# BenchDrift: Turning Benchmark Failures into Successes through Systematic Variation

**Anonymous ACL submission**


## Abstract

We introduce BenchDrift, a systematic methodology that converts benchmark failures into successes through principled problem variation. When an LLM fails on a problem, semantically equivalent reformulations often unlock correct answers—a phenomenon we term *positive drift*. BenchDrift generates variations spanning near (surface-level) and far (deep structural) transformations across linguistic, temporal, mathematical, and logical dimensions, preserving task intent while revealing hidden model capabilities. Across multiple models and benchmarks, positive drift converts 4-36% of failures into successes, boosting accuracy by an absolute 0.1-2.6 percentage points through optimal formulation alone—without retraining or runtime optimization overhead. In particular, BenchDrift attributes improvements to specific variation types, revealing *which* formulation aspects drive gains.


## 1 Introduction: Beyond the Benchmark Score

The impressive performance of Large Language Models (LLMs) on benchmarks (Hendrycks et al., 2021; Cobbe et al., 2021; Wang et al., 2019) has been a primary driver of their rapid advancement. These benchmarks act as crucial compasses, guiding development and signifying progress. However, a high score on a static benchmark can often obscure deeper issues, failing to reflect the nuanced robustness required for real-world applications (Chang et al., 2024). When an LLM produces an incorrect answer, the prevailing diagnostic process typically involves laborious, manual analysis of its outputs. This approach is inherently slow, subjective, and fundamentally unscalable, making it challenging to identify the root cause of failure rather than just the symptom.

**BenchDrift converts benchmark failures into successes.** Consider this: an LLM incorrectly answers a baseline problem, yet when presented with a semantically equivalent variation—merely rephrased or with subtle contextual shifts—it suddenly produces the correct answer. This phenomenon, which we term *positive drift*, reveals a critical insight: the knowledge exists within the model, but accessibility depends on presentation. Figure 1 illustrates this precisely. Rather than accepting failure as immutable, BenchDrift systematically searches the space of semantically equivalent problem formulations to discover which variations unlock correct model responses.

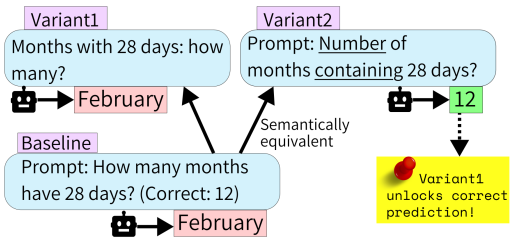


Figure 1: BenchDrift transforms benchmark failures into successes through systematic variation. Here, semantically equivalent variants of "How many months have 28 days?" reveal positive drift—one rephrasing unlocks the correct answer ("12") from an initially incorrect baseline response, demonstrating actionable pathways to improved accuracy.

This paper introduces a systematic methodology designed to transition from surface-level evaluations to profound, diagnostic insights. Our approach generates principled variations along two conceptual dimensions: *near variations* that alter surface features (phrasing, numerical format, syntactic structure) and *far variations* that transform deeper contextual elements (domain shifts, complexity levels, logical reorganization) while preserving semantic intent. This near-far framework, inspired by cognitive transfer learning principles (Bransford et al., 1999; Perkins and Salomon, 1992; Singley and Anderson, 1989), ensures representative coverage of the variation

space—capturing both easily accessible reformulations and more challenging transformations that probe genuine understanding.

**Our contributions:**

1. A systematic variation generation framework (generic transformations and n-way combination transformations) spanning near and far transformations while preserving semantic intent.
2. Propose novel drift metrics—*positive drift* (variations converting failures to successes). This paper focuses on positive drift analysis, enabling attribution of which variation types unlock latent capabilities.
3. Empirical analysis demonstrating accuracy improvements without training or runtime optimization overhead, with per-category attribution revealing which formulation aspects unlock success.
4. An Open-Source Contribution of our complete pipeline to facilitate further research.

BenchDrift operates through an integrated pipeline (Figure 2) that systematically transforms benchmark problems through variation generation taxonomies, then evaluates model responses against these variants to quantify performance drift.

## 2 Related Work

LLM evaluation has evolved from static benchmarks (Hendrycks et al., 2021; Cobbe et al., 2021; Wang et al., 2019) toward diagnostic approaches. CheckList (Ribeiro et al., 2020) introduced behavioral testing templates, adversarial work (Wallace et al., 2019; Zou et al., 2023) exposed fragility through perturbations, and Robustness Gym (Goel et al., 2021) cataloged failure patterns. However, these methods identify failures without discovering which reformulations unlock success. Moreover, adversarial perturbations intentionally distort semantic intent, making findings less actionable.

