Martingale Tests for Model Misspecification in Bayesian Sequence Prediction

Kee Siong Ng
The Australian National University

Sigfrido D. Ciletti
The Australian National University

Abstract

Universal Bayesian sequence predictors like Context Tree Weighting (CTW) offer strong asymptotic convergence guarantees but remain vulnerable to structural misspecification and finite-sample miscalibration issues when deployed in complex, non-stationary environments. This paper introduces an online monitoring framework for Bayesian sequence prediction in the form of a suite of test martingales designed to continuously audit CTW and related methods. The martingales are constructed using betting functions on prequential and conformal p-values, including a universal betting function based on the Transformer architecture. We show that this approach can effectively isolate diverse failure modes in CTW, including static marginal errors and hidden temporal dependencies. Further, we provide an active recalibration mechanism for CTW that can dynamically respond to martingale alerts.

1 Introduction

Bayesian mixture estimators are known to be optimal for a wide range of learning problems. In sequential prediction, the Solomonoff prediction scheme [37, 38], which uses a Bayesian mixture over all Turing machines weighted by their description length, has been shown to converge rapidly to any true computable environment. A related but less well-known result concering probabilistic logics [10, 18] is that mixtures over the possible interpretations of a theory in higher-order logic allow Bayesian inductive reasoning and learning in the limit. In particular, this approach enables the confirmation of universally quantified hypothesis—a notorious problem in confirmation theory [32].

In game theory, the celebrated result of Kalai and Lehrer [20] shows that a group of agents will converge to an ϵ-Nash equilibrium in repeated games, provided that they (1) use Bayesian mixture and updating to track other agents' strategies, and (2) produce a best-response policy to that mixture. This convergence holds as long as there is a "grain of truth" in their beliefs; i.e., the true strategy profile of the opponents is assigned a non-zero prior probability. Both the Solomonoff and game-theoretic results have since been extended to the general reinforcement learning (GRL) setting and the multi-agent setting [16, 19, 22, 24].

The condition for success in each of the cases above is that the true unknown environment the Bayesian mixture estimator is "trying to learn" is in the model class and is assigned a non-zero prior probability. If this condition is violated we say the model is misspecified; under such circumstances Bayesian mixture estimation can exhibit problematic and pathological behaviour [15, 39]. This is not an issue for the aforementioned idealised Bayesian mixture estimators, precisely because they range over maximally extensive model classes for their respective problem settings; however, this in turn makes them impractical. These incomputable estimators must be approximated using restricted model classes, which is where an algorithm's ability to handle model misspecification becomes a key design consideration.

Preprint.

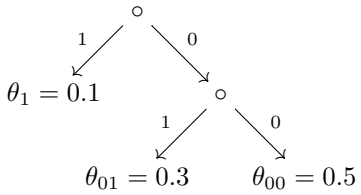

Figure 1: An example prediction suffix tree

In this paper, we study how sequential test martingales [7, 48] can be used to detect and quantify model misspecification when using Bayesian mixture estimators for online sequence prediction; a foundational setting underpinning the multi-agent GRL framework. We now list the technical contributions of the paper.

1. We introduce a suite of test martingales that provide online monitoring and isolation of model misspecification in CTW [51]. These methods belong to the class of universal Bayesian sequence predictors for n-order Markov processes.
2. In constructing the null hypotheses for these test martingales, we apply the randomised probability integral transform theorem (PIT) on confidence-ordered categorical variables to construct efficiently computable prequential and conformal p-values for CTW. The randomised ranked PIT is a straightforward application of ideas in [36, 6, 47] but appears not to have been previously published.
3. We also introduce a universal test martingale that adaptively constructs betting functions based on transformers [40] trained on p-values and raw input data.
4. Finally, we introduce a new adaptive CTW algorithm that can adjust the model parameters whenever a model misspecification issue is detected.

Related Work The design of our sequential testing procedure builds on several well-established lines of research: the e-values and safe testing framework [50, 27, 31], conformal e-testing [47], prequential probability [7], Bayesian mixture estimators for sequence prediction in general reinforcement learning [37, 51, 43], and the recently demonstrated universality of transformer neural networks [26, 13]. A theoretical treatment of the relationship between e-values, hypothesis testing, and Bayesian learning can be found in [29]. Building on this body of work, we focus on designing efficient sequential testing procedures for Bayesian sequence prediction that address the explainability and robustness requirements inherent in high-risk AI applications. Our intentions are similar to that of Podkopaev and Ramdas in [28]; but we go beyond distribution shifts.

Paper Organisation The remainder of this paper is organised as follows. §2 provides an brief exposition of the CTW algorithm and its extensions. §3 describes the sequential hypothesis testing protocol, in the form of test martingales on e-values [50]. In §4 we describe a new dynamic CTW algorithm. §5 outlines our experimental results. This is followed by a discussion and concluding remarks in §6. Technical proofs are deferred to the appendix.

2 Context Tree Weighting and Related Methods

CTW is an online Bayesian model averaging scheme that computes a probability, at each time step t:

$$P(x_{1:t}) = \sum_{M} P(x_{1:t} \mid M) P(M), \tag{1}$$

where $x_{1:t}$ is the observed binary sequence, M is a prediction suffix tree, $P(M)$ is the prior probability of M. The summation is taken over the set of *all* prediction suffix trees of bounded depth D—a large class covering all D-order Markov processes. While a naïve computation of 1 would require $O(2^{2^D})$ time, under the CTW algorithm this is $O(D)$.

Definition 1. A prediction suffix tree (PST) is a pair (M, Θ), where M is a binary tree, and associated with each leaf node l in M is a Bernoulli distribution parametrised by $\theta_l \in \Theta$.

A prediction suffix tree (M, Θ) maps each binary string $x_{1:t}$—where t is greater than or equal to the depth of M—to the probability distribution $\theta_{M(x_{1:t})}$, where $M(x_{1:t})$ denotes the leaf node of M reached by traversing the tree using $x_{1:t}$ in reverse order. The intended meaning is that $\theta_{M(x_{1:t})}$ is the probability that the next bit following $x_{1:t}$ is 1. For example, the PST in Fig. 1 maps the string 0110 to $\theta_{M(0110)} = \theta_{01} = 0.3$, which means the next bit after 0110 is 1 with probability 0.3.

In the CTW algorithm, the Bernoulli parameter at each node of a PST is learned from data using the Krichevsky-Trofimov Estimator [21]. Given a binary string $x_{1:t}$ with a zeros and b ones, the KT estimate of the probability of the next symbol is as follows:

$$P_{\mathrm{KT}}(X_{t+1} = 1 \mid x_{1:t}) := \frac{b + 1/2}{a + b + 1} \tag{2}$$

$$P_{\mathrm{KT}}(X_{t+1} = 0 \mid x_{1:t}) := 1 - P_{\mathrm{KT}}(X_{t+1} = 1 \mid x_{1:t}). \tag{3}$$

The KT estimator is obtained via a Bayesian analysis by placing a Jeffreys prior on the unkown parameter $\theta \in [0, 1]$ of a Bernoulli process. The KT estimator is known to converge to the true Bernoulli distribution P_θ at a rate of $KL(P_\theta \mid\mid P_{\mathrm{KT}}) = O\big(\frac{\log t}{t}\big)$.

The CTW algorithm operates on a context tree data structure. A context tree is a perfect binary tree of depth D such that attached to each node (both internal and leaf) is a probability distribution over the finite binary strings, denoted $\{0, 1\}^*$; these distributions are estimated from the data using KT estimators. At each time step t, the CTW probability is computed recursively at each node s in the context tree using the update rule

$$P_w^s(x_{1:t}) = \begin{cases} P_{\mathrm{KT}}(x_{1:t|s}) & \text{if } s \text{ is a leaf node} \\ \frac{1}{2} P_{\mathrm{KT}}(x_{1:t|s}) + \frac{1}{2} \prod_{i \in \{0,1\}} P_w^{si}(x_{1:t}) & \text{otherwise,} \end{cases} \tag{4}$$

where $x_{1:t|s}$ is the substring of $x_{1:t}$ identified with node s. Thus the computation starts at the leaf nodes of the context tree and finishes at the root node. A key result of [51] is that the probability $P_w^\epsilon(x_{1:t})$ computed at the root node ϵ of the context tree satisfies 1. Consequently, the predictive distribution for the next bit x_t, is given by:

$$P(x_t \mid x_{1:t-1}) = \frac{P(x_{1:t})}{P(x_{1:t-1})} = \frac{P_w^\epsilon(x_{1:t})}{P_w^\epsilon(x_{1:t-1})}. \tag{5}$$

Theorem 1. [51] Let P^* be a stationary tree-source distribution with suffix set S, and let Q_t denote the CTW predictive distribution at time t. For a sequence of length T, the average expected KL divergence is bounded by:

$$\frac{1}{T} \sum_{t=1}^{T} \mathbb{E}[\mathrm{KL}(P^* \mid\mid Q_t)] \leq \frac{|S| \log T}{2T} + \frac{2|S| - 1}{T}.$$

As $t \to \infty$, the average expected KL divergence goes to 0 at a rate dominated by the first term on the RHS i.e., $O(\frac{\log t}{t})$. Note how the CTW convergence result mirrors the Solomonoff Induction convergence result, but with the additional cost of using the KT estimator to learn the Bernoulli distributions at each leaf node.

CTW has been extended in various ways to compute the Bayesian mixture for larger model classes, including for non-stationary environments [44, 42] and arbitrary alphabets [41]. Variations of this algorithm have been used successfully to approximate GRL agents such as AIXI [19, 43, 53, 25], including in a human-machine teaming setup that allows the model class to be adjusted dynamically at run time by a human operator using domain knowledge [52].

3 Prequential and Conformal Test Martingales

In our setup, we use a CTW of depth D to learn an environment assumed to be a stationary n-order Markov process. If the true environment is misspecified (e.g., if it is non-stationary, possesses Markov dependencies longer than D, or follows a structure that cannot be captured by a prediction suffix tree) then the CTW model is considered misspecified. To detect

this, we begin by assuming the null hypothesis H_0, which posits that the CTW predictive probabilities are correct at every time step. We then employ test martingales (sequential betting strategies) to look for systematic deviations. A sequential betting strategy will "win" against H_0 in proportion to the degree of misspecification; in some cases, this allows us to diagnose the specific form of misspecification the CTW is exhibiting.

