0.1), \log(0.9)], [\log(0.9), \log(0.9), \log(0.9)]]$$

before averaging by block,

$$[\log((0.9 + 0.1 + 0.9)/3), \log((0.9 + 0.9 + 0.9)/3)] = [\log(0.63), \log(0.9)]$$

In this example, note that the score $\log(0.1)$ is likely incorrect since the car cannot momentarily disappear. Downsampling and then smoothing reduce the impact of this incorrect local prediction hence essentially capturing the property that confidence scores cannot drastically change in a local window. Note that this is done online for each successive block and the shifting for temporal operators is handled by the Next operator (line 16).

Handling of logical operators ($\neg, \wedge, \vee$). The LogCSTOP for $\neg\varphi$ at timestep t is intuitively high when the score for φ is low (line 10). For example, given a high score $\hat{y}(car, t, w) = \log(0.9)$, the score for "not car" $\hat{y}(\neg car, t, w) = \log(1 - 0.9) = \log(0.1)$ is low.

The LogCSTOP for $\varphi_1 \wedge \varphi_2$ at timestep t is high only when the scores for both φ_1 and φ_2 are high (line 12). For example, given high scores $\hat{y}(car, t, w) = \log(0.9)$ and $\hat{y}(pedestrian, t, w) = 0.9$, the score for "car and pedestrian" $\hat{y}(car \wedge pedestrian, t, w) = \log(0.9) + \log(0.9) = \log(0.81)$. However, if either of the scores are low, e.g., if $\hat{y}(pedestrian, t, w) = 0.1$, the score drops significantly to $\hat{y}(car \wedge pedestrian, t, w) = \log(0.9) + \log(0.1) = \log(0.09)$. Inspired by DeMorgan's law, the LogCSTOP for $\varphi_1 \vee \varphi_2$ is simply the score for equivalent property $\neg(\neg\varphi_1 \wedge \neg\varphi_2)$ (line 14). This is intuitively only low when both the scores are low.

Note that these operators are defined not only over local properties c as in the examples above, but over any temporal property $\varphi, \varphi_1, \varphi_2$. Hence, for temporal property φ = "(car or truck) and (not pedestrian)", the score is high only if it is high for both φ_1 = "(car or truck)" and φ_2 = "(not pedestrian)". The scores for φ_1, φ_2 can be recursively computed.

Handling of temporal operators ($\bigcirc, \Box, \mathcal{U}$). As discussed above, the property Next φ ($\bigcirc\varphi$) evaluates whether φ is expressed starting at the next block $t + w$ (line 16). When $w = 1$, this represents the standard Next operator. The property Always φ ($\Box\varphi$) is interpreted here as a "temporal and" operator over the sequence (line 18). Hence, $\Box\varphi$ can be equivalently written as $\varphi \wedge \bigcirc\Box\varphi$: the property φ is expressed by x_t and always after by $[x_{t+w}, \ldots]$ ($\bigcirc\Box\varphi$). Similar to the logical $\wedge$ operator, the score is high only if it is high for all timesteps of the sequence. The computation in $\log$ space is beneficial to prevent any underflow here with fixed precision.

The property φ_1 Until φ_2 ($\varphi = \varphi_1 \,\mathcal{U}\, \varphi_2$) can be equivalently written as $\varphi = \varphi_2 \vee (\neg\varphi_2 \wedge \varphi_1 \wedge \bigcirc(\varphi_1 \,\mathcal{U}\, \varphi_2))$ (line 20). This informally translates to evaluating whether either (1) φ_2 is expressed by x_t, or (2) φ_1 is expressed instead and $\varphi = \varphi_1 \,\mathcal{U}\, \varphi_2$ is expressed by $[x_{t+w}, \ldots]$.

An adaptive threshold for LogCSTOP. As discussed in Section 2, the usefulness of a local property predictor $\hat{y}(c, \cdot)$ can be determined by how accurately it predicts the true label $y(c, \cdot)$ as $\tilde{y}(c, \cdot) = \hat{y}(c, \cdot) > \tau$ for some threshold τ. Similarly, the usefulness of a scoring function for temporal property φ is determined by how accurately it predicts $y(\varphi, \cdot)$ (from the standard LTL semantics) as $\tilde{y}(\varphi, \cdot) = \hat{y}(\varphi, \cdot) > \tau$ for some τ.

A natural first choice for such a threshold for LogCSTOP is the constant $\tau = \log 0.5$. This, however, does not scale with the length of the sequence. For instance, with $w = 1$, given a sequence of log scores for "car",

$$\hat{y}(car, t \in [0, 5]) = [\log(0.9), \log(0.9), \log(0.9), \log(0.9), \log(0.9), \log(0.9)]$$

the LogCSTOP for temporal property "Always car" is

$$\hat{y}(\Box car, t = 0, w = 1) = \log(0.9^6) > \log(0.5).$$

However, when another data item with the same high score $\log(0.9)$ is added, the score drops to

$$\hat{y}(\Box car, t = 0, w = 1) = \log(0.9^7) < \log(0.5).$$

We would ideally like both these sequences to match the query $\Box\, car$ but only the first one does with the $\log 0.5$ threshold. We hence define a length and query-adaptive threshold $\tau : \varphi \times T \mapsto [-\infty, 0]$ for LogCSTOP as follows: Let $y_{0.5}(c, t) = 0.5$ for all c, t and $\hat{y}_{0.5}(\varphi, t, w)$ be the assigned LogCSTOP. Then $\tau(\varphi, t) = \min\{\log(0.5), \hat{y}_{0.5}(\varphi, t, w)\}$. The estimate of $y(\varphi, t)$ is then $\tilde{y}(\varphi, t, w) = \hat{y}(\varphi, t, w) > \tau(\varphi, t)$. In other words, a data sequence expresses a temporal property if the LogCSTOP is higher than both random chance $\log(0.5)$ and the score assigned using random chance predictors for local properties.