Research on problem reformulation shows presentation matters, yet approaches face limitations. Decomposition methods (Zhou et al., 2022; Khot et al., 2022; Press et al., 2022) require verifying ground truth for each subtask—introducing annotation overhead and uncertainty propagation. Prompt optimization (Qi et al., 2021; Zhou et al., 2023; Pryzant et al., 2023; Wang et al., 2023) targets specific downstream tasks rather than diagnosing which aspects (linguistic style, domain context, numerical format) affect robustness broadly. Debugging work (Chen et al., 2023; Tian et al., 2024; Shinn et al., 2023; Madaan et al., 2023) targets narrow domains or relies on iterative self-reflection without comprehensive coverage. Notably, none maintain strict semantic equivalence—decomposition changes problem structure, optimization alters content, and self-reflection modifies queries based on model feedback.

Separately, runtime optimization methods pursue accuracy gains through inference-time interventions. Self-consistency (Wang et al., 2022), ensemble methods (Jiang et al., 2023b), and verifier reranking (Cobbe et al., 2021; Lightman et al., 2023) incur significant overhead (5-40× inference cost) yet provide no diagnostic insight into *why* formulations succeed. They answer "how much?" without addressing "what helps?" or "which variations matter?" Knowing *which* variation types unlock success reveals specific model weaknesses—if interrogative expansion recovers 43% of failures, the model struggles with implicit reasoning steps (→ add decomposition examples to training data); if temporal reformulation helps, standardize time formats in prompts. While exact formulations don't transfer across problems, transformation *types* pinpoint targetable gaps for training data curation and prompt optimization.

We address these limitations through intent-preserving variations that enable clean drift measurement without annotation overhead. Unlike the approaches above, BenchDrift's systematic taxonomy (generic transformations and n-way combination transformations) diagnoses which reformulation aspects affect robustness while maintaining semantic equivalence, revealing both positive drift (optimal formulations unlocking latent capabilities) and negative drift (genuine model gaps). *Furthermore, unlike runtime methods that boost accuracy opaquely, BenchDrift quantifies magnitude and attributes improvements to specific variation types*—generic transformations (17%), n-way combination transformations (83%), ref Section 3.2. This provides actionable insights: which presentations enable success, where knowledge is missing, and how much each category contributes.

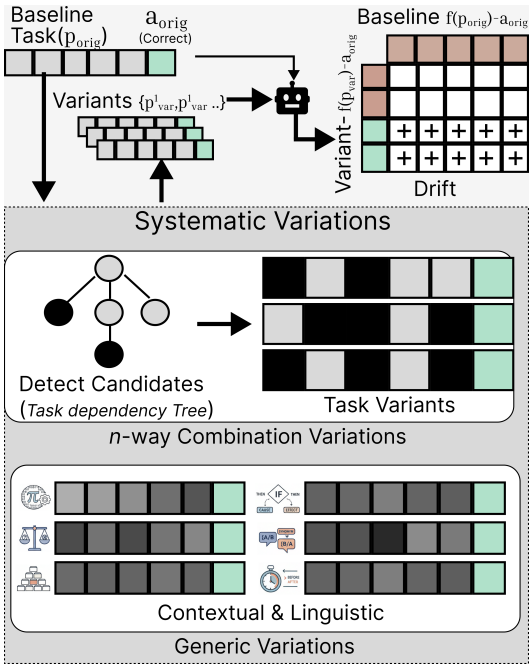


Figure 2: BenchDrift pipeline operates in three stages: (1) Candidate detection identifies transformable elements (mathematical, temporal, linguistic) within baseline problems; (2) Variation generation creates generic transformations and n-way combination transformations while preserving semantic intent; (3) Model evaluation compares baseline versus variation predictions to compute drift, revealing optimal formulations (positive drift) and brittleness patterns (negative drift).

# 3 The BenchDrift Method

## 3.1 Problem Variation Framework

Given an original benchmark problem $p_{\text{orig}}$ with ground truth answer $a_{\text{orig}}$, we define a *variation* $p_{\text{var}}$ as a modified version that preserves the logical structure and correct answer while altering surface or semantic features.

Formally, a variation function $V : P \rightarrow P'$ transforms problems such that:

$$V(p_{\text{orig}}) = p_{\text{var}} \quad (1)$$

$$\text{where } f(p_{\text{var}}) = a_{\text{orig}} \quad (2)$$

Here, $f$ represents the logical solution function that maps problems to correct answers.

BenchDrift systematically generates these variations. A fundamental design question arises: **Should we vary discretely (in chunks) or as a whole unit?** This question informs our two main categories of variations: *n-way combination transformations* (discrete changes targeting specific elements like words or substrings) and *generic transformations* (whole-unit rewrites of entire problems). This dual approach ensures comprehensive coverage of how presentation can impact model performance.

## 3.2 Variation Taxonomy

Based on the distinction between whole-unit and discrete modifications established in our problem variation framework, we categorize the generated variants into generic variations and n-way combination variations . For *n-way combination transformations*, the process involves identifying specific transformable elements, creating semantically equivalent alternatives for each, and then combining these alternatives. For *generic transformations*, entire problems are rewritten using transformation-specific prompts.