3.1 The Testing Protocol

Computing prequential and conformal p-values The testing protocol begins with the computation of prequential and conformal p-values. At time t, given the history $h_{<t} = x_{1:t-1}$, we compute on the basis of the newly observed symbol x_t a randomised prequential p-value (or u-variable [6]), based on the CTW predictive distribution (5):

$$u_t = \sum_{a:P(a|h_{<t})<P(x_t|h_{<t})} P(a \mid h_{<t}) + \tau_t \cdot P(x_t \mid h_{<t}), \tag{6}$$

where $\tau_t \sim U(0,1)$ is sampled from the uniform distribution on the interval $[0,1]$. The value u_t can be interpreted as a form of Rosenblatt Transformation [33]—a cornerstone technique for checking the calibration of probability forecasts [12]. If the CTW model is correctly specified, then the resulting sequence of u-variables is independent and identically distributed (iid) according to the uniform distribution $U(0,1)$, as shown in Proposition 2 in Appendix A.

Using a nonconformity score function $\alpha_t = \alpha(u_t)$, we then compute a conformal p-value using the standard rank-in-history with randomised tie-breaking formula

$$p_t = \frac{|\{i \in \{1,\dots,t\} : \alpha_i \geq \alpha_t\}| + \tau_t \cdot |\{i \in \{1,\dots,t\} : \alpha_i = \alpha_t|}{t}, \tag{7}$$

where $\tau_t \sim U(0,1)$ and p_t is basically the proportion of past nonconformity scores that are at least as large as the last value α_t for x_t. Given the $u_{1:t}$ sequence is independent and uniformly distributed by Proposition 2, the $\alpha_{1:t}$ sequence is thus exchangeable, which then implies the conformal p-values $p_{1:t}$ are independent and uniformly distributed by Proposition 3.

Converting to e-values To transform these p-values into so-called e-values [50] that can be used as a test for model misspecification, we employ a betting function $e_t = f(u_t; u_{<t})$, where the input to f can be either the prequential p-values (6) or the conformal p-values (7). The primary requirement for the betting function f is that under the null hypothesis it satisfy

$$\mathbb{E}[f(u_t; u_{<t})] = \int_0^1 f(u_t; u_{<t})\, du_t = 1. \tag{8}$$

Note that, by Proposition 2, the u_t variable is uniformly distributed in $[0,1]$. This function is designed to increase in magnitude if the values $u_1, u_2, \dots$ are not independent and uniformly distributed. A common choice of the betting function that tests for the uniformity of each u_t is the memoryless Power (sometimes Linear) gambler:

$$f(u_t) = \kappa u_t^{\kappa-1} \tag{9}$$

where $\kappa \in (0,1)$ or $\kappa > 1$ depending on what kind of misspecification we suspect e.g., over-dispersed vs under-dispersed. A simple betting function that tests for the independence of the u_t values is the autocorrelation function

$$g(u_t; u_{<t}) = 1 + \lambda(u_t - 0.5)(u_{t-s} - 0.5) \tag{10}$$

where $\lambda \in (0,4)$ and s is a positive integer. The betting function "wins" when there is a positive correlation between u_t and u_{t-s}. The design of betting functions is studied in more detail in § 3.2, where we also propose a universal betting function based on Transformers [40].

The e-martingale update We track the evidence against the model using two e-martingales M_n^u and M_n^p defined as the running product of our e-values:

$$M_n^u = \prod_{t=1}^{n} f(u_t; u_{<t}) \qquad\qquad M_n^p = \prod_{t=1}^{n} g(p_t; u_{<t})$$

here f and g are appropriately chosen betting functions. To see that M_n^u is a martingale under the null hypothesis that the CTW is learning the true underlying distribution, we observe that by Proposition 2 and Equation 8

$$\mathbb{E}[M_n^u \mid u_{<n}] = \mathbb{E}[M_{n-1}^u \cdot f(u_n; u_{<n}) \mid u_{<n}] = M_{n-1}^u \cdot \mathbb{E}[f(u_n; u_{<n})] = M_{n-1}^u.$$

A similar argument can be used to show that M_n^p is a martingale under the null hypothesis that the sequence $\alpha_{1:n}$ is exchangeable.

Decision rule and diagnostics As we have seen, under the null hypotheses, the expected value of both M_n^u and M_n^p is 1. By Ville's Inequality [8, 23], for $i \in \{u, p\}$, for any $a > 0$

$$P\left(\sup_{n \geq 0} M_n^i \geq a\right) \leq \frac{\mathbb{E}[M_1^i]}{a} = \frac{1}{a}.$$

Note that the Ville Inequality is an anytime property that bounds the probability that the martingale process ever crosses a certain threshold, regardless of how long it runs. So, for example, if $M_n^i \geq 20$, then we have strong evidence (at the $a = 0.05$ confidence level) that the CTW model is misspecified. In § 4.1, we will look at how we can use multiple test martingales to diagnose and, in some cases, isolate exact misspecification issues in CTW.

3.2 The Design of Betting Functions

The power of these test martingales depends on the design of betting functions capable of exploiting structural regularities in the p-values that arise when the null hypothesis—independence and uniformity—is violated. We have seen the Power gambler and autocorrelation betting functions in § 3.1. Betting functions based on Poisson distributions are also possible [17]. Since any convex combination of betting functions is itself a valid betting function [50], a robust practical strategy is to aggregate a diverse ensemble of functions to ensure the testing algorithm can detect a wide range of misspecifications.

Learning Betting Functions

Betting functions can be learned adaptively from data rather than remaining fixed. For instance, it has been demonstrated [35] that a sequence of betting functions can be learned as witness functions for the Kernel Maximum Mean Discrepancy metric [14], measuring the distance between the observed sequence and a hypothetical iid process. Alternatively, proposals [27] include an adaptive strategy utilising Reverse Information Projection to select a distribution that minimizes the KL divergence from the observed data, effectively optimizing the betting performance against the null.

A Universal Betting Function

Motivated by the practical success of transformers [40], their universality [26], and recent results showing they can learn in context through mechanisms like Bayesian model averaging [11, 55], and gradient descent [45, 1, 54] a variety of statistical learning algorithms [3, 13]; we propose in this paper the use of transformers to learn a sequence of betting functions.

To construct the betting functions, a transformer is trained sequentially on the conformal p-values u_t produced by the CTW. At every time step, the transformer outputs a $p_{bet} = P(u_t > 0.5 \mid u_{1:t-1})$ that predicts whether u_t will land in $[0, 0.5]$ or $(0.5, 1]$. The betting function used is given by

$$f(u_t) = \begin{cases} 2 \cdot p_{bet} & u_t > 0.5 \\ 2(1 - p_{bet}) & \text{otherwise,} \end{cases} \tag{11}$$

which integrates to 1 as required. To see the rationale behind (11), note that we simply divide the interval $[0, 1]$ into two halves $B_1 = [0, 0.5]$ and $B_2 = (0.5, 1]$ and compute the Kelly criterion ratio of the probabilities of u_t falling into one of B_1 or B_2 under the alternative hypothesis (the transformer) and the null hypothesis. Thus, if $u_t \leq 0.5$, then the ratio is

$$f(u_t) = \frac{P(u_t \in B_1) \text{ under transformer model}}{P(u_t \in B_1) \text{ under null hypothesis } u_t \sim U[0,1]} = \frac{1 - p_{bet}}{0.5} = 2(1 - p_{bet}).$$

This can be generalised to the case where we divide $[0, 1]$ into $K > 2$ equally-sized intervals.

More generally and in keeping with the shift [34] to the design of betting functions that do not rely on p-values, we can use a transformer to learn from the same input data sequence given to CTW and adopt the following betting function derived from the Kelly criterion:

$$k(x_t) = \frac{Q_t(x_t \mid h_{<t})}{P_t(x_t \mid h_{<t})},$$

where x_t is the observed symbol at time t, $h_{<t} = x_{1:t-1}$, Q_t is the transformer model learned from $h_{<t}$, and P_t is the CTW model learned from $h_{<t}$. Under the null hypothesis that P_t is the true model, we have $\mathbb{E}_{P_t} k(x_t) = 1$ as required and the martingale capital will flatline over time. However, if the transformer model is more accurate than the CTW model, it will assign higher probabilities to events that actually occur and the martingale will grow. Depending on the specific Transformer learning objective adopted, attention maps [5] and other mechanistic interpretability techniques [4] can serve as diagnostic tools. These methods help elucidate the underlying features that enable a Transformer-based betting function to successfully exploit the CTW model's misspecifications.

4 A Monitoring and Dynamic Recalibration Algorithm for CTW

The analysis in Appendix A.4 shows if the underlying environment is an n-markov process, then there are only two possible sources of model misspecification in CTW. These are: (1) Insufficient model depth, and (2) Distribution shift. § 4.1 shows how each of these issues can be precisely detected via the use of one or more martingales. In § 4.2, we move from mere detection of misspecification to active calibration.

4.1 A Suite of Test Martingales for CTW

In general, the prequential martingale with a sufficiently powerful betting function faces a confounding problem: it tests a compound joint null hypothesis, H_0, which posits that there is no distribution shift *and* that the predictive CTW model perfectly captures the true underlying data-generating process.

If the test martingale M_n^u exhibits exponential growth, we can reject H_0. The issue is that the negation of H_0 has three possibilities that the prequential martingale cannot distinguish between these scenarios: (1) there is a distribution shift in the underlying environment; (2) the CTW model does not capture the true underlying environment; and lastly both (1) and (2) hold. We now look at how we can run multiple test martingales in parallel and use their combined behaviour to diagnose issues in a CTW model.

Isolating Distribution Shifts We first note that, unlike general black-box machine learning models that can exhibit different forms of (hard-to-control) statistical biases, the structure and in-built calibration features of CTW allows us to isolate distribution shift using the prequential martingale with the (memoryless) power gambler betting function, as shown in Proposition 4 in Appendix A.2. The result implies that, after an appropriate burn-in period, a prequential martingale with the power gambling function will not be able to detect model specification issues in a CTW related to either wrong marginals or wrong conditionals as described in § A.4. If such a martingale starts "winning", the only explanation is that there is a distribution shift in the underlying environment.