Complexity analysis for LogCSTOP. The computational complexity of Algorithm 1, for a temporal property φ with length $|\varphi|$ and a sequence of predictions of length / time horizon T, is $\mathcal{O}(|\varphi| * T)$. This assumes $\mathcal{O}(|\varphi| * T)$ space for caching scores for each sub-formula over the sequence. The computational complexity follows from the observation that a formula / temporal property can be written as a parse tree with nodes corresponding to sub-formulae and leaves corresponding to local properties. This tree can then be processed from leaves to root, at each step caching the sequence scores for the sub-formula corresponding to the node being processed. The score for any sub-formula rooted at a node can be computed in $\mathcal{O}(T)$, given the sequences of scores for its children. Since the tree has $|\varphi|$ nodes in total, this leads to a computational complexity of $\mathcal{O}(|\varphi| * T)$ for the root formula.

3.1 Comparison with other Quantitative Semantics for Temporal Logic.

Quantitative semantics for Signal Temporal Logic [8] define a *robustness* measure, i.e., a measure of how well a signal satisfies or violates a given formula. If the sequences of confidence score predictions over local properties are viewed as signals, these semantics offer an alternate solution to the CSTOP problem where the robustness represents the confidence score for a temporal property.

We argue that LogCSTOP offers advantages over such semantics in the context of the CSTOP problem. This is primarily because the traditional robustness measure is defined using $\max$ and $\min$ functions over temporal and logical formulae. The measure, hence, only reflects the most violating or most satisfying timestep in the sequence. For example, consider assigning confidence scores to the property "Always car" in two different scenarios:

$$\hat{y}_1(car, t \in [0, 2]) = [\log(0.9), \log(0.9), \log(0.1)]$$

$$\hat{y}_2(car, t \in [0, 2]) = [\log(0.1), \log(0.1), \log(0.1)]$$

Ideally, the confidence score for "Always car" should follow the order: $\hat{y}_1(\Box car, \cdot) > \hat{y}_2(\Box car, \cdot)$. The standard STL semantics, however, would assign the same robustness to both sequences for any $\tau > 0.1$ since the most violating score is $\log(0.1)$ in either case. This makes the robustness measure unsuitable for downstream applications that require such ordering: for example, ranking / search. Moreover, the use of the $\min$ and $\max$ operators result in non context robustness measures, making it a poor choice for applications such as fine-tuning local predictors using these scores for feedback. We provide more details about the standard robustness measure and how it is computed for different temporal properties in Appendix A, with more examples demonstrating differences with LogCSTOP.

4 Experimental Evaluation

We evaluate LogCSTOP on the downstream application of query matching with temporal properties (TP-query matching) over videos and speech and compare it with other methods for that task. Concretely, the TP-query matching task on video involves predicting whether a given temporal query matches, or is expressed by, a video. For instance, the query "always car" matches any video where the car is present in all frames.

Temporal properties as queries. We use the following criteria for selecting temporal properties (of increasing difficulty) as queries for evaluation: (1) number of local properties: 1-3, (2) number of nested operators in the temporal property: 1-3, and (3) number of temporal operators (1-2). Given these criteria, we choose 15 temporal property templates from 5 broad categories, in the order of increasing difficulty of operator selection and nesting:

1. **Simple temporal operators:** Eventually p1, Always p1, p1 Until p2.
2. **Boolean over temporal operators:** Always p1 and Eventually p2, Always p1 or Eventually p2.
3. **Temporal over boolean operators:** (Not p1) Until p2, p1 Until (Not p2), Always (p1 and p2), (p1 and p2) Until p3.
4. **Temporal over temporal operators:** p1 Until Always p2, Eventually Always p1, Always Eventually p1.
5. **Temporal operators over boolean and temporal operators:** (Not p1) Until Eventually p2, (Not p1) Until Always p2, (p1 and p2) Until Eventually p3.

4.1 The CSTOP-RealTLV-video Benchmark for TP-query matching over videos.

There are no existing benchmarks that evaluates TP-query matching on video sequences with the breadth of properties discussed above. We introduce the CSTOP-RealTLV-video benchmark that provides ground-truth labels for different temporal properties over video sequences collected from the Real TLV dataset [6] . The Real TLV dataset consists of a subset of videos from NuScenes [3] and Waymo [21] driving datasets with frame-level annotations for various objects / entities present in each scene (for example, "car", "truck", etc). The dataset can be used for query matching with temporal properties over objects in videos ("car until pedestrian", for example) . Note that the Real TLV dataset provides ground-truth labels for two temporal property templates: "p1 Until p2" and "(p1 and p2) Until p3" that are also included in our list of 15 templates discussed above.

For any temporal property template (for example, "p1 Until p2") and samples from this dataset, we identify matching and non-matching sequences of desired length as follows: for every sample, we first identify candidates for local properties in the template (p1, p2, etc.) as the set of all ground-truth objects in the sequence. We then use the standard LTL semantics over the frame-level ground-truth labels to collect matching and non-matching subsequences of the desired length. This creates a TP-query matching dataset for an arbitrary set of temporal properties as long as these properties are sufficiently expressed by sequences from the underlying dataset. Moreover, this pipeline is agnostic to the choice of the dataset since it only requires sequences of ground-truth labels for local properties. We use this pipeline to create the CSTOP-RealTLV-video dataset. This dataset consists of 7468 samples, 3750 matching and 3718 non-matching, with video sequences of lengths $\{10, 20, 30, 40, 50\}$. For each target length, this dataset contains approximately 100 samples corresponding to each of the 15 property templates.