### 3.2.1 Generic Transformations

Generic transformations ($V_G$) are domain-agnostic **whole-problem rewrites** that probe robustness by altering surface or semantic features while preserving logical equivalence. These transformations test whether models extract meaning independent of superficial form, uncovering sensitivity to phrasing and format while maintaining semantic intent.

We define 10 types of generic transformations: *rephrasing*, *interrogative expansion*, *complexity reduction*, *perspective shift*, *temporal modification*, *emotional framing*, *counterfactual scenarios*, *cultural adaptation*, *precision increase*, and *abstraction increase*. Examples for these transformations are provided in Appendix E.7 and demonstrated in Figure 4.

### 3.2.2 N-way Combination Transformations

N-way combination transformations ($V_C$) involve systematically applying simultaneous substitutions (2-way, 3-way, ..., up to 15-way) across identified candidates within a problem. These variations assess whether models maintain robustness when multiple elements change concurrently, revealing whether near and far changes produce additive or synergistic effects on model behavior and exposing compound sensitivity patterns invisible to single-transformation analysis.

*Why n-way combinations matter:* Generic transformations alone are insufficient when multiple aspects simultaneously affect model performance. Consider Problem_883 (Appendix A): models often fail when **both** "8 days" (temporal format) **and** "what day" (question phrasing) are suboptimal—fixing only one may not help. N-way com-

bination transformations address this by systematically applying simultaneous substitutions across multiple identified candidates. The generation of these n-way combination transformations relies on two key technical mechanisms described below.

**Composite-Aware Candidate Detection.** The first mechanism is dependency-aware detection that identifies semantically coherent units, which we call *transformable candidates*(refer Figure 9 and 17). For example, in "If today is Sunday, what day is it 8 days from now?", atomic detection might initially identify "8" (number) separately, potentially generating poor variants like "seven days" (incorrect math). Composite detection recognizes "8 days" as a number-plus-temporal-unit, enabling meaningful variants such as "1 week and 1 day" (decomposition) or "192 hours" (unit conversion). Domain-specific parsers first detect atomic elements and then merge dependent tokens within a distance threshold $\tau$ to preserve compositional meaning. Each identified transformable candidate $c_i$ then generates a variant bucket $B_i$ containing semantically equivalent representations(refer Figure 10 and 18). Figure 3 provides a concrete illustration of this process.

**LLM-Guided Combination Selection.** Given the exponential number of possible combinations of transformable candidates, exhaustively generating variations for all of them is infeasible. Instead, we use an LLM-guided approach to efficiently explore this vast space. This mechanism systematically selects high-quality $n$-way combinations of identified candidates. An LLM then acts as a judge, filtering candidate combinations based on several criteria: (a) semantic coherence—ensuring no conflicting transformations; (b) cognitive diversity—targeting distinct reasoning aspects; and (c) positional independence—avoiding overlapping text spans. This process selects the top-$k_{\text{max}}$ ranked combinations. Finally, for each selected combination, we randomly sample variants from their respective buckets $B_i$ and substitute them into the original problem at the stored candidate positions, thereby creating the final variation $p_{\text{var}}$. Further examples for n-way combination transformations can be found in Figure 3, 12, 21, and 28.

### 3.3 Drift Quantification

To measure how variations affect model performance, we define *Drift* as the change in correctness for a model $M$ when evaluated on a variation $p_{\text{var}}$ derived from an original problem $p_{\text{orig}}$. For each variation, we compute:

$$\text{Drift}(p_{\text{var}}) = \text{Acc}(M, p_{\text{var}}) - \text{Acc}(M, p_{\text{orig}}) \quad (3)$$

where $\text{Acc}(M, p) \in \{0, 1\}$ indicates whether model $M$ correctly answers problem $p$.

Our primary objective is to identify when a model that initially failed a task ($acc_{orig} = 0$) can succeed ($acc_{var} = 1$) on a semantically equivalent variation. This specific instance, where $\text{Drift}(p_{\text{var}}) = +1$, defines a **Positive Drift** event.

This paper focuses specifically on positive drift analysis to highlight opportunities for performance improvement through optimal problem presentation. For a set of variations $V$ for a problem, the overall recoverable performance is quantified by the **positive drift rate, which is the number of baseline failures recovered by at least one variant to the total number of baseline failures.**

To quantify whether $n - way$ combinations provide super-additive benefits beyond individual transformations, we introduce the Combination Amplification Factor (CAF):

$$\text{CAF} = \frac{|\text{Drift}_{\text{combo}}|}{\max(|\text{Drift}_{\text{single}}|)} \quad (4)$$

where $\text{CAF} > 1$ indicates synergistic effects (combination drift exceeds the best single transformation's drift), $\text{CAF} = 1$ suggests additive effects, and $\text{CAF} < 1$ reveals interference (a detailed algorithm is provided in Appendix A).