Controlling False Positives In many practical scenarios, there is a need to minimise the number of alerts to the system operator due to noise in the data owing to (non-structural) statistical fluctuations in the underlying data-generating process. This issue can be addressed by running both prequential and conformal martingales with the same betting functions in parallel. Proposition 5 in Appendix A.3 shows that the power of the prequential martingale, as measured by the expected stepwise logarithmic growth of martingale score, dominates that of a conformal martingale. The flipside of that result is that prequential martingale can be too sensitive to non-structural noise fluctuations in the input data, but the conformal martingale's empirical rank normalisation will be robust against such noise.

Thus, using the two martingales together and allowing the system operator to set joint thresholds for alerts can help in this scenario.

4.2 Active Recalibration

Algorithm 1 shows how we can use the suite of test martingales described in § 4.1 to actively recalibrate a CTW as needed. The algorithm uses three martinagles: the prequential martingale using the power gambler betting function, denoted P; the prequential martingale with transformer betting function, denoted T; and the conformal martingale with transformer betting function, denoted C, together with user defined "confidence levels", denoted α, β, γ, for P, T, C respectively. If $P > \alpha$ then a distribution shift has been detected, and the model if reinitialised. While if $T > \beta$ and $C > \gamma$ then an insufficiency in model depth has been detected and the model depth is extended.

Algorithm 1 Active Recalibration with Martingale Detection

Input : $\alpha, \beta, \gamma \in \mathbb{N}$
Initialise : $C_{D=3}$, P, T, C

For each new observation x_i **:**
  C_D.`updateCTW`(x_i)
  p_i = `p-value`(C_D)
  u_i = `u-value`(C_D)
  P.`update-prod`(u_i) ▷ Update running products
  T.`update-prod`(u_i)
  C.`update-prod`(p_i)

  If $P > \alpha$ **Then** C_D.`reinitialise()`
  Else if $T > \beta$ and $C > \gamma$ **Then** C_D.`extend`($D + 1$)

We now describe a number of possible optimisations to Algorithm 1. First of all, under benign conditions, it is possible to extend the depth of CTW by more than 1. Algorithm 2 in Appendix B describes a principled way to do that, through an analysis of the CTW model's redundancy in the two regimes $D < n$ and $D \geq n$, where D is the current CTW depth and n is the order of the underlying n-markov process.

Secondly, an obvious deficiency of the algorithm is the lack of any momentum parameter that can prevent the models from being modified during re-training time. In the absence of this burn-in rate, extensions and reinitialisaitons will occur before the model has been exposed to a sample sufficient to have learned the distribution from.

Thirdly, instead of reinitialising every time a distribution shift is detected ($P > \alpha$), the algorithm could store the existing CTWs in a list to be retrieved when distribution shift is detected. The idea being that for each distributrion an effective solution is saved, then when distribution shift is detected these solutions are tested against the "new" source, only when all past models fail is a new CTW initialised. Such a list of previous CTW models can be maintained using the Hedge algorithm [9, 2] (see [52]).

Fourthly, the martingale threshold checks can be improved using the CUSUM or Shiryaev-Roberts procedure [47, §8], which only triggers an alert when the difference in successive martingale scores exceed a certain threshold. This helps to minimise false alarms.

In addition to the above, we note that the techniques for protecting predictive systems, and in particular the Composite Jumper predictor algorithm developed by Vovk et al. [49], can by their general nature be adapted to our work. These methods would provide a so called "insurance policy" for a trained CTW. These techniques help smooth the inevitable awkward gaps between the change in distribution and its detection [49].

Finally, as suggested in § 3.2, it may be possible to run a transformer in parallel to a CTW on an input stream; having this transformer learn the true depth of the environment. This can save the substantial time lost via iterative deepening, which forces us to pass through every depth between D and n, provided the transformer learn faster than the CTW.

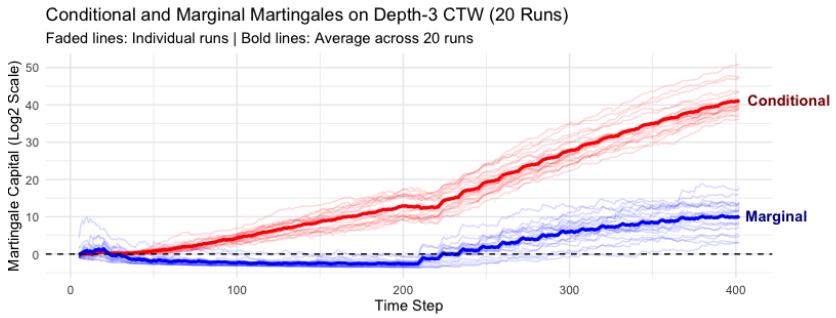

Figure 2: Scores for a conditional martingale and a marginal martingale when the context depth is inadequate and there is a distribution shift at time step 200.

5 Experiments

A few experiments can be found in Appendix C showing how the test martingales respond to three scenarios: (1) there is a distribution shift in the underlying environment; (2) the underlying environment has some occasional non-structural noise fluctuations and there is a distribution shift (robust detection is thus important here); and (3) a CTW's depth is inadequate. Here we look at two scenarios with more complexity. All experiments are run on a single computer with an Apple M3 chip and 16 GB of RAM.

5.1 Distribution Shift and Inadequate Context Depth

In this experiment, the underlying environment starts by repeating a length-12 sequence like 110010111000 and then shifts to a new repeating pattern, which is the reverse of the length-12 sequence, at time step 200. We use a depth-3 CTW to learn the environment, which means we have both distribution shift as well as inadequate context depth in this experiment. Figure 2 shows the result of using a prequential martingale with the autocorrelation betting function (referred to as conditional martingale, coloured red) and a prequential martingale with power gambler betting function (referred to as a marginal martingale, coloured blue) to test the depth-3 CTW over 20 runs with different random-number generator seeds. Note that the conditional martingale wins big but it cannot reliably detect the distribution shift at time 200. In contrast, the marginal martingale successfully detects the distribution shift. This experiment confirms the theoretical arguments in § 4.1.

5.2 A Universal Auditor

In the experiments described in Appendix C.3, we show how the autocorrelation betting function can be used to detect a CTW's depth is inadequate. A key issue with the use of the autocorrelation betting function is that, as the order and temporal structure of the underlying n-markov environment increases, there is a combinatorial explosion of autocorrelation betting functions that must be tried. In this experiment, we show how the Transformer-based betting function described in §3.2 can be used instead. The environment generates a data stream where the first 20 bits are randomly generated and, for $i \geq 20$, $x_i = x_{i-4} \oplus x_{i-13} \oplus x_{i-17}$, where $\oplus$ is the XOR operator. A depth-3 CTW is used to predict the sequence. A transformer with context window of 20 is trained sequentially on the prequential p-values produced by the CTW. Fig. 3 shows the result for this Transformer-based martingale over 20 random runs. In most cases, the Transformer model found the XOR pattern in the data after more than 1000 steps and was able to win consistently against the CTW, which failed to learn the pattern given its limited depth. In practice, there may be a need to tune the exact architecture of the Transformer network to achieve desired results.

The transformer's representational power over CTW comes down to two key aspects: (1) Feature Combination: instead of just looking at adjacent strings, attention heads can look at non-contiguous patterns e.g., the 1^{st}, 5^{th}, and n^{th} tokens, and (2) Semantic Generalisation: the embedding layer maps tokens into a continuous space, which allows the model to learn

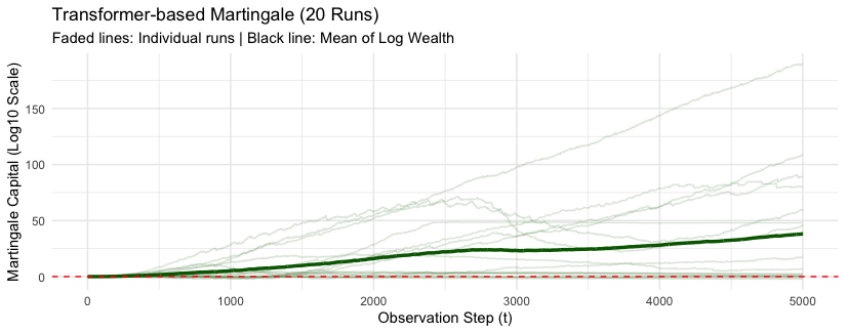

Figure 3: Scores for a martingale that uses a Transformer as the betting function

the semantic similarity between two different contexts. When we use the Transformer as an auditor against the CTW, we are using the Transformer's larger representational power to find complex, non-contiguous, or semantic patterns within the context window that the strict suffix-matching mechanism of the CTW responds slowly to. Once such patterns are found by the transformer, one can use the generalised CTW [53, 52] to dynamically adjust the feature functions used in the internal nodes of a context tree to speed up learning.

6 Discussion and Conclusion

In this paper, we have introduced a suite of test martingales to detect and correct model-misspecification issues in CTW and related Bayesian mixture estimators for online sequence prediction. The use of modern sequential hypothesis testing procedures to check the modelling assumptions of Bayesian estimators seems to be a promising way of getting value out of combining the Bayesian and frequentist approaches to probability. While the general problem remains unsolved, our work consitutues a substantive step in this research program.

A key technique of our approach is the application of randomised ranked PIT to map—under the null hypothesis—discrete predictive distributions into a sequence of independent and uniformly distributed prequential and conformal p-values. We showed that by routing these p-values through a range of betting functions, including adaptive Transformer-based architectures, we can successfully isolate specific failure modes, such as static marginal errors and hidden temporal dependencies in CTW. We note the interest in recent literature [34, 30, 50] in e-values that can bypass the need for PITs. For the purposes of this paper, we find it convenient to construct e-values via betting functions defined over PIT-based p-values. This approach allows us to audit the CTW model for a broad class of potential misspecifications without requiring a pre-specified alternative. However, when a specific alternative hypothesis is available—such as a Transformer model trained on the same data—it is more efficient to design the betting function directly to compare the two models. We demonstrate how to construct such a function in § 3.2.