4.2 Methods.

We use TP-query matching as an application and compare LogCSTOP against methods for this task. Briefly, these methods include:

LogCSTOP with simple trained neural predictors: We use YOLOv8 [12], Grounding DINO [15], and OWLv2 [18], to get confidence scores for object detection in video frames . Since the scores are in the $[0, 1]$ range, we normalize them in the $[-\infty, 0]$ range using the $\log$ operation, as required by LogCSTOP. A video matches query φ if LogCSTOP $\hat{y}(\varphi, 0, w) > \tau(\varphi, 0)$ (and vice versa for non-matching examples). The estimates are evaluated against ground-truth labels $y(\varphi, 0)$. The window w is selected as follows: $w = 2$ for $T < 20$ and $w = 5$ otherwise.

Large Vision Language Models (LVLMs). We evaluate two popular LVLMs on query matching for videos: `Video-LLava-7B` [14] and `LongVA-7B` [24]. For the "always car" example, we provide the models with the video sequence and a text prompt "Is a car detected in all frames of this video?". The response is considered correct if the model responds with "Yes" or "No" for matching and non-matching samples respectively. `Video-LLava-7B` supports a context window of 4096 tokens while `LongVA-7B` can handle up to 2000 frames. We set the maximum tokens to generate to 60 and 1024 respectively and use a temperature of 0.1 and standard values for the other parameters.

NSVS-TL [6]. Proposed for event detection in videos, NSVS-TL [6] uses the PCTL-based model checker STORM [11] to identify video frame subsequences where a certain event is detected. NSVS-TL reports state-of-the-art performance on detecting temporal events in videos, surpassing large language models such as GPT-4. For our task, we specify the target query in PCTL ("always car" is $P > 0.5[G \text{ "}car\text{" }]$) and the response is considered correct if NSVS-TL returns / does not return the entire video sequence as output for a matching / non-matching query respectively.

Table 1: LogCSTOP reports the best overall balanced accuracy on the CSTOP-RealTLV-video dataset, outperforming LVLMs and NSVS-TL. Moreover, it reports the best performance on all queries. The dataset contains 3750 matching and 3718 non-matching sequences of lengths $\{10, 20, 30, 40, 50\}$. The best performing method is highlighted in **bold** and the second best is underlined.

Query	NSVS-TL	Video-LLaVA-7B	LongVA-7B	LogCSTOP		
				OWLv2	GroundingDINO	YOLOv8
Eventually p1	0.62	0.5	0.71	0.65	0.62	**0.84**
Always p1	0.68	0.51	0.72	0.54	0.76	**0.81**
p1 Until p2	0.7	0.5	0.61	0.63	0.73	**0.8**
Always p1 and Eventually p2	0.55	0.5	0.65	0.51	0.65	**0.72**
Always p1 or Eventually p2	0.71	0.5	0.61	0.68	0.61	**0.78**
Not p1 Until p2	0.5	0.5	0.68	0.71	0.77	**0.85**
p1 Until Not p2	0.5	0.5	0.64	0.53	0.66	**0.78**
Always (p1 and p2)	0.55	0.51	0.62	0.51	0.65	**0.73**
p1 and p2 Until p3	0.5	0.5	0.55	0.56	0.71	**0.78**
p1 Until Always p2	0.64	0.5	0.57	0.51	0.62	**0.75**
Eventually Always p1	0.64	0.5	0.62	0.7	0.76	**0.79**
Always Eventually p1	0.64	0.5	0.69	0.55	0.72	**0.76**
(Not p1) Until Eventually p2	0.5	0.5	0.61	0.59	0.59	**0.81**
(Not p1) Until Always p2	0.5	0.5	0.68	0.66	0.71	**0.78**
(p1 and p2) Until Eventually p3	0.5	0.5	0.55	0.65	0.62	**0.84**
Overall	0.58	0.5	0.63	0.6	0.68	**0.79**

4.3 Results for TP-query matching on the CSTOP benchmark.

R1: LogCSTOP outperforms other methods on TP query matching for videos. Table 1 reports the overall balanced accuracies of different methods on CSTOP-RealTLV-video . LogCSTOP, with detections from YOLOv8, reports the best performance with a balanced accuracy of 79%, outperforming `Video-LLava-7B` and `LongVA-7B` by 29% and 16% respectively, and NSVS-TL by 21%. Moreover, LogCSTOP with other object detection models, such as Grounding DINO and OWLv2, also report accuracies better or at par with other baselines. The differences arise from the varying object detection performances of these methods. The low performances on the CSTOP benchmark suggest that understanding of temporal queries is still an open problem for LVLMs. Moreover, the higher accuracy of LogCSTOP with much smaller neural models (YOLO) suggests that using well-defined logics for reasoning is beneficial.

R2: LogCSTOP performs well on all queries, across sequences of different lengths. Table 1 also shows the query-wise breakdown on the CSTOP-RealTLV-video dataset. LogCSTOP with YOLOv8 performs the best amongst all methods, reporting accuracies over 70% on all 15 queries. With detections from Grounding DINO, LogCSTOP reports the second best performance on $9/15$ queries. `LongVA-7B` performs the second best on 3 queries while `Video-LLava-7B` performs poorly with a 50% accuracy on all queries. We find that this can be attributed to only "Yes" responses from the model. NSVS-TL also reports worse performance on queries from two categories: boolean expressions over temporal operators, and temporal operators over boolean and temporal operators.

In Figure 2, we show the breakdown of TP-query matching performances of different methods by video length. Interestingly, no method reports any drop in performance as the length of the sequence increases. LogCSTOP with YOLO and Grounding DINO consistently outperform or match other methods across lengths. LogCSTOP with YOLO reports accuracies $> 75\%$ on $4/5$ lengths while all other methods, namely NSVS-TL and the LVLMs, report accuracies $< 70\%$ across lengths.