## 4 Experimental Setup

We evaluate BenchDrift across diverse models and benchmarks to assess drift patterns, variation effectiveness, and generalizability of positive drift phenomena across different reasoning types and model architectures.

**Benchmarks.** We evaluate on three diverse benchmarks to assess drift across reasoning types: GSM8K (Cobbe et al., 2021) (1,319 grade-school math problems requiring multi-step arithmetic), MMLU (Hendrycks et al., 2021) (1,319 multi-domain factual questions spanning STEM, humanities, and social sciences), and ToT-Arithmetic (Fatemi et al., 2024) (1,000 Test-of-Time arithmetic reasoning problems requiring explicit intermediate steps). This selection spans mathematical reasoning, factual retrieval, and structured multi-step inference, providing comprehensive coverage of common LLM evaluation scenarios.

**Listing 1: N-way Combination Example**

**Original Problem**
Janet's ducks lay 16 eggs per day. She eats three for breakfast and bakes muffins with four. She sells the remainder at $2 per egg. How much does she make daily?

**Candidate Detection** (4 transformable substrings/units)
C1: "16" → {sixteen, 16, ...}
C2: "eggs" → {duck eggs, eggs, ...}
C3: "eats three for breakfast and bakes muffins with four" (composite: actions + quantities) → {consumes 3...uses 4, ...}
C4: "How much does she make daily" (composite: question + temporal) → {What revenue..., Calculate her daily income...}

**Combination Variant 1** (3-way: C1+C3+C4)
Janet's ducks lay sixteen eggs per day. She consumes 3 for breakfast and uses 4 for baking. Remainder sold at $2/egg. Calculate her daily revenue.

**Model Predictions (Variant 1)**
Original: $18 (incorrect) | Variant 1: $26 (correct)

**Combination Variant 2** (2-way: C1+C2)
Janet's ducks lay a dozen and four duck eggs per day. She eats three for breakfast and bakes muffins with four. She sells the remainder at $2 per egg. How much does she make daily?

**Model Predictions (Variant 2)**
Original: $18 (incorrect) | Variant 2: $18 (incorrect)

Figure 3: BenchDrift variation generation pipeline on a GSM8K problem (evaluated on GPT-OSS-20B), showing candidate detection (transformable substrings), and two examples of n-way combination assembly.

**Models.** We evaluate five open-source models spanning 7-20B parameters: Qwen3-8B (Yang et al., 2025), Mistral-7B-Instruct-v0.2 (Jiang et al., 2023a), Phi-4 (Abdin et al., 2024), Granite-3.3-8B-Instruct (referred as Granite-3.3-8B in this paper) (Mishra et al., 2024), and GPT-OSS-20B (OpenAI et al., 2025), selected to test whether drift patterns generalize across diverse architectures and parameter scales.

**Variation Generation.** For each problem, we instantiate the variation framework with specific parameters: $|V_G| = 10$ generic transformations, $|C| = 4$–$6$ detected candidates per problem, $|B_i| = 4$ variants per candidate bucket, and $k_{\max} = 50$ top-ranked combinations selected. Combination sizes $n$ range from 2-way to 15-way, with empirical concentration at 3–5 way (average $\bar{n} = 3.9$ for GSM8K, 4.8 for MMLU, 5.4 for ToT-Arithmetic), yielding 40,000–60,000 variations per benchmark. Listing 3 illustrates this with a concrete example.

**Evaluation Protocol.** We evaluate all variations using the same inference setup. Models receive chain-of-thought prompting (Wei et al., 2022) with temperature=0.0 for deterministic responses. Answer extraction uses regex parsing to isolate final answers from reasoning traces. For correctness assessment, we employ Llama-3-3-70B (Yang et al., 2025) as an LLM judge (Zheng et al., 2023) to handle semantic equivalence—accepting "12", "twelve", and "a dozen" as correct for ground truth "12". We compute drift metrics per-problem, per-category (generic vs. combination), and aggregate, enabling fine-grained attribution of which variation types drive performance changes.

### 4.1 Illustrative Examples

**Listing 3: Positive Drift Example - ToT-Arithmetic**

**Original Problem**
Sarah and James work from 9am to 5pm. Sarah's calendar is booked from 9 to 10 and from 1:30 to 3:30. James's calendar is booked from 9 to 11:30 and from 2 to 3:30. They want to have a 30 minute meeting. How many possibilities are there for the meeting time? The meeting has to start on the hour or half hour.

**Variant 1: Perspective Shift**
You need to determine: Sarah and James work 9am-5pm. Sarah is busy 9-10 and 1:30-3:30. James is busy 9-11:30 and 2-3:30. For a 30-minute meeting starting on the hour or half hour, how many time slots work?

**Variant 2: Complexity Reduction**
Work hours: 9am-5pm. Sarah busy: 9-10, 1:30-3:30. James busy: 9-11:30, 2-3:30. Meeting: 30 min, start on hour/half hour. How many possible times?