From an AI safety perspective, our results highlight that while universal predictors like CTW possess strong asymptotic convergence guarantees, their finite-sample behaviour requires active monitoring in high-risk or complex environments. The introduction of an adaptive CTW algorithm that can recalibrate the model and optionally employ defensive forecasting techniques upon martingale alerts also confers some agility and robustness when the true environment deviates from the model class. The general observation here is that, in high-risk AI applications, the problem is often in designing safeguards *around* a learning algorithm. Further, in applications where explainability is a requirement, it is practical to use a non-explainable neural network to audit an explainable model like CTW.

Future work will explore the application of our sequential testing framework in broader reinforcement learning settings, especially in human-AI teaming setups that allow dynamic knowledge injection [52]. Additionally, extending this framework to multi-agent scenarios—where non-stationarity is driven by the shifting strategies of other agents—presents a compelling avenue for integrating martingale-based testing into game-theoretic sequence prediction.

References

[1] Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? Investigations with linear models. *arXiv:2211.15661*, 2022.

[2] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012.

[3] Yu Bai, Fan Chen, Huan Wang, Caiming Xiong, and Song Mei. Transformers as statisticians: Provable in-context learning with in-context algorithm selection. *NeurIPS*, 36:57125–57211, 2023.

[4] Leonard Bereska and Efstratios Gavves. Mechanistic interpretability for AI safety – A review. *arXiv:2404.14082*, 2024.

[5] Hila Chefer, Shir Gur, and Lior Wolf. Transformer interpretability beyond attention visualization. In *CVPR*, pages 782–791, 2021.

[6] A Philip Dawid. Statistical theory the prequential approach. *Journal of the Royal Statistical Society: Series A (General)*, 147(2):278–290, 1984.

[7] A. Philip Dawid and Vladimir G. Vovk. Prequential probability: principles and properties. *Bernoulli*, 5(1):125–162, 1999.

[8] Richard M Dudley. *Real analysis and probability*. Chapman and Hall/CRC, 2018.

[9] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[10] Haim Gaifman and Marc Snir. Probabilities over rich languages, testing and randomness. *The Journal of Symbolic Logic*, 47(3):495–548, 1982.

[11] Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. What can transformers learn in-context? A case study of simple function classes. *NeurIPS*, 35:30583–30598, 2022.

[12] Tilmann Gneiting, Fadoua Balabdaoui, and Adrian E Raftery. Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 69(2):243–268, 2007.

[13] Jordi Grau-Moya et al. Learning universal predictors. In *ICML*, pages 16178–16205, 2024.

[14] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(1):723–773, 2012.

[15] Peter Grünwald and Thijs Van Ommen. Inconsistency of Bayesian inference for misspecified linear models, and a proposal for repairing it. *Bayesian Analysis*, 12:1069–1103, 2017.

[16] Marcus Hutter. *Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability*. Springer, Berlin, 2005.

[17] Marcus Hutter. Testing independence of exchangeable random variables. *arXiv:2210.12392*, 2022.

[18] Marcus Hutter, John W Lloyd, Kee Siong Ng, and William TB Uther. Probabilities on sentences in an expressive logic. *Journal of Applied Logic*, 11(4):386–420, 2013.

[19] Marcus Hutter, David Quarel, and Elliot Catt. *An Introduction to Universal Artificial Intelligence*. CRC Press, 2024.

[20] Ehud Kalai and Ehud Lehrer. Rational learning leads to Nash equilibrium. *Econometrica: Journal of the Econometric Society*, pages 1019–1045, 1993.

[21] R.E. Krichevsky and V.K. Trofimov. The performance of universal coding. *IEEE Transactions on Information Theory*, IT-27:199–207, 1981.

[22] Jan Leike, Jessica Taylor, and Benya Fallenstein. A formal solution to the grain of truth problem. In *UAI*, pages 427–436, 2016.

[23] Laurent Mazliak and Glenn Shafer. *The Splendors and Miseries of Martingales: Their History from the Casino to Mathematics*. Springer, 2022.

[24] Alexander Meulemans, Rajai Nasser, Maciej Wołczyk, Marissa A Weis, Seijin Kobayashi, Blake Richards, Guillaume Lajoie, Angelika Steger, Marcus Hutter, and James Manyika. Embedded universal predictive intelligence: a coherent framework for multi-agent learning. *arXiv:2511.22226*, 2025.

[25] Kee Siong Ng, Samuel Yang-Zhao, and Timothy Cadogan-Cowper. The problem of social cost in multi-agent general reinforcement learning: Survey and synthesis. *arXiv:2412.02091*, 2025.

[26] Jorge Pérez, Pablo Barceló, and Javier Marinkovic. Attention is Turing-complete. *Journal of Machine Learning Research*, 22(75):1–35, 2021.

[27] Wouter Koolen Peter Grünwald, Rianne de Heide. Safe testing. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 86:1091–1128, 2024.

[28] Aleksandr Podkopaev and Aaditya Ramdas. Tracking the risk of a deployed model and detecting harmful distribution shifts. In *ICLR*. OpenReview, 2022.

[29] Nicholas G Polson, Vadim Sokolov, and Daniel Zantedeschi. Bayes, e-values and testing. *arXiv:2602.04146*, 2026.

[30] Aaditya Ramdas, Peter Grünwald, Vladimir Vovk, and Glenn Shafer. Game-theoretic statistics and safe anytime-valid inference. *Statistical Science*, 38(4):576–601, 2023.

[31] Aaditya Ramdas and Ruodu Wang. Hypothesis testing with e-values. *Foundations and Trends® in Statistics*, 1(1-2):1–390, 2025.

[32] Samuel Rathmanner and Marcus Hutter. A philosophical treatise of universal induction. *Entropy*, 13(6):1076–1136, 2011.

[33] Murray Rosenblatt. Remarks on a multivariate transformation. *The Annals of Mathematical Statistics*, 23(3):470–472, 1952.

[34] Glenn Shafer. Testing by betting: A strategy for statistical and scientific communication. *Journal of the Royal Statistical Society Series A*, 184:407–431, 2021.

[35] Shubhanshu Shekhar and Aaditya Ramdas. Nonparametric two-sample testing by betting. *IEEE Transactions on Information Theory*, 70(2):1178–1203, 2023.

[36] JQ Smith. Diagnostic checks of non-standard time series models. *Journal of Forecasting*, 4(3):283–291, 1985.

[37] Ray J Solomonoff. A formal theory of inductive inference. Part I. *Information and control*, 7(1):1–22, 1964.

[38] Ray J Solomonoff. A formal theory of inductive inference. Part II. *Information and control*, 7(2):224–254, 1964.

[39] Tim van Erven, Peter Grunwald, Nishant A Mehta, Mark Reid, and Robert Williamson. Fast rates in statistical and online learning. *Journal of Machine Learning Research*, 2015.

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[41] Badri N. Vellambi and Marcus Hutter. Convergence of binarized context-tree weighting for estimating distributions of stationary sources. In *IEEE International Symposium on Information Theory*, pages 731–735, 2018.

[42] Joel Veness, Kee Siong Ng, Marcus Hutter, and Michael H. Bowling. Context tree switching. In James A. Storer and Michael W. Marcellin, editors, *2012 Data Compression Conference*, pages 327–336. IEEE, 2012.

[43] Joel Veness, Kee Siong Ng, Marcus Hutter, William Uther, and David Silver. A Monte-Carlo AIXI approximation. *Journal of Artificial Intelligence Research*, 40:95–142, 2011.

[44] Joel Veness, Martha White, Michael Bowling, and András György. Partition tree weighting. In Ali Bilgin, Michael W. Marcellin, Joan Serra-Sagristà, and James A. Storer, editors, *2013 Data Compression Conference*, pages 321–330. IEEE, 2013.

[45] Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In *ICML*, pages 35151–35174, 2023.

[46] Vladimir Vovk. Derandomizing stochastic prediction strategies. In *Proceedings of the Annual Conference on Computational Learning Theory*, pages 32–44, 1997.

[47] Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*. Springer, 2nd edition, 2022.

[48] Vladimir Vovk, Ilia Nouretdinov, and Alex Gammerman. Conformal e-testing. *Pattern Recognition*, page 111841, 2025.

[49] Vladimir Vovk, Ivan Petej, and Alex Gammerman. Protected probabilistic classification. In *Conformal and Probabilistic Prediction and Applications*, pages 297–299. PMLR, 2021.

[50] Vladimir Vovk and Ruodu Wang. E-values: Calibration, combination and applications. *The Annals of Statistics*, 49(3):1736–1754, 2021.

[51] Frans MJ Willems, Yuri M Shtarkov, and Tjalling J Tjalkens. The context-tree weighting method: Basic properties. *IEEE Transactions on Information Theory*, 41(3):653–664, 1995.

[52] Samuel Yang-Zhao, Kee Siong Ng, and Marcus Hutter. Dynamic knowledge injection for AIXI agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38(15), pages 16388–16397, 2024.

[53] Samuel Yang-Zhao, Tianyu Wang, and Kee Siong Ng. A direct approximation of AIXI using logical state abstractions. *Advances in Neural Information Processing Systems*, 35:36640–36653, 2022.

[54] Ruiqi Zhang, Spencer Frei, and Peter L Bartlett. Trained transformers learn linear models in-context. *Journal of Machine Learning Research*, 25(49):1–55, 2024.

[55] Yufeng Zhang, Fengzhuo Zhang, Zhuoran Yang, and Zhaoran Wang. What and how does in-context learning learn? Bayesian model averaging, parameterization, and generalization. *arXiv:2305.19420*, 2023.

A Technical Notes

A.1 The iid Property of P-values

Variations of the following result have been known since at least [6, 36].

Proposition 2. If the CTW's predictive distribution is the true data-generating distribution, then the randomised Rosenblatt transformation given by Equation (6), produces a uniform distribution.