R3: The design choices for LogCSTOP are empirically optimal. Finally, we evaluate how the various design choices for LogCSTOP affect the performance. The choices include the aggregation method used (LogCSTOP vs. other quantitative semantics for LTL), the effect of local smoothing, and the adaptive thresholding for query matching. We find that using a different aggregation method, namely the standard STL semantics discussed in Section 3.1, reduces the accuracy of query matching with YOLOv8 detections by 2%. A 2% drop is also observed if the local smoothing and downsampling step (Algorithm 1, line 8) is not performed. Finally, when the adaptive threshold is replaced by the constant $\log 0.5$ threshold, the accuracy drops by 3%.

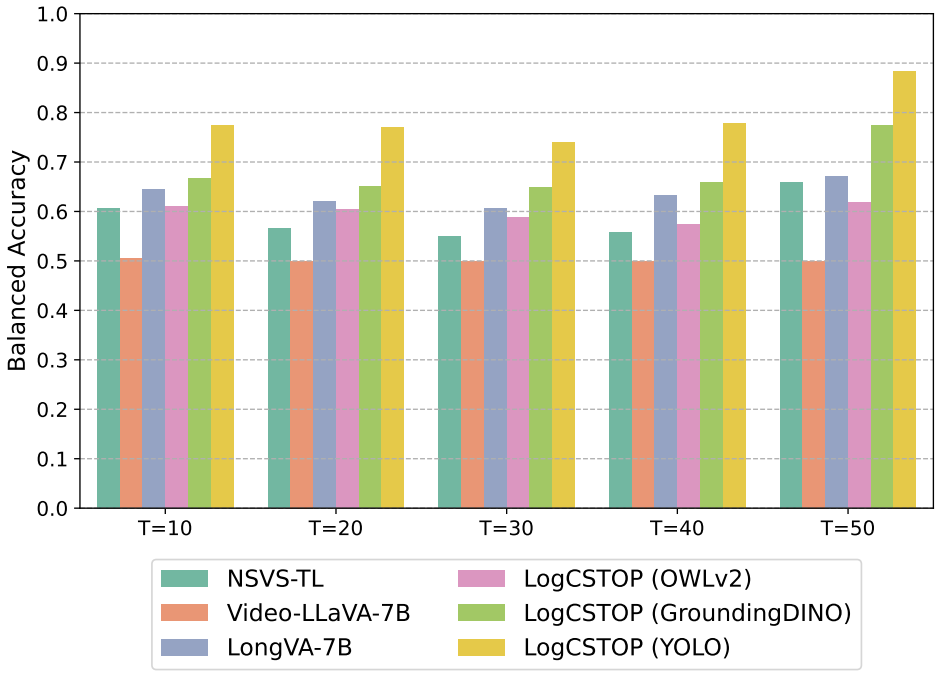

Figure 2: LogCSTOP reports the best and second-best performances (with YOLOv8 and Grounding DINO respectively) across sequences of different lengths on the CSTOP-RealTLV-video dataset.

5 Related Work

Temporal Logic for video understanding. Recent works [23, 6] use the probabilistic model checker STORM to verify temporal properties in LTL and PCTL over object detections in videos.

Quantiative semantics for Temporal Logic. Several quantitative semantics have been proposed for different temporal logics [1, 9, 17, 8], for monitoring, control, etc. To the best of our knowledge, no other work has explored using these for assigning confidence scores for temporal properties.

Benchmarks for video understanding. Existing benchmarks for video understanding such as Video-MME [10], RexTIME [5], Next-qa [22], QVHighlights [13], TemporalBench [4] and TempCompass [16] include tasks that require temporal understanding of actions and events in videos. These tasks are fundamentally different from the CSTOP benchmark which focuses on more fine-grained temporal properties.

6 Conclusion, Limitations and Future Work

In this work, we study the problem of assigning confidence scores for temporal properties (CSTOPs) given potentially noisy confidence score predictors for local properties. We represent these properties using LTL and propose a scoring function LogCSTOP inspired by quantitative semantics for Temporal Logic. We introduce the CSTOP-RealTLV-video benchmark for evaluating LogCSTOP on the downstream application of query matching with temporal properties (TP-matching) and find that it outperforms LVLMs on TP-matching with videos.

Limitations. Firstly, there are properties such as "there are always 2 cars" that cannot be expressed in LTL. Future work should hence explore more expressive logics [2] or construct local predictors for complex properties. Secondly, we only evaluate LogCSTOP on one downstream application but these scores can also be useful for other applications such as ranking and fine-tuning of local predictors with high level feedback. Finally, we only consider one input modality for local properties (video). It will be interesting to see LogCSTOP being used for multimodal applications where the local properties are over different modalities with scores from different local predictors.

References

[1] Takumi Akazaki and Ichiro Hasuo. Time robustness in mtl and expressivity in hybrid system falsification. In *International Conference on Computer Aided Verification*, 2015.

[2] Jacob Anderson, Georgios Fainekos, Bardh Hoxha, Hideki Okamoto, and Danil Prokhorov. Pattern matching for perception streams. In *International Conference on Runtime Verification*, pages 251–270. Springer, 2023.

[3] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[4] Mu Cai, Reuben Tan, Jianrui Zhang, Bocheng Zou, Kai Zhang, Feng Yao, Fangrui Zhu, Jing Gu, Yiwu Zhong, Yuzhang Shang, et al. Temporalbench: Benchmarking fine-grained temporal understanding for multimodal video models. *arXiv preprint arXiv:2410.10818*, 2024.