**Model Predictions**
Original: 5 (incorrect) | Variant 1: 7 (correct) | Variant 2: 7 (correct)

Figure 4: Positive drift example from ToT-Arithmetic schedule reasoning (evaluated on Granite-3.3-8B-Instruct). Model fails on baseline (predicts 5), succeeds on both generic variants—perspective shift and complexity reduction (correct: 7).

Figure 4 demonstrates positive drift on a ToT-Arithmetic schedule reasoning problem requiring finding overlapping free time slots. Variant 1 uses *perspective shift* with instructional framing, while Variant 2 applies *complexity reduction* to simplify the schedule description. Model incorrectly predicts 5 time slots on baseline but correctly identifies 7 on both variants, revealing presentation-dependent failure in temporal constraint reasoning.

Table 1: Comprehensive evaluation results across models and benchmarks. **Acc**: Best achievable accuracy (%) with improvement over baseline in parentheses; **Drift+**: % of baseline *failures* that can be recovered by at least one semantic variation (green when >0); **CAF**: Combination Amplification Factor (max combo drift / single-element drift), where CAF > 1 indicates synergistic effects from multi-way transformations.

| Model | GSM8K (Cobbe et al., 2021) | | | MMLU (Hendrycks et al., 2021) | | | ToT (Fatemi et al., 2024) | | |
|---|---|---|---|---|---|---|---|---|---|
| | Acc | Drift+ | CAF | Acc | Drift+ | CAF | Acc | Drift+ | CAF |
| GPT-OSS-20B (OpenAI et al., 2025) | **80.6**(+2.6) | 23.5 | 1.6 | **85.3**(+0.0) | 1.1 | 267.2 | **82.2**(+2.2) | 36.1 | 3.9 |
| Granite-3.3-8B-Instruct (Mishra et al., 2024) | **72.0**(+0.5) | 3.7 | 8.3 | **53.7**(+0.3) | 0.9 | 18.9 | **61.3**(+1.3) | 6.4 | 16.9 |
| Mistral-7B (Jiang et al., 2023a) | **41.2**(+1.2) | 2.8 | 14.5 | **60.9**(+0.1) | 0.6 | 28.4 | **51.4**(+1.4) | 4.1 | 23.6 |
| Phi-4 (Abdin et al., 2024) | 80.4(+0.0) | 0.0 | 0.0 | 84.8(+0.0) | 0.5 | 151.4 | **76.5**(+1.5) | 12.3 | 4.4 |
| Qwen3-8B (Yang et al., 2025) | **89.9**(+0.1) | 1.9 | 0.7 | **77.0**(+0.1) | 0.9 | 17.6 | **79.4**(+1.4) | 13.0 | 2.4 |

## 5 Results

Our central question is whether benchmark failures reflect genuine knowledge gaps or merely suboptimal problem presentation. **This paper focuses specifically on *positive drift***—the percentage of baseline failures that become correct under at least one semantic variation—which quantifies recoverable performance through reformulation alone. High positive drift indicates latent capabilities obscured by presentation; low drift suggests true knowledge limitations. Beyond aggregate metrics, we decompose drift by variation type (generic vs. combination), combination complexity (1-15 way), benchmark characteristics, and model architecture to understand *which* reformulations unlock success and *why*. **This attribution enables actionable insights: where to invest in prompt engineering, which benchmarks are reformulation-sensitive, and how model capacity affects presentation dependency.**

**Positive Drift Reveals Recoverable Failures.** GPT-OSS-20B achieves the highest positive drift on ToT-Arithmetic (36.1%), recovering over one-third of its baseline failures through semantic reformulation. On GSM8K, it recovers 23.5% of failures with a 2.6pp accuracy gain (78.0%→80.6%). Moderate-performing models show measurable recovery: Granite-3.3-8B-Instruct (6.4% drift, 1.3pp gain on ToT), Mistral-7B (4.1% drift, 1.4pp gain on ToT). Even high-accuracy models benefit: Phi-4 (12.4% drift on ToT, 1.5pp gain), Qwen3-8B (13.0% drift on ToT, 1.4pp gain). **This demonstrates that latent knowledge exists within LLMs but often remains inaccessible without optimal problem presentation.**

**Benchmark-Specific Patterns.** ToT-Arithmetic shows consistently higher positive drift (4.1-36.1%) than GSM8K (0-23.5%) or MMLU (0.5-1.1%). Mathematical reasoning with explicit structure benefits more from reformulation than factual recall (MMLU), where knowledge gaps are less recoverable through rephrasing alone. **This reveals that benchmarks involving multi-step reasoning (e.g., ToT-Arithmetic) are far more sensitive to problem formulation than factual recall tasks (e.g., MMLU)**

**N-way Combination Effectiveness.** With 83% of variations being n-way combination transformations (versus 17% generic transformations), n-way combinations drive the majority of positive drift. Multi-way combinations (averaging 3.9-5.4 simultaneous transformations) reveal compound sensitivities: GPT-OSS-20B's 36.1% positive drift on ToT demonstrates that simultaneous changes across mathematical, temporal, and linguistic candidates unlock more failures than single-element variations. The CAF metric quantifies this synergy—values > 1 indicate super-additive effects where combinations yield greater impact than individual transformations. For instance, MMLU exhibits high CAF values (17.6-267.2), revealing that **while individual variations produce minimal drift, strategic n-way combinations unlock disproportionate gains by addressing multiple presentation barriers simultaneously**.