Proof. We first show, at every time step t having observed history $h_{<t} = x_{1:t-1}$, we have $P(u_t \leq z \mid h_{<t}) = z$ for any $z \in [0,1]$. Let $X = \{o_1, o_2, \ldots, o_m\}$ be the observation space and let $p_i = P(o_i \mid h_{<t})$ denote the predicted probability of the symbol o_i from the CTW model. Assume without loss of generality that, for each $i \in (1, m-1)$, we have $p_i \leq p_{i+1}$, this may be achieved via a suitable permutation of the indices of X. Now, by the Law of Total Probability, for all $z \in [0,1]$, we have

$$\begin{aligned} P(u_t \leq z \mid h_{<t}) &= \sum_i P(u_t \leq z \mid h_{<t}o_i) P(o_i \mid h_{<t}) \\ &= \sum_{i<i^*} P(u_t \leq z \mid h_{<t}o_i) p_i + \sum_{i>i^*} P(u_t \leq z \mid h_{<t}o_i) p_i + P(u_t \leq z \mid h_{<t}o_{i^*}) p_{i^*}, \end{aligned} \tag{12}$$

where $i^* \in (1, m)$ is the (unique) index satisfying $\sum_{i \leq i^*} p_i \leq z < \sum_{i \leq (i^*+1)} p_i$. We now examine each term in (12). The first term in (12) simplifies to $\sum_{i<i^*} p_i$ because for each $i < i^*$, when conditioned on $h_{<t}o_i$, we have from (6)

$$u_t = \sum_{j<i} p_j + \tau_t \cdot p_i \leq \sum_{j \leq i} p_j \leq \sum_{j \leq i^*} p_j \leq z$$

so $P(u_t \leq z \mid h_{<t}o_i) = 1$. By a similar reasoning, the second term in (12) evaluates to 0 because, for each $i > i^*$, we have $P(u_t > z \mid h_{<t}o_i) = 1$. Consider now the third term in (12). We have

$$\begin{aligned} P(u_t \leq z \mid h_{<t}o_{i^*}) &= P\left[\sum_{i<i^*} p_i + \tau_t \cdot p_{i^*} \leq z\right] \\ &= P\left[\tau_t \leq \frac{z - \sum_{i<i^*} p_i}{p_{i^*}}\right] = \frac{z - \sum_{i<i^*} p_i}{p_{i^*}}, \end{aligned} \tag{13}$$

thus yielding $P(u_t \leq z \mid h_{<t}o_{i^*}) p_{i^*} = z - \sum_{i<i^*} p_i$. Summing up all the terms, we now have

$$P(u_t \leq z \mid h_{<t}) = \sum_{i<i^*} p_j + z - \sum_{i<i^*} p_i = z. \tag{14}$$

Chaining together (14), we can now show that $u_{1:t}$ is independently and uniformly distributed since, for all $z_{1:t}$,

$$\begin{aligned} &P(u_1 \leq z_1, u_2 \leq z_2, \ldots, u_t \leq z_t \mid x_{1:t-1}) \\ &\qquad = P(u_1 \leq z_1) P(u_2 \leq z_2 \mid x_1) \cdots P(u_t \leq z_t \mid x_{1:t-1}) = \prod_i z_i. \end{aligned} \tag{15}$$

□

Remark 1. It is not hard to see that Proposition 2 does not depend on the specific properties of CTW in any way. In fact, the result would work for any blackbox model. This means the conformal e-testing procedure described in Section 3.1 can be applied for essentialy any blackbox model. Having said that, CTW is one of those algorithms where all the computations required by Proposition 2 can be done exactly and efficiently, by virtue of (4) and (5).

The following are two examples to give some intuition on what can happen to the u_t values when the premise of Proposition 2 does not hold.

Example 1. Here is a scenario where the CTW model cannot represent the underlying environment μ. Suppose $\mu(1 \mid h) = 1$ whenever there is an even number of 0s in the history h, and $\mu(1 \mid h) = 0.4$ otherwise. The ratio of 1s to 0s in the resultant string will converge to around 57% to 43%. Using a depth-0 CTW i.e., a KT estimator, we will thus have $P_{kt}(1 \mid h) = 0.57$ whenever h has an even number of zero's. To calculate u_t, we sample an x_t from $\mu(\cdot \mid h)$, which yields $x_t = 1$ with probability 1. This means the resultant u_t value will always be in the range $[0.43, 1]$, so u_t is definitely not uniformly distributed in $[0, 1]$.

Example 2. Suppose the underlying environment μ generates the data stream by first repeating the string '10' five times, then repeating the string '10010' indefinitely thereafter. Suppose the depth of the CTW is 2. In that scenario, for every time step $t > 10$ where $(t-10) \,\%\, 5 = 1$, we have $\mu(x_t = 1 \mid h) = 1$ and $P_{CTW}(x_t = 1 \mid h)$ at around 0.5 but strictly larger than 0.5. This means all these u_t values are in the range $[0.5, 1]$ and thus correlated. In other words, they are not independent.

Proposition 3 ([47])**.** Let $\alpha_{1:n}$ be a sequence of non-conformity scores that is exchangeable. Then the sequence of conformal p-values $p_{1:n}$ computed from $\alpha_{1:n}$ using Formula (7) are mutually independent and each p_t is uniformly distributed in $[0, 1]$.

A.2 The Isolating Power of Marginal Martingales

Here is the result that shows the ability of marginal prequential martingales to isolate distribution shifts.

Proposition 4. Let the true environment be a stationary, ergodic K-markov process over a finite alphabet $\mathcal{X}$. Let the forecasting model be a CTW with maximum depth D, where $D < K$, and let u_t be the prequential p-values generated from the CTW model using (6). As $t \to \infty$, we have $\mathbb{E}[b(u_t)]$ converging to 1 at the rate of $O(\frac{\log t}{t})$, where $b()$ is the power gambler betting function.

Proof. (Sketch) Given the environment is stationary and ergodic, there exists a stationary distribution $\pi_K(r)$ for any sequence r of length K, which can be used to define a stationary distribution $\pi_m(s)$ for any string s of length $m < K$ via marginalisation as follows

$$\pi_m(s) = \sum_{a \in \mathcal{X}^{K-m}} \pi_K(sa).$$

For any specific context s of length $m \leq D$, there is then a true conditional probability for the next symbol given by

$$P_\pi(x \mid s) = \frac{\pi_{m+1}(sx)}{\pi_m(s)}.$$

Let $Q_t(\cdot \mid s)$ be the KT estimator located at node s in the contex tree at time t. By the convergence property of the KT estimator, we have $Q_t(x \mid s) \to P_\pi(x \mid s)$ at the rate of $O(\frac{\log t}{t})$.

Now, for any specific context s of length $m = D$, the CTW predictive distribution at time t is given by

$$Q_t^W(x \mid s) = \sum_{T \in \mathcal{T}_D} w_t(T) Q_t^T(x \mid s),$$

where $\mathcal{T}_D$ is the set of all predictive suffix trees (PSTs) of depth at most D, $w_t(T)$ is the posterior probability assigned to T given all the data seen so far, and $Q_t^T(x \mid s)$ is the prediction made by T, which maps s to the leaf node $T(s)$ in T matching its suffix. By the Shannon-McMillan-Breiman Theorem for stationary ergodic processes and the convergence property of CTW, we have

$$Q_t^W(x \mid s) \to Q_t^{T^*}(x \mid s) \to P_\pi(x \mid T^*(s)) = P_\pi(x \mid s)$$

at the rate of $O(\frac{|T^*| \log t}{t})$, where $T^* \in \mathcal{T}_D$ is the depth D PST with the smallest KL divergence from the true unknown K-markov environment satisfying the property $w_t(T^*) \to 1$, and $|T^*|$ denotes the number of leaf nodes in T^*.

Having established the convergence characteristics of the CTW model, we next look at the statistical property of the conformal p-values generated from the CTW. At time t, given a specific context s_t and a true observed symbol x, the value u_t is sampled uniformly from an interval $\text{Interval}(x \mid s_t)$ of width $Q_t^W(x \mid s_t)$. Therefore, the conditional probability density function of u_t, given both the context s_t and the symbol x, is:

$$\text{Density}(u_t \mid x, s_t) = \frac{1}{Q_t^W(x \mid s_t)} \mathbb{I}(u_t \in \text{Interval}(x \mid s_t)),$$

where $\mathbb{I}$ is the indicator function. To find the probability density of u_t given only the context s_t, we sum over all possible symbols, weighted by their true probability of occurring in that context:

$$f(u \mid s_t) = \sum_{x \in \mathcal{X}} P_\pi(x \mid s_t) \cdot \text{Density}(u \mid x, s_t) \tag{16}$$

$$= \sum_{x \in \mathcal{X}} P_\pi(x \mid s_t) \left(\frac{1}{Q_t^W(x \mid s_t)} \right) \mathbb{I}(u \in \text{Interval}(x \mid s_t)). \tag{17}$$

Substituting the CTW convergence property ($Q_t^W \to P_\pi$) into Equation (17) yields

$$f(u \mid s_t) \to \sum_{x \in \mathcal{X}} \mathbb{I}(u \in \text{Interval}(x \mid s_t)) = 1,$$

where the last step follows from the fact that the intervals form a partition of $[0, 1]$ so each u will fall into exactly one interval. To find the unconditional marginal density $g(u)$ across all time steps, we marginalise over all possible contexts as follows

$$g(u) = \sum_{s \in \mathcal{X}^D} \pi(s) f(u \mid s).$$

Since $f(u \mid s) \to 1$ for every possible context s, we have

$$g(u) \to \sum_{s \in \mathcal{X}^D} \pi(s) \times 1 = 1.$$

Since the power gambler betting function $b(u_t) = \kappa u_t^{\kappa - 1}$ does not look at the context s_t, its expected value can be obtained from the density $g(u) = 1$:

$$\mathbb{E}[b(u_t)] = \int_0^1 \kappa u_t^{\kappa - 1} du = 1.$$

Because the expected value of the bet is 1 (after the CTW model has converged), the martingale property holds, and the test will not exponentially diverge beyond that point. □

We have shown that the marginal prequential martingale is blind to inadequate context depth. It will only be triggered if the true environment shifts, breaking the stationary probabilities and causing the KT estimators in the CTW model to suddenly fail and become uncalibrated.

A.3 The Relative Power of Different Test Martingales

In sequential testing, the power of a test martingale is given by the Kelly criterion, which is the expected log-growth rate of the martingale. Under the alternative hypothesis H_1, the most powerful test martingale is the one that maximises the expected stepwise logarithmic accumulation of wealth:

$$W = \mathbb{E}_{H_1}[\log f_t(X_t)]$$

A martingale with a higher expected log-growth rate will cross any rejection threshold (e.g., $1/\alpha$) exponentially faster, requiring fewer samples to confidently detect the misspecification.