[5] Jr-Jen Chen, Yu-Chien Liao, Hsi-Che Lin, Yu-Chu Yu, Yen-Chun Chen, and Frank Wang. Rextime: A benchmark suite for reasoning-across-time in videos. *Advances in Neural Information Processing Systems*, 37:28662–28673, 2024.

[6] Minkyu Choi, Harsh Goel, Mohammad Omama, Yunhao Yang, Sahil Shah, and Sandeep Chinchali. Towards neuro-symbolic video understanding. In *European Conference on Computer Vision*, 2024.

[7] Giuseppe De Giacomo and Moshe Y. Vardi. Linear temporal logic and linear dynamic logic on finite traces. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, page 854–860. AAAI Press, 2013.

[8] Alexandre Donzé and Oded Maler. Robust satisfaction of temporal logic over real-valued signals. In *International Conference on Formal Modeling and Analysis of Timed Systems*, pages 92–106. Springer, 2010.

[9] Georgios E. Fainekos and George J. Pappas. Robustness of temporal logic specifications for continuous-time signals. *Theoretical Computer Science*, 410(42):4262–4291, 2009.

[10] Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, et al. Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis. *arXiv preprint arXiv:2405.21075*, 2024.

[11] Christian Hensel, Sebastian Junges, Joost-Pieter Katoen, Tim Quatmann, and Matthias Volk. The probabilistic model checker storm. *International Journal on Software Tools for Technology Transfer*, pages 1–22, 2022.

[12] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Ultralytics yolov8, 2023.

[13] Jie Lei, Tamara L Berg, and Mohit Bansal. Detecting moments and highlights in videos via natural language queries. *Advances in Neural Information Processing Systems*, 34:11846–11858, 2021.

[14] Bin Lin, Yang Ye, Bin Zhu, Jiaxi Cui, Munan Ning, Peng Jin, and Li Yuan. Video-llava: Learning united visual representation by alignment before projection. *arXiv preprint arXiv:2311.10122*, 2023.

[15] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In *European conference on computer vision*, pages 38–55. Springer, 2024.

[16] Yuanxin Liu, Shicheng Li, Yi Liu, Yuxiang Wang, Shuhuai Ren, Lei Li, Sishuo Chen, Xu Sun, and Lu Hou. Tempcompass: Do video llms really understand videos? *arXiv preprint arXiv:2403.00476*, 2024.

[17] Noushin Mehdipour, Cristian-Ioan Vasile, and Calin Belta. Generalized mean robustness for signal temporal logic. *IEEE Transactions on Automatic Control*, pages 1–8, 2024.

[18] Matthias Minderer, Alexey Gritsenko, and Neil Houlsby. Scaling open-vocabulary object detection. *Advances in Neural Information Processing Systems*, 36:72983–73007, 2023.

[19] Amir Pnueli. The temporal logic of programs. In *18th annual symposium on foundations of computer science (sfcs 1977)*, pages 46–57. ieee, 1977.

[20] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[21] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[22] Junbin Xiao, Xindi Shang, Angela Yao, and Tat-Seng Chua. Next-qa: Next phase of question-answering to explaining temporal actions. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9777–9786, 2021.

[23] Yunhao Yang, Jean-Raphaël Gaglione, Sandeep Chinchali, and Ufuk Topcu. Specification-driven video search via foundation models and formal verification. *arXiv preprint arXiv:2309.10171*, 2023.

[24] Peiyuan Zhang, Kaichen Zhang, Bo Li, Guangtao Zeng, Jingkang Yang, Yuanhan Zhang, Ziyue Wang, Haoran Tan, Chunyuan Li, and Ziwei Liu. Long context transfer from language to vision. *arXiv preprint arXiv:2406.16852*, 2024.

Acknowledgments and Disclosure of Funding

Use unnumbered first level headings for the acknowledgments. All acknowledgments go at the end of the paper before the list of references. Moreover, you are required to declare funding (financial activities supporting the submitted work) and competing interests (related financial activities outside the submitted work). More information about this disclosure can be found at: `https://neurips.cc/Conferences/2025/PaperInformation/FundingDisclosure`.

Do **not** include this section in the anonymized submission, only in the final paper. You can use the `ack` environment provided in the style file to automatically hide this section in the anonymized submission.

A More Details on Quantitative Semantics for Temporal Logic

We now present more details on the standard quantitative semantics for Signal Temporal Logic (STL) discussed in Section 3.1. A formula in Signal Temporal Logic can be written as:

$$\varphi := \top \mid \mu \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \mathcal{U}_I \varphi_2$$

where $\mu = f(s(t)) \geq 0$ is a Lipschitz continuous function over the signal s and $I = [t_1, t_2]$ is a time interval, $t_2 \geq t_1 \geq 0$. The operators *Eventually* and *Always* can be defined as follows:

$$\Diamond_I \varphi := \top \mathcal{U}_I \varphi$$
$$\Box_I \varphi := \neg \Diamond_I \neg \varphi$$

In the context of the CSTOP problem, $s(t) = \hat{y}(\cdot, t)$ and $f_c(s(t)) = \hat{y}(c, t) - \tau$. The query "car until pedestrian" can be written as $\mu_{car}(s(t)) \, \mathcal{U}_I \, \mu_{pedestrian}(s(t))$ where $I = [0, T]$ and $\mu_{car}(s(t)) = \hat{y}(car, t) - \tau \geq 0$ ($\mu_{pedestrian}$ is defined similarly).