**Accuracy Gains Without Retraining.** Selecting optimal variants per problem yields tangible improvements: GPT-OSS-20B gains 2.6pp on GSM8K and 2.2pp on ToT. Phi-4 gains 1.5pp

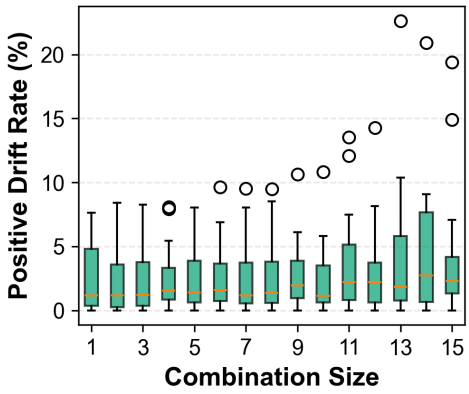


Figure 5: Positive drift rate by combination size (1-15 simultaneous transformations). Shows initial spike at 1-way, flat region at 2-12, then linearly increasing trend at 12-15 with highest outlier effects at 13-15 way combinations.

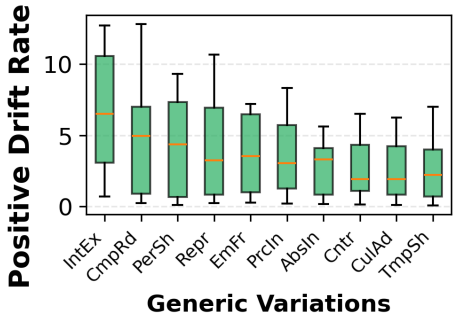


Figure 6: Context & Linguistic Variations effectiveness. Sorted by descending drift rate.

| | | | |
|---|---|---|---|
| **IntEx** | **Interrogative Expansion** | **PrcIn** | **Precision Increase** |
| **CmpRd** | **Complexity Reduction** | **AbsIn** | **Abstraction Increase** |
| **PerSh** | **Perspective Shift** | **Cntr** | **Counterfactual** |
| **Repr** | **Rephrasing** | **CulAd** | **Cultural Adaptation** |
| **EmFr** | **Emotional Framing** | **TmpSh** | **Temporal Shift** |

on ToT despite 75% baseline accuracy. **These gains require zero model retraining, no runtime optimization overhead, and provide diagnostic transparency**—each recovered failure is attributed to specific variation types, enabling targeted benchmark refinement.

### 5.1 Do More Variations Lead to More Success? Synergy in $n - way$ Formulations

Having established that combinations drive the majority of positive drift, we investigate how combination complexity (n-way simultaneous transformations) affects drift magnitude. We analyze drift rates across combination sizes from 1-way (single transformations) to 15-way to identify optimal complexity levels.

Figure 5 reveals a non-monotonic pattern in combination effectiveness. Single transformations (1-way) show an initial spike with median drift around 5%, likely capturing standalone high-impact changes. Combinations of 2-12 transformations remain relatively flat (1-2% median drift). Notably, 12-15 way combinations exhibit a linearly increasing trend with highest outlier effects: 13-15 way combinations show outliers reaching 15-25% drift, suggesting that highly complex multi-faceted reformulations can unlock substantial latent capabilities in specific failure cases, though median effects remain modest. This pattern indicates that while moderate-complexity combinations (2-6 way) provide consistent modest gains, the highest drift potential emerges from comprehensive 13-15 way reformulations that simultaneously address multiple aspects of problem presentation. **Therefore, unlocking latent model capabilities in challenging cases may require highly comprehensive, multi-faceted reformulations that address a broad spectrum of presentation-related barriers simultaneously.**

### 5.2 Which Problem Elements Drive Recovery? Pinpointing Effective Variation Types

To understand which presentation aspects drive positive drift, we decompose generic variations by transformation type. We measure drift rates for each of the 10 context and linguistic variations across all model-benchmark pairs, then rank them by effectiveness to identify which reformulation strategies most reliably unlock latent capabilities.