Proposition 5. If the underlying environment exhibits temporal dependence such that the prequential p-values u_t generated by a forecasting model is stationary but non-independent, then the optimal expected log-growth rate of a prequential test martingale strictly dominates the optimal expected log-growth rate of a conformal test martingale.

Proof. (Sketch) Let the base forecasting model generate a sequence of prequential variables $U = (u_1, u_2, \dots)$. We have the following hypotheses.

- The Null Hypothesis (H_0): The model is perfectly specified. The sequence U consists of independent and identically distributed (idd.) variables drawn from $U(0, 1)$.
- The Alternative Hypothesis (H_1): The data has a hidden temporal structure (e.g., a Markov dependency). The sequence U is stationary and ergodic, but not independent. Therefore, the true conditional density $q(u_t \mid u_1, \dots, u_{t-1})$ is not equal to the marginal density $q(u_t)$.

For the prequential test martingale $M_N^{\text{preq}} = \prod_{t=1}^{N} f_t(u_t)$, the betting function f_t is parameterised by the exact, ordered history $u_{<t} = (u_1, \dots, u_{t-1})$. Now, the optimal bet f_t^* that maximises log-growth is the true density ratio between H_1 and H_0. Since the H_0 density is 1 (Uniform), the optimal bet is exactly the true conditional density: $f_t^*(u_t) = q(u_t \mid u_{<t})$. The maximum expected log-growth rate per step is the conditional KL divergence:

$$W_{\text{preq}} = \mathbb{E}_{H_1}[\log q(u_t \mid u_{<t}) = -h(u_t \mid u_1, \dots, u_{t-1}],$$

where h is the negative conditional differential entropy function.

For the conformal test martingale $M_N^{\text{conf}} = \prod_{t=1}^{N} g_t(p_t)$, the input sequence $p_1, p_2, \dots$ is generated by applying a non-conformity measure and an empirical rank transformation (see (7)). Let this deterministic transformation be denoted as a function C, such that the conformal p-value is $p_t = C_t(u_t, \{u_1, \dots, u_{t-1}\})$. Because the bag is an unordered multiset, the chronological filtration is destroyed. The conformal betting function g_t can only be parameterised by the transformed sequence $p_{<t} = (p_1, \dots, p_{t-1})$. The maximum expected log-growth rate for the conformal martingale is similarly bound by the entropy of this transformed sequence:

$$W_{\text{conf}} = -h(p_t \mid p_1, \dots, p_{t-1}).$$

We next show that $W_{\text{preq}} > W_{\text{conf}}$ when a temporal dependency exists. Because a temporal dependency exists under H_1, knowing the exact chronological order of the past reduces the uncertainty of the next observation. Therefore, the conditional entropy is strictly less than the marginal entropy:

$$h(u_t \mid u_1, \dots, u_{t-1}) < h(u_t).$$

The conformal transformation C acts as a deterministic channel that maps the ordered sequence $(u_1, \dots, u_t)$ to p_t. By definition, mapping an ordered sequence into an unordered multiset is a many-to-one mapping so it is a lossy compression. The Data Processing Inequality (DPI) states that post-processing cannot increase information. If $X \to Y \to Z$ forms a Markov chain, then the mutual information $I(X; Y) \geq I(X; Z)$. In our setup, the true temporal signal exists in the chronological sequence U. The conformal sequence is a post-processed derivation of U. Therefore, the conformal p-values cannot contain more information about the underlying generative process than the raw prequential variables:

$$I(p_t; p_{<t}) \leq I(u_t; u_{<t}).$$

Because the mutual information (the predictable signal) is strictly bounded, the conditional entropy of the conformal sequence must be greater than or equal to the conditional entropy of the prequential sequence

$$h(p_t \mid \mathcal{P}_{t-1}) \geq h(U_t \mid \mathcal{H}_{t-1}).$$

Substituting this into the log-growth equations yields $W_{\text{preq}} \geq W_{\text{conf}}$. □

A.4 The Possible Failure Modes of CTW

There are three possible failure modes for a predictive model: wrong marginal, wrong conditional, or non-stationarity. We now examine how these can occur in a CTW model.

Static Misspecification (Wrong Marginal): This is the scenario where the true data-generating process is stable and the depth of the CTW is adequate to model any temporal dependencies in the underlying process but somehow the predictive next-symbol marginal distribution is wrong. Fortunately, this scenario is actually only possible in a small-sample scenario, because of how CTW's weighting and estimation procedures work. First of all, since the Jeffreys prior in the KT estimator essentially adds half a count to every possible symbol, the KT estimator has a bias to pull predictions away from 0 or 1 toward 0.5 when the data size is small. Another source of bias in CTW is that it does not use the best PST for its prediction but computes a weighted average of the predictive probabilities of all the PSTs of a certain maximum depth. This weighting mechanism can cause the predictive marginal to be incorrectly estimated when there is only a small data sample to learn from.

Temporal Dependency (Wrong Conditional): In this scenario, the CTW model's marginal calibration is fine, but the model misses a sequential pattern (e.g., a Markov dependency, or a hidden cyclic state). This can certainly occur when the depth of the CTW is not long enough to represent the sequential pattern in the underlying data-generating process. CTW has a fixed maximum depth D. If a context tree has depth 8, but the data has a dependency of length 12 (like a long period or a specific structural repeat), then to the CTW, the data looks like random noise because it cannot see the context. In that scenario, it will output a flat, high-entropy distribution. Because the pattern is actually there, the u_t sequence will not be independent; it will be autocorrelated.

Distribution Shift (Break in Exchangeability) This is the scenario where the data-generating process changes abruptly over time. CTW is a cumulative learner and, in its standard form, never forgets old data. If the data source undergoes a regime shift (e.g., the probability of 1 moves from 0.1 to 0.9), the CTW will need a long time for the new data to offset the old counts that are no longer accurate. In particular, during this transition, the CTW will consistently under-predict 1s and the u_t values will clump toward the high end.

The above provides the motivation for using a variety of betting functions when constructing the test martingale. If the goal is just to detect the existence of model misspecification issues, the various betting functions can be combined into a mixture function that can provide large coverage. The Aggregating Algorithm [46] can be used to compute such mixture e-martingales efficiently, as shown in the Composite Jumper Martingale algorithm in [47, Chap 10]. However, if the goal is not just detection but also include diagnosis, then the various betting functions should be used in multiple martingales that are run in parallel.

B An Active Recalibration Algorithm

Algorithm 1 leverages the misspecification detection and diagnostic techniques developed in section 3.1 to actively correct misspecification in CTW. It functions by tracking the growth of the cumulative redundancy (a notion introduced in Proposition 6, which allows us to detect misspecification) of a given CTW C_D, as the depth D is restricted. In particular, the cumulative redundancy of C_D, C_{D-1}, C_{D-2} on an observed sequence x is computed, the decision to extend the depth—and hence correct misspecification—is made on this basis. Exactly how this is to be accomplished, and the exact justification of this procedure comprises the bulk of this section. We now detail the justification of Algorithm 2, beginning with a few preliminaries.

Definition 2 (n-Markov Rules)**.** Let $k : \mathbb{B}^n \to \mathbb{B}$ be a transition function such that given an initial seed $s \in \mathbb{B}^n$, the function k constructs an infinite sequence $x \in \mathbb{B}^\infty$ by induction. If k satisfies the following recurrence relation:

$$x_i = k(x_{i-n}, \ldots, x_{i-1}) \quad \text{for } i \geq n,$$

then the function k is considered an n-Markov rule. The collection of all such n-Markov rules we denote K_n, and define $K_n \coloneqq \{k : \mathbb{B}^n \to \mathbb{B}\}$.

We note a slight ambiguity in stating that an n-Markov rule k constructs an infinite sequence x. Strictly speaking, the sequence depends on both k and the seed s. In general, different seeds may lead to different infinite sequences. However, since the seed affects only the initial

n symbols, and all results below concern finite-prefix probabilities or asymptotic behaviour in m, this dependence will be suppressed in the notation.

Definition 3 (Induced Markov Measure)**.** Each $k \in K_n$ induces a (degenerate) stochastic process p_k defined via the conditional probabilities

$$p_k(x_i \mid x_{<i}) = \begin{cases} 1 & \text{if } x_i = k(x_{i-n}, \ldots, x_{i-1}), \\ 0 & \text{otherwise,} \end{cases} \qquad \text{for } i \geq n.$$

This defines a probability measure on finite prefixes via

$$p_k(x_{1:m}) = \mathbf{1}\{x_{1:n} = s\} \cdot \prod_{i=n+1}^{m} p_k(x_i \mid x_{<i}),$$

where the first n bits are fixed by the seed. We denote the collection of all such measures by

$$P_n \coloneqq \{p_k \mid k \in K_n\}.$$

Recall that we denote the root conditional probability mass computed by CTW on some bit $x_i \in \mathbb{B}$, given some context $x_{<i}$ as, $P_w^\epsilon(x_i \mid x_{<i})$. Recall also that we denote the context-weighted tree of depth D, as C_D. When used as a parameter, we will—in an abuse of notation—substitute P_w^ϵ for D. No ambiguity results, since the behaviour of the mixture P_w^ϵ is determined by the depth D of the associated CTW C_D. Similarly, and for like reasons, we substitute P_n for n. We now prove a few simple lemmas.