The STL quantitative semantics, also called *robustness* ρ, is defined as follows to indicate how much a signal satisfies or violates the formula:

$$\rho(\top, s, t) := \rho_\top$$
$$\rho(\mu, s, t) := f(s(t))$$
$$\rho(\neg\varphi, s, t) := -\rho(\varphi, s, t)$$
$$\rho(\varphi_1 \wedge \varphi_2, s, t) := \min(\rho(\varphi_1, s, t), \rho(\varphi_2, s, t))$$
$$\rho(\varphi_1 \vee \varphi_2, s, t) := \max(\rho(\varphi_1, s, t), \rho(\varphi_2, s, t))$$
$$\rho(\varphi_1 \, \mathcal{U}_I \, \varphi_2, s, t) := \sup_{t' \in t+I} (\min\{\rho(\varphi_2, s, t'), \inf_{t'' \in [t, t']} \rho(\varphi_1, s, t'')\})$$
$$\rho(\Box_I \, \varphi, s, t) := \inf_{t' \in [t+I]} \rho(\varphi, s, t')$$
$$\rho(\Diamond_I \, \varphi, s, t) := \sup_{t' \in [t+I]} \rho(\varphi, s, t')$$

where, $\rho_\top$ is the maximum robustness, i.e., $\rho_\top = b - \tau$ for the CSTOP problem where $b = \max(\hat{y}(\cdot, \cdot))$.

As discussed earlier, the robustness measure assigns similar scores to a wide range of sequences, making it an unsuitable CSTOP solution for downstream applications such as ranking and optimization. We present an example with the *Always* operator in Section 3.1. We now present more examples with other operators where the confidence score assigned by the robustness measure differs from LogCSTOP (here $\hat{y}(\cdot, \cdot)$ are in the $\log$ scale, $\tau = \log(0.5)$ and the window $w = 1$ for LogCSTOP):

An example with Boolean operators. For example, consider assigning confidence scores to the property "car and pedestrian" at $t = 0$ in two different scenarios:

$$\hat{y}_1(car, t = 0) = \log(0.9), \hat{y}_1(pedestrian, t = 0) = \log(0.6)$$

$$\hat{y}_2(car, t = 0) = \log(0.6), \hat{y}_2(pedestrian, t = 0) = \log(0.6)$$

Ideally, the confidence scores for "car and pedestrian" should follow the order: $\hat{y}_1(car \wedge pedestrian, t = 0) > \hat{y}_2(car \wedge pedestrian, t = 0)$ since $\hat{y}_1(car, t = 0) > \hat{y}_2(car, t = 0)$. The robustness for both the cases is the same, i.e., $\log(0.6) - \log(0.5)$, because of the $\min$ semantics for the Boolean *and* operator. The LogCSTOP for the two cases are $\log(0.9) + \log(0.6)$ and $\log(0.6) + \log(0.6)$ respectively, which reflect the expected order.

An example with the Until operator. Consider assigning confidence scores to the property "car Until pedestrian" in two different scenarios:

$$\hat{y}_1(car, t \in [0, 2]) = [\log(0.6), \log(0.6), \log(0.6)]$$

$$\hat{y}_1(pedestrian, t \in [0, 2]) = [\log(0.4), \log(0.4), \log(0.9)]$$

and,

$$\hat{y}_2(car, t \in [0,2]) = [\log(0.6), \log(0.6), \log(0.6)]$$
$$\hat{y}_2(pedestrian, t \in [0,2]) = [\log(0.4), \log(0.4), \log(0.6)]$$

Note that the only difference between the two scenarios is the score for "pedestrian" at $t = 2$ (a high score of 0.9 for the first scenario and a lower score of 0.6 for the second scenario) . The robustness for the two cases is the same because of the $\min$ semantics within the *Until* operator. LogCSTOP assigns a higher score for the first scenario because of the difference at $t = 2$.

B Other Experiment Details

Compute Resources. All experiments were run on a shared cluster with the following GPUs: eight NVIDIA A100 PCIe (80GB RAM each) and eight NVIDIA RTX A6000 (48GB RAM each).

C LogCSTOP and NSVS-TL Queries

We choose 15 temporal property templates for the experiments in Section 4.

The LogCSTOP queries for these templates are as follows:

1. Eventually p1: $\Diamond\, p1$
2. Always p1: $\Box\, p1$
3. p1 Until p2: $p1\, \mathcal{U}\, p2$
4. Always p1 and Eventually p2: $\Box\, p1 \wedge \Diamond\, p2$
5. Always p1 or Eventually p2: $\Box\, p1 \vee \Diamond\, p2$
6. (Not p1) Until p2: $\neg p1\, \mathcal{U}\, p2$
7. p1 Until (Not p2): $p1\, \mathcal{U}\, \neg p2$
8. Always (p1 and p2): $\Box(p1 \wedge p2)$
9. (p1 and p2) Until p3: $(p1 \wedge p2)\, \mathcal{U}\, p3$
10. p1 Until Always p2: $p1\, \mathcal{U}\, \Box\, p2$
11. Eventually Always p1: $\Diamond\, \Box\, p1$
12. Always Eventually p1: $\Box\, \Diamond\, p1$
13. (Not p1) Until Eventually p2: $\neg p1\, \mathcal{U}\, \Diamond\, p2$
14. (Not p1) Until Always p2: $\neg p1\, \mathcal{U}\, \Box\, p2$
15. (p1 and p2) Until Eventually p3: $(p1 \wedge p2)\, \mathcal{U}\, \Diamond\, p3$

NSVS-TL [6] uses the model checker STORM [11] to verify if a given sequence satisfies a temporal property, where the temporal properties are represented in Probabilistic Computation Tree Logic (PCTL). In PCTL, the F, G and U operators represent the *Eventually*, *Always* and *Until* operators respectively. The $\sim$, $\&$ and $|$ operators represent the Boolean *negation*, *and*, and *or* operators respectively. The operator P is used to indicate the ranges of probability of a given property being satisfied: for example, $P > 0.5[F\, \varphi]$ translates to "the probability of φ eventually being satisfied is more than 0.5".