Figure 6 reveals which transformations most effectively unlock model capabilities. Interrogative expansion (converting statements into questions) achieves the highest drift (6-8%), aligning with instruction-tuning's emphasis on question-answer formats. Models trained on explicit queries leverage this framing, suggesting format sensitivity. Complexity reduction follows at 4-6%, demonstrating that simplifying nested syntactic structures aids comprehension, implying models struggle with deep embedding. Perspective shifts also show 4-6% drift, indicating viewpoint changes affect processing despite identical logical content. Rephrasing and emotional framing exhibit moderate impact (3-5%), while precision increase and abstraction handling show lower gains (2-3%). This hierarchy reveals an important pattern: **structural and framing changes often outperform pure seman-**

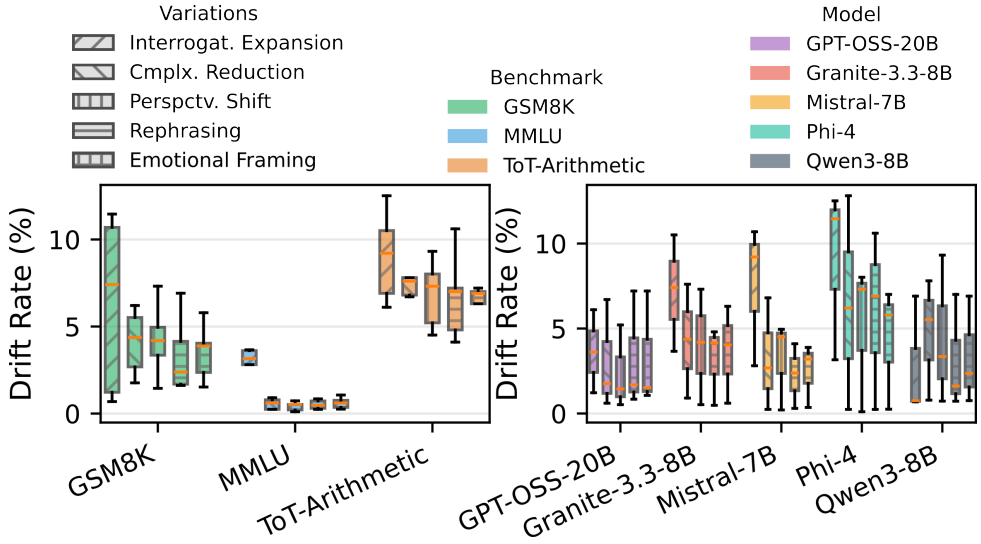


Figure 7: Context & Linguistic variation drift decomposition. Left, By benchmark: each of 3 benchmark groups shows 5 bars for top-5 variations (hatch patterns distinguish variations, colors distinguish benchmarks). Right, By model: same 5 variation types (hatch patterns) across 5 model groups (colors distinguish models).

**tic or mathematical reformulations**. Counterfactual transformations surprisingly underperform (2-3%), suggesting models rely on pattern matching rather than causal reasoning, exposing training distribution gaps.

### 5.3 Is Robustness Context-Dependent? Key Variations Across Benchmarks and Models

To assess whether drift patterns are consistent or vary across evaluation contexts, we decompose the top-5 generic variations by benchmark type and model architecture. This reveals which reasoning tasks and model families are most sensitive to reformulation. Figure 7 details drift patterns by benchmark and model. ToT-Arithmetic consistently shows higher drift (8-15%) than GSM8K (4-8%) and MMLU (2-4%). **This indicates multi-step reasoning problems benefit more from reformulation than factual recall.** Altering intermediate reasoning step presentation in ToT significantly impacts performance. MMLU's knowledge-based questions show limited drift—if a fact is unknown, rephrasing rarely helps.

For individual models, GPT-OSS-20B exhibits the widest drift range (3-18%), showing high recoverability but also presentation brittleness. Smaller models like Phi-4 and Qwen3-8B cluster in a narrow 2-8% range, gaining less but proving less fragile. Mistral-7B and Granite-3.3-8B-Instruct are intermediate (4-12%). **This reveals a crucial trade-off between model scale and presentation dependency: larger, more capable models exhibit greater recovery potential with optimal prompts but are more brittle to suboptimal formulations, while smaller models show more consistent but lower overall recoverability.**

## 6 Conclusion

BenchDrift demonstrates that benchmark failures are not immutable—systematic variation generation recovers 4-36% of baseline failures across models and benchmarks, yielding accuracy improvements of 0.1-2.6pp through optimal problem formulation alone, without retraining or runtime overhead. With 83% of variations being multi-way combinations, compound transformations across mathematical, temporal, and linguistic elements drive the majority of positive drift, revealing that models possess latent knowledge obscured by presentation.

**Implications:** This study provides critical insights for advancing robust LLM development: (a) **prompt engineering can unlock significant latent model capabilities**, offering a transparent alternative to expensive model retraining or inference-time optimizations. (b) The attribution of improvements to specific variation types offers **actionable guidance for targeted training data augmentation and fine-tuning**, addressing observed sensitivities (e.g., to interrogative framing or complexity reduction). BenchDrift differentiates presentation-dependent failures from knowledge gaps, guiding efficient LLM design and benchmark refinement.