Proposition 6 (Cumulative Redundancy)**.** Let $k \in K_n$, and let $x_{1:m}$ be the prefix generated by k from some seed. Let $p_k \in P_n$ be the induced measure. Then the cumulative redundancy of P_w^ϵ on $x_{1:m}$ with respect to p_k is

$$R_{D,p_k}(x_{1:m}) = \sum_{i=1}^{m} -\log_2 P_w^\epsilon(x_i \mid x_{<i}).$$

Proof. By definition of cumulative redundancy,

$$R_{D,p_k}(x_{1:m}) = \sum_{i=1}^{m} \log_2 \frac{p_k(x_i \mid x_{<i})}{P_w^\epsilon(x_i \mid x_{<i})}.$$

Since $x_{1:m}$ is generated by k, we have $p_k(x_i \mid x_{<i}) = 1$ for all i, hence

$$R_{D,p_k}(x_{1:m}) = \sum_{i=1}^{m} \log_2 \frac{1}{P_w^\epsilon(x_i \mid x_{<i})} = \sum_{i=1}^{m} -\log_2 P_w^\epsilon(x_i \mid x_{<i}).$$

□

Proposition 7 (Expectation of Cumulative Redundancy)**.** Let P_n be equipped with the uniform prior over K_n. For each $k \in K_n$, let $x_{1:m}^{(k)}$ denote the prefix generated by k. Then the expected cumulative redundancy satisfies

$$\mathbb{E}_m(n \parallel D) = \frac{1}{2^{2^n}} \sum_{k \in K_n} \sum_{i=1}^{m} -\log_2 P_w^\epsilon\big(x_i^{(k)} \mid x_{<i}^{(k)}\big).$$

Proof. By definition of expectation under the uniform prior,

$$\mathbb{E}_m(n \parallel D) = \frac{1}{|K_n|} \sum_{k \in K_n} R_{D,p_k}(x_{1:m}^{(k)}).$$

Since $|K_n| = 2^{2^n}$ and by Proposition 6,

$$\mathbb{E}_m(n \parallel D) = \frac{1}{2^{2^n}} \sum_{k \in K_n} \sum_{i=1}^{m} -\log_2 P_w^\epsilon(x_i^{(k)} \mid x_{<i}^{(k)}).$$

□

Proposition 8 (Ensemble Symmetry)**.** Let K_n be equipped with the uniform distribution, and let P_n be the induced collection of measures. For a CTW C_D with $D < n$, we have

$$\mathbb{E}_{P_n}[P_w^\epsilon(x_i = 1 \mid x_{<i})] = \frac{1}{2}.$$

Proof. The collection K_n is the set of all functions $k : \mathbb{B}^n \to \mathbb{B}$ satisfying the n-Markov rule. This set is symmetric under bit-inversion; for every rule $k \in K_n$, there exists a unique "dual" rule $k' \in K_n$ such that $k'(x) = 1 - k(x)$ for all $x \in \mathbb{B}^n$.

Since $D < n$, the CTW context $x_{<i}$ does not uniquely determine the underlying n-bit state. Under the uniform distribution over K_n, the induced processes are symmetric with respect to exchanging 0 and 1.

At each leaf, the KT estimator satisfies $P_{KT}(1 \mid a, b) = P_{KT}(0 \mid b, a)$ so under the symmetric distribution of counts, we obtain

$$\mathbb{E}_{P_n}[P_{KT}(1 \mid a, b)] = \frac{1}{2}.$$

As the CTW root probability P_w^ϵ is a recursive linear mixture of these symmetric leaf estimators, the expectation is preserved:

$$\mathbb{E}_{P_n}[P_w^\epsilon(x_i = 1 \mid x_{<i})] = \frac{1}{2^{2^n}} \sum_{p \in P_n} P_w^\epsilon(x_i = 1 \mid x_{<i}) = \frac{1}{2}$$

□

Proposition 9 (Structural Lemma)**.** [51] Let C_D be a context tree of depth D and P_w^ϵ the associated CTW mixture. For any sequence $x_{1:m}$, we have

$$-\log_2 P_w^\epsilon(x_{1:m}) \leq \sum_{s \in \mathbb{B}^D} -\log_2 P_{KT}(x_s) + O(2^D),$$

where x_s denotes the subsequence of symbols assigned to context s, and $P_{KT}(x_s)$ is the Krichevsky–Trofimov probability of that subsequence.

Corollary 10. If the full-depth tree C_D is (asymptotically) the optimal model in the CTW mixture for the sequence $x_{1:m}$, then

$$-\log_2 P_w^\epsilon(x_{1:m}) = \sum_{s \in \mathbb{B}^D} -\log_2 P_{KT}(x_s) + O(2^D).$$

We note, if $D < n$ then full-depth tree C_D is asymptotically the best model. (The argument is similar to the first part of the proof of Proposition 4.)

We now prove the central lemma of this section; providing bounds on the Expected Cumulative Redundancy $\mathbb{E}_m(n \parallel D)$ in the specified and misspecified cases. In an attempt to thwart possible confusion; we note that the quantity $\mathbb{E}_m(n \parallel D)$ is directly related to the KL-divergence referenced in Theorem 1. However, while the KL-divergence captures the expected average error per symbol, the $\mathbb{E}_m(n \parallel D)$ term captures the expected total accumulated errors.

Proposition 11 (Expected Redundancy Transition)**.** Let P_n be equipped with the uniform prior over K_n. For each $k \in K_n$, let $x_{1:m}^{(k)}$ denote the prefix generated by k. Then, for large enough m the expected cumulative redundancy satisfies:

$$\mathbb{E}_m(n \parallel D) = \begin{cases} m + O(2^D \log_2(m)) & \text{if } D < n \\ \log_2(m) \cdot 2^n & \text{if } D \geq n \end{cases}$$

Proof. In the case that $D \geq n$ it is a well-known fact that $\mathbb{E}(n \parallel D) = 2^n \log_2(m)$ [51]; and this is because the CTW mixture contains every n-Markov model (see also Theorem 1). Thus, we only provide a proof for the equality in the misspecified case, that is where $D < n$.

We establish the result by deriving matching upper and lower bounds for $\mathbb{E}_m(n \parallel D)$. We begin with the equality established in Proposition 7:

$$\begin{aligned}\mathbb{E}_m(n \parallel D) &= \frac{1}{2^{2^n}} \sum_{k \in K_n} \sum_{i=1}^{m} -\log_2 P_w^{\epsilon}(x_i^{(k)} \mid x_{<i}^{(k)}) \\ &= \sum_{i=1}^{m} \frac{1}{2^{2^n}} \sum_{k \in K_n} -\log_2 P_w^{\epsilon}(x_i^{(k)} \mid x_{<i}^{(k)}),\end{aligned}$$

by interchange of finite sums.

We first establish a lower bound. By Jensen's inequality, since $-\log_2(\cdot)$ is convex,

$$\frac{1}{2^{2^n}} \sum_{k \in K_n} -\log_2 P_w^{\epsilon}(x_i^{(k)} \mid x_{<i}^{(k)}) \;\geq\; -\log_2 \left(\frac{1}{2^{2^n}} \sum_{k \in K_n} P_w^{\epsilon}(x_i^{(k)} \mid x_{<i}^{(k)}) \right).$$

By Proposition 8, the inner expectation equals $1/2$, hence

$$\frac{1}{2^{2^n}} \sum_{k \in K_n} -\log_2 P_w^{\epsilon}(\cdot) \geq 1.$$

Summing over i yields

$$\mathbb{E}_m(n \parallel D) \geq m.$$

Since the CTW redundancy bound is an individual-sequence bound, and hence uniform over all sequences $x_{<i}^{(k)}$, applying it to any sequence $x_{1:m}$ yields

$$-\log_2 P_w^{\epsilon}(x_{1:m}) = m + O(2^D \log_2 m),$$

when the underlying process is not representable at depth D. Since this bound holds uniformly over all $k \in K_n$ when $D < n$, averaging over K_n yields

$$\mathbb{E}_m(n \parallel D) \leq m + O(2^D \log_2 m).$$

Since $\mathbb{E}_m(n \parallel D) \geq m$ and we have established that $\mathbb{E}_m(n \parallel D) \leq m + O(2^D \log_2 m)$, the desired equality follows from the definition of asymptotic upper bounds

$$\mathbb{E}_m(n \parallel D) = m + O(2^D \log_2 m).$$

□

We have now established the behaviour of $\mathbb{E}_m(n \parallel D)$ both when $D < n$, and when $D \geq n$. On the strength of Proposition 8, we conjecture that the phase transition between these cases is sharp occuring at $n - D = 2 \pm 1$, but application-specific empirical work are required to exactly determine this. These facts motivate the Active Recalibration Algorithm 2.

Algorithm 2 Detection and Calibration via Redundancy

Input : $\eta \in \mathbb{N}$
Initialise : $C_{D=3}$, $x = \epsilon$, $R = 0$

For each new observation x_i **:**
  $x = xx_i$
  C_D.`updateCTW`(x_i) ▷ Optional
  $R = R - \log_2[P_w^{\epsilon}(x_i)]$
  $C_{D-1} =$ `dropLayer`$(C_D, 1)$
  $C_{D-2} =$ `dropLayer`$(C_D, 2)$
  $R1 =$ `redun`(C_{D-1}, x)
  $R2 =$ `redun`(C_{D-2}, x)

  If $\frac{R-R1}{|x|} < \frac{R1-R2}{|x|}$ **:** ▷ Sub-linearity Check
    C_D.`extend`$(D+1)$
  Else :
    C_D.`extend`$(D+\eta)$

As can be seen, the active calibration algorithm 2 takes as input a learning rate η defined by the user. It is initialised with a CTW of depth 3, denoted $C_{D=3}$, an empty observation string $x = \epsilon$, and set the cumulative redundancy $R = 0$. This initialisation is given for the sake of completeness, but the body of the algorithm, can be executed on arbitrary initial values provided that the depth of the CTW is at least three.

We now give an outline in plainterms of the body of the algorithm. With the exception of `extend` and `dropLayer` function – which we deal with separately – the following outline is sufficient to resolve any ambiguity in the reading of Algorithm 1. Line (5) concatenates the new observation at time i, denoted x_i, to the cumulative history x. Line (6) updates the node values of C_D to incorporate the new evidence x_i, we note that this line is optional in the case that C_D has been pre-trained upto convergence (for exact details on how this is done see [51]). Line (7) updates the redundancy R of C_D with respect to x (see Proposition 6 for details). Lines (8–9) restrict C_D to a tree of depth $D-1$ and $D-2$ respectively. Lines (10–11) compute the total redundancy $R1$, and $R2$ of the restricted trees C_{D-1}, and C_{D-2}. Based on the linearity check, Lines (13—16) increment the depth of C_D either by 1 or according to the exploratory learning rate η

We now detail the sub-procedures `extend`, and `dropLayer`. The `extend(`n`)` procedure takes the CTW C_D initiliased on line (1) and extends the depth D to n, resulting in C_n. Though this procedure is clear, for the algorithm to function correctly, it is necessary that `extend` also train the added nodes, on the history x, and then have the mixtures of non "new" nodes updated to account for the transformation.