The PCTL queries for the 15 temporal property templates are as follows:

1. Eventually p1: $P > 0.5\, [F$ "$p1$" $]$
2. Always p1: $P > 0.5\, [G$ "$p1$" $]$
3. p1 Until p2: $P > 0.5\, [$ "$p1$" U "$p2$" $]$
4. Always p1 and Eventually p2: $P > 0.5\, [\, G$ "$p1$" $\&\, F$ "$p2$" $]$
5. Always p1 or Eventually p2: $P > 0.5\, [\, G$ "$p1$" $|\ F$ "$p2$" $]$
6. (Not p1) Until p2: $P > 0.5\, [\sim$ "$p1$" U "$p2$" $]$

7. p1 Until (Not p2): $P > 0.5\,[\,\text{"}p1\text{"}\ U \sim \text{"}p2\text{"}\,]$
8. Always (p1 and p2): $P > 0.5\,[\,G\ \text{"}p1\text{"}\ \&\, G\ \ \text{"}p2\text{"}\,]$
9. (p1 and p2) Until p3: $P > 0.5\,[\,\text{"}p1\text{"}\ \&\ \text{"}p2\text{"}\ U\ \text{"}p3\text{"}\,]$
10. p1 Until Always p2: $P > 0.5\,[\,\text{"}p1\text{"}\ U\, G\ \ \text{"}p2\text{"}\,]$
11. Eventually Always p1: $P > 0.5\,[F\, G\ \ \text{"}p1\text{"}\,]$
12. Always Eventually p1: $P > 0.5\,[G\, F\ \ \text{"}p1\text{"}\,]$
13. (Not p1) Until Eventually p2: $P > 0.5\,[\sim \text{"}p1\text{"}\ U\, F\ \ \text{"}p2\text{"}\,]$
14. (Not p1) Until Always p2: $P > 0.5\,[\sim \text{"}p1\text{"}\ U\, G\ \ \text{"}p2\text{"}\,]$
15. (p1 and p2) Until Eventually p3: $P > 0.5\,[\,\text{"}p1\text{"}\ \&\ \text{"}p2\text{"}\ U\, F\ \ \text{"}p3\text{"}\,]$

D Detailed Results

In Section 4, we present the query matching results for the CSTOP-RealTLV-video dataset, across sequences of different lengths and query templates. We now present the detailed results. Table 2, Table 3, Table 4, Table 5, and Table 6 present the query-wise results for the CSTOP-RealTLV-video dataset with sequences of lengths $10, 20, 30, 40$, and 50 respectively.

Table 2: Results on the CSTOP-RealTLV-video dataset for sequences of length 10. The best performing method is highlighted in **bold** and the second best is underlined.

Query	NSVS-TL	Video-LLaVA-7B	LongVA-7B	LogCSTOP		
				OWLv2	GroundingDINO	YOLOv8
Eventually p1	0.69	0.5	0.76	0.76	0.67	**0.84**
Always p1	0.71	0.53	0.7	0.54	0.72	**0.74**
p1 Until p2	0.75	0.5	0.69	0.65	0.75	**0.84**
Always p1 and Eventually p2	0.59	0.5	0.64	0.53	0.64	**0.71**
Always p1 or Eventually p2	0.74	0.5	0.6	0.68	0.68	**0.81**
Not p1 Until p2	0.5	0.5	0.69	0.71	0.74	**0.83**
p1 Until Not p2	0.5	0.49	0.68	0.56	0.64	**0.76**
Always (p1 and p2)	0.57	0.54	0.6	0.52	0.64	**0.72**
p1 and p2 Until p3	0.5	0.5	0.57	0.5	0.64	**0.72**
p1 Until Always p2	0.63	0.5	0.54	0.5	0.6	**0.76**
Eventually Always p1	0.67	0.5	0.64	0.68	0.68	**0.77**
Always Eventually p1	0.65	0.51	0.72	0.56	0.72	**0.76**
(Not p1) Until Eventually p2	0.5	0.5	0.65	0.64	0.57	**0.77**
(Not p1) Until Always p2	0.5	0.5	0.64	0.64	0.66	**0.73**
(p1 and p2) Until Eventually p3	0.5	0.49	0.56	0.7	0.66	**0.84**
Overall	0.6	0.5	0.65	0.61	0.67	**0.77**

Table 3: Results on the CSTOP-RealTLV-video dataset for sequences of length 20. The best performing method is highlighted in **bold** and the second best is underlined.

Query	NSVS-TL	Video-LLaVA-7B	LongVA-7B	LogCSTOP		
				OWLv2	GroundingDINO	YOLOv8
Eventually p1	0.57	0.5	0.71	0.74	0.62	**0.83**
Always p1	0.67	0.5	0.75	0.54	0.77	**0.83**
p1 Until p2	0.67	0.5	0.63	0.63	0.71	**0.78**
Always p1 and Eventually p2	0.54	0.5	0.69	0.51	0.62	**0.71**
Always p1 or Eventually p2	0.66	0.5	0.62	0.7	0.54	**0.75**
Not p1 Until p2	0.5	0.5	0.62	0.68	0.71	**0.86**
p1 Until Not p2	0.5	0.5	0.57	0.55	0.58	**0.76**
Always (p1 and p2)	0.54	0.5	0.68	0.51	0.63	**0.7**
p1 and p2 Until p3	0.5	0.5	0.53	0.59	0.7	**0.79**
p1 Until Always p2	0.61	0.5	0.56	0.5	0.61	**0.69**
Eventually Always p1	0.58	0.5	0.52	0.71	0.75	**0.78**
Always Eventually p1	0.58	0.5	0.6	0.52	0.65	**0.71**
(Not p1) Until Eventually p2	0.5	0.5	0.62	0.62	0.63	**0.82**
(Not p1) Until Always p2	0.5	0.5	0.65	0.6	0.66	**0.77**
(p1 and p2) Until Eventually p3	0.5	0.5	0.55	0.67	0.57	**0.77**
Overall	0.56	0.5	0.62	0.6	0.65	**0.77**

Table 4: Results on the CSTOP-RealTLV-video dataset for sequences of length 30. The best performing method is highlighted in **bold** and the second best is underlined.