The `dropLayer(`C_D, n`)` sub-procedure takes an arbitrary CTW of depth D together with an integer n and returns the truncated tree C_{D-n}. This operation is simple enough, though in order for the resulting tree to be *the* CTW of depth $D-n$ on x, it is necessary that the following substitution, $P_w^s = P_{KT}$, be made for every *new* leaf node s at depth $D-n$ i.e., the mixture is assigned the value of the KT-estimator for s. Following this, for every non-leaf t the mixture P_w^t, must be recomputed to reflect the truncation.

C Additional Experiments

C.1 Distribution Shift

In this experiment, the underlying environment is a Bernoulli distribution with parameter 0.95 for the first 100 time steps, which then switches to a Bernoulli distribution with parameter 0.05 after time step 100. We use the depth-0 CTW (i.e., the KT estimator) and the power betting function for $\kappa \in (0.1, 3.0)$, evenly spaced. Multiple experimental runs are done using different random-number generator seeds. The typical behaviour of the two martingales in $\log_2$ scale are shown in Figures 4 and 5. As can be seen, the prequential martingale score drops till around time step 100 as the KT estimator learns the underlying Bernoulli distribution. The score starts spiking soon after time step 100 when the KT estimator's predictions start deviating from data drawn from the changed underlying Bernoulli distribution, and then eventually drops to near zero again as the KT estimator adapts to the data from the new underlying Bernoulli distribution. The conformal martingale behaves in a similar manner, but the score starts plateauting a lot faster just before time step 150. This is because the rank normalisation in the conformal p-value calculation allows the conformal test to more easily adapt to the new shifted distribution.

C.2 Distribution Shift: Robust Detection

In this experiment, we want to check whether the lower sensitivity of the conformal martingale can be used to manage false alarms in an environment where occasional noisy data could be confused with actual distribution shift. The underlying environment is a Bernoulli distribution with parameter 0.05 for the first 100 time steps, then we have a noisy burst of 1's for 20 time steps, after which the environment reverts to same Bernoulli with parameter 0.05 till time step 200, and then the environment switches to a Bernoulli with parameter 0.8 after time step 200. Multiple experimental runs are done using different random-number generator seeds. Figures 6 and 7 shows the typical behaviour of the prequential martin-

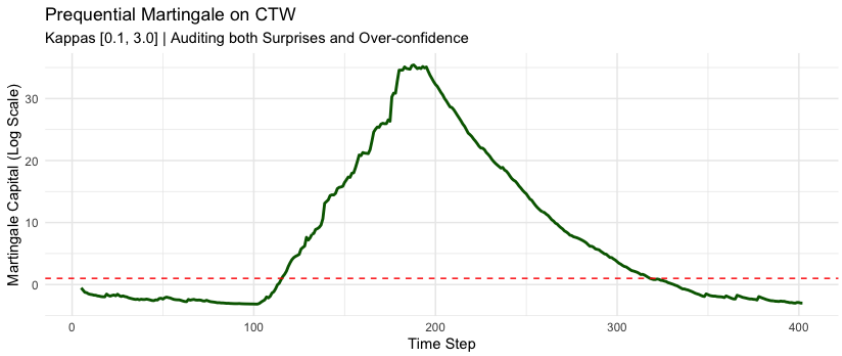

Figure 4: Prequential martingale scores for KT estimator under environment distribution shift at step 100

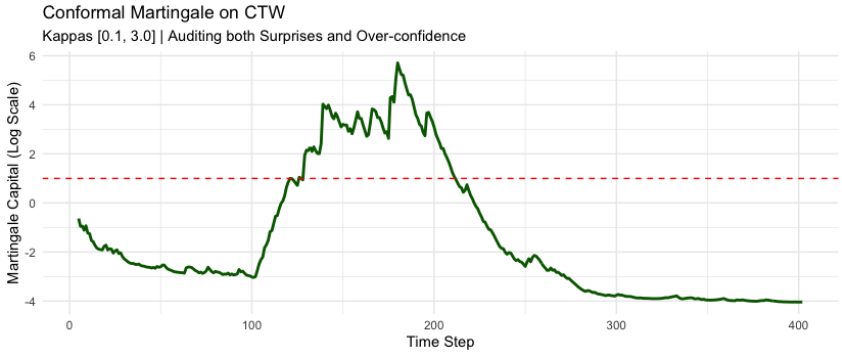

Figure 5: Conformal martingale scores for KT estimator under environment distribution shift at step 100

gale and conformal martingale with power gambler betting function on a depth-0 CTW. It looks plausible that a user-defined threshold like that represented by the red dotted lines in the diagrams can be tuned to minimise false alarms and reduce operator fatigue in actual applications.

C.3 Inadequate Context Depth

In this experiment, the underlying environment simply repeats the following length-12 sequence 110010111000 repeatedly. When we run a depth-12 CTW on the data sequence using a prequential martingale with a mixture of betting functions of the form (10), we get something like Fig. 8, which is as expected since the depth-12 CTW can learn the pattern well.

What happens when we use a CTW with lower depth, say, 3? We conducted multiple experimental runs using different random-number generator seeds. Fig. 9 shows the typical behaviour of a prequential martingale using the autocorrelation betting function (10) for $s \in (4, 11)$. As observed, the martingale scores did not pick up any surprises. However, adding the betting function for $s = 12$ to the mix yields Fig. 10, which detects that the depth-3 CTW is indeed misspecified because the u_t values are not independent since every pair (u_t, u_{t-12}) of values are correlated. Also, for a sanity check on Proposition 4, Fig. 11 shows that the inadequate context depth issue is not picked up by a prequential martingale with power gambler function, as expected.

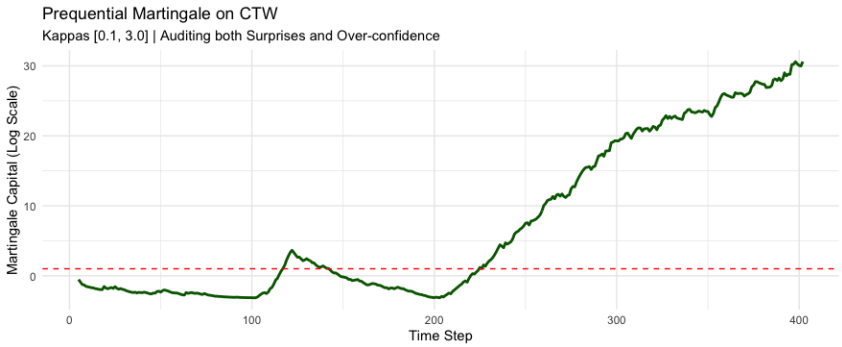

Figure 6: Prequential martingale scores for KT estimator under environment distribution shift at step 100

Figure 7: Conformal martingale scores for KT estimator under environment distribution shift at step 100

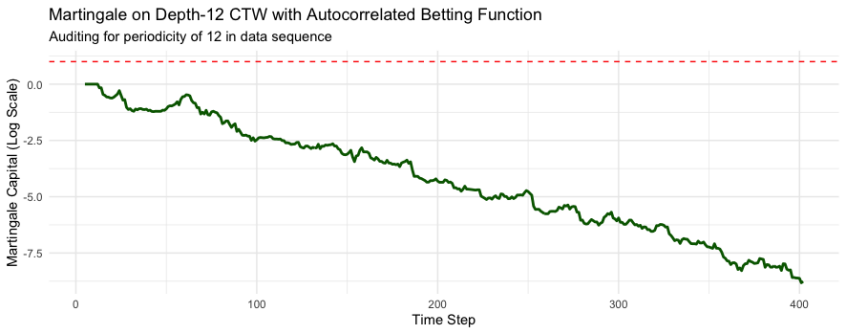

Figure 8: Scores for a prequential martingale using a range of betting functions on a depth-12 CTW learning on a data stream with periodicity 12

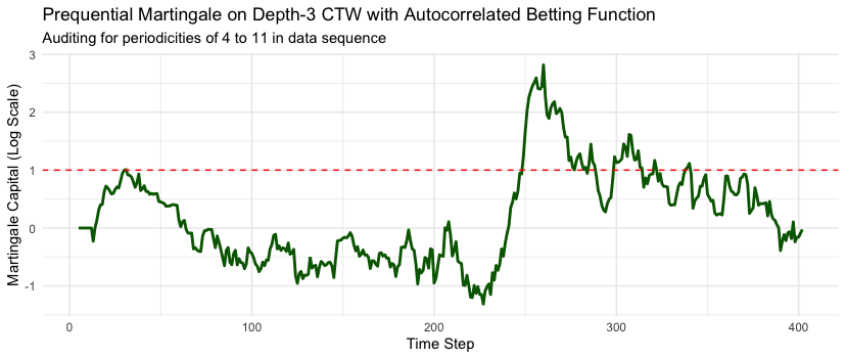

Figure 9: Scores for a prequential martingale looking for periodicity patterns of length 4 to 11 on a depth-3 CTW learning on a data stream with periodicity 12

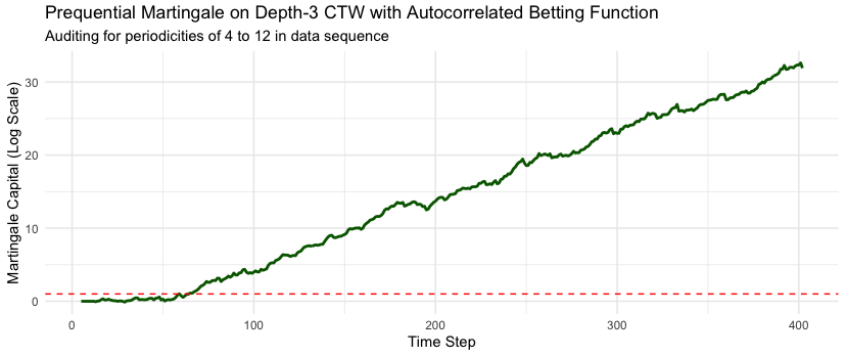

Figure 10: Scores for a prequential martingale looking for periodicity patterns of length 4 to 12 on a depth-3 CTW learning on a data stream with periodicity 12

Figure 11: Scores for a prequential martingale with power gambler betting function on a depth-3 CTW learning on a data stream with periodicity 12