Query	NSVS-TL	Video-LLaVA-7B	LongVA-7B	LogCSTOP		
				OWLv2	GroundingDINO	YOLOv8
Eventually p1	0.55	0.5	0.67	0.62	0.67	**0.8**
Always p1	0.61	0.5	0.71	0.53	0.67	**0.75**
p1 Until p2	0.61	0.5	0.59	0.6	**0.77**	0.74
Always p1 and Eventually p2	0.5	0.5	0.65	0.49	0.63	**0.69**
Always p1 or Eventually p2	0.67	0.5	0.59	0.62	0.6	**0.69**
Not p1 Until p2	0.5	0.5	0.69	0.64	0.75	**0.84**
p1 Until Not p2	0.5	0.5	0.58	0.56	0.66	**0.81**
Always (p1 and p2)	0.5	0.5	0.55	0.5	0.53	**0.62**
p1 and p2 Until p3	0.5	0.5	0.58	0.61	0.68	**0.76**
p1 Until Always p2	0.66	0.5	0.58	0.52	0.59	**0.72**
Eventually Always p1	0.54	0.5	0.53	0.61	0.68	**0.69**
Always Eventually p1	0.56	0.5	0.67	0.54	0.6	**0.68**
(Not p1) Until Eventually p2	0.5	0.5	0.52	0.56	0.56	**0.71**
(Not p1) Until Always p2	0.5	0.5	0.65	0.66	0.67	**0.75**
(p1 and p2) Until Eventually p3	0.5	0.5	0.53	0.76	0.68	**0.84**
Overall	0.55	0.5	0.61	0.59	0.65	**0.74**

Table 5: Results on the CSTOP-RealTLV-video dataset for sequences of length 40. The best performing method is highlighted in **bold** and the second best is underlined.

Query	NSVS-TL	Video-LLaVA-7B	LongVA-7B	LogCSTOP		
				OWLv2	GroundingDINO	YOLOv8
Eventually p1	0.55	0.5	0.7	0.53	0.56	**0.82**
Always p1	0.62	0.5	0.73	0.52	0.78	**0.84**
p1 Until p2	0.62	0.5	0.56	0.58	0.65	**0.73**
Always p1 and Eventually p2	0.51	0.5	0.58	0.5	0.61	**0.7**
Always p1 or Eventually p2	0.65	0.5	0.65	0.71	0.67	**0.85**
Not p1 Until p2	0.5	0.5	0.65	0.69	0.74	**0.79**
p1 Until Not p2	0.5	0.5	0.61	0.49	0.6	**0.7**
Always (p1 and p2)	0.54	0.5	0.66	0.51	0.65	**0.75**
p1 and p2 Until p3	0.5	0.5	0.55	0.53	0.69	**0.73**
p1 Until Always p2	0.61	0.5	0.57	0.51	0.51	**0.72**
Eventually Always p1	0.61	0.5	0.67	0.72	**0.79**	**0.79**
Always Eventually p1	0.61	0.5	0.72	0.52	0.71	**0.75**
(Not p1) Until Eventually p2	0.5	0.5	0.61	0.56	0.58	**0.85**
(Not p1) Until Always p2	0.5	0.5	0.7	0.65	0.7	**0.8**
(p1 and p2) Until Eventually p3	0.5	0.5	0.55	0.56	0.61	**0.86**
Overall	0.55	0.5	0.63	0.57	0.66	**0.78**

Table 6: Results on the CSTOP-RealTLV-video dataset for sequences of length 50. The best performing method is highlighted in **bold** and the second best is underlined.

Query	NSVS-TL	Video-LLaVA-7B	LongVA-7B	LogCSTOP		
				OWLv2	GroundingDINO	YOLOv8
Eventually p1	0.74	0.5	0.71	0.59	0.57	**0.93**
Always p1	0.8	0.5	0.73	0.57	0.88	**0.9**
p1 Until p2	0.87	0.5	0.6	0.67	0.76	**0.93**
Always p1 and Eventually p2	0.63	0.5	0.69	0.54	0.73	**0.77**
Always p1 or Eventually p2	**0.85**	0.5	0.58	0.71	0.56	0.81
Not p1 Until p2	0.5	0.5	0.75	0.84	0.9	**0.95**
p1 Until Not p2	0.5	0.5	0.78	0.5	0.84	**0.88**
Always (p1 and p2)	0.62	0.5	0.61	0.5	0.82	**0.85**
p1 and p2 Until p3	0.5	0.5	0.53	0.57	0.86	**0.92**
p1 Until Always p2	0.68	0.5	0.61	0.5	0.77	**0.85**
Eventually Always p1	0.78	0.5	0.76	0.76	0.9	**0.93**
Always Eventually p1	0.79	0.5	0.72	0.59	**0.9**	0.89
(Not p1) Until Eventually p2	0.5	0.5	0.65	0.58	0.59	**0.89**
(Not p1) Until Always p2	0.5	0.5	0.75	0.75	**0.87**	**0.87**
(p1 and p2) Until Eventually p3	0.5	0.5	0.58	0.58	0.59	**0.89**
Overall	0.65	0.5	0.67	0.62	0.77	**0.88**