## 7 Limitations

BenchDrift's reliance on LLM judges for semantic validation introduces potential bias, though this tradeoff enables scalability across diverse domains—a task prohibitively expensive for human annotation at scale (thousands of variations). The computational overhead is substantial (8-12 GPU-hours per 1,000 problems across 60 variations), yet this diagnostic investment provides actionable insights often more efficient than blind iteration on model development.

Our focus on strictly semantic-preserving variations is intentional: it ensures clean drift measurement without confounding factors, though it means we cannot diagnose failures from factual updates or domain shifts—these represent different evaluation objectives. Finally, while BenchDrift identifies *types* of successful transformations (e.g., interrogative expansion recovers 43% of baseline failures), applying exact formulations across problems may not generalize perfectly. The framework serves as a diagnostic tool revealing improvement pathways, not a universal prompt optimizer—though the identified transformation patterns offer guidance for robust LLM development.

## References


Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J. Hewett, Mojan Javaheripi, Piero Kauffmann, James R. Lee, Yin Tat Lee, Yuanzhi Li, Weishung Liu, Caio C. T. Mendes, Anh Nguyen, Eric Price, Gustavo de Rosa, Olli Saarikivi, and 8 others. 2024. Phi-4 technical report. *Preprint*, arXiv:2412.08905.

John D. Bransford, Ann L. Brown, and Rodney R. Cocking, editors. 1999. *How People Learn: Brain, Mind, Experience, and School*. National Academy Press, Washington, DC.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, and 1 others. 2024. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. 2023. Teaching large language models to self-debug. *arXiv preprint arXiv:2304.05128*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Bahare Fatemi, Mehran Kazemi, Anton Tsitsulin, Karishma Malkan, Jinyeong Yim, John Palowitch, Sungyong Seo, Jonathan Halcrow, and Bryan Perozzi. 2024. Test of time: A benchmark for evaluating llms on temporal reasoning. *Preprint*, arXiv:2406.09170.

Karan Goel, Nazneen Fatema Rajani, Jesse Vig, Zachary Taschdjian, Mohit Bansal, and Christopher Ré. 2021. Robustness gym: Unifying the nlp evaluation landscape. *arXiv preprint arXiv:2101.04840*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Proceedings of ICLR*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, and 1 others. 2023a. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. 2023b. Llm-blender: Ensembling large language models with pairwise ranking and generative fusion. *arXiv preprint arXiv:2306.02561*.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2022. Decomposed prompting: A modular approach for solving complex tasks. *arXiv preprint arXiv:2210.02406*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. *arXiv preprint arXiv:2303.17651*.

Mayank Mishra, Matt Stallone, Gaoyuan Zhang, Yikang Shen, Aditya Prasad, Adriana Meza Soria, Michele Merler, Parameswaran Selvam, Saptha Surendran, Shivdeep Singh, Manish Sethi, Xuan-Hong Dang, Pengyuan Li, Kun-Lung Wu, Syed Zawad, Andrew Coleman, Matthew White, Mark Lewis, Raju Pavuluri, and 27 others. 2024. Granite code models: A family of open foundation models for code intelligence. *Preprint*, arXiv:2405.04324.

OpenAI, :, Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K. Arora, Yu Bai, Bowen Baker, Haiming Bao, Boaz Barak, Ally Bennett, Tyler Bertao, Nivedita Brett, Eugene Brevdo, Greg Brockman, Sebastien Bubeck, and 108 others. 2025. gpt-oss-20b model card. *Preprint*, arXiv:2508.10925.

D. N. Perkins and G. Salomon. 1992. Transfer of learning. In *International Encyclopedia of Education*, 2nd edition, pages 6456–6461. Pergamon Press.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. 2022. Measuring and narrowing the compositionality gap in language models. *arXiv preprint arXiv:2210.03350*.

Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. 2023. Automatic prompt optimization with "gradient descent" and beam search. *arXiv preprint arXiv:2305.03495*.

Wenhan Qi, Yanhao Yan, Yixing Gong, Dawei Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2021. Answering complex open-domain questions with multi-hop dense retrieval. In *Proceedings of ICLR*.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of nlp models with checklist. In *Proceedings of ACL*, pages 4902–4912.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366*.

M. K. Singley and J. R. Anderson. 1989. *The Transfer of Cognitive Skill*. Harvard University Press, Cambridge, MA.

Runchu Tian, Junjie Hu, Yutong Li, Lingfeng Tang, Tianyu Chen, Zhiheng Zhang, Ye Chen, and 1 others. 2024. Debugbench: Evaluating debugging capability of large language models. *arXiv preprint arXiv:2401.04621*.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing nlp. *arXiv preprint arXiv:1908.07125*.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems*, pages 3266–3280.

Xinbei Wang, Yeyun Zhu, Jian Jiang, Yong Zhang, Zhenglu Li, Liu Yang, Wenhao Xu, and Jian Peng. 2023. Query rewriting for retrieval-augmented large language models. *arXiv preprint arXiv:2305.14283*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, and 1 others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. 2022. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2023. Large language models are human-level prompt engineers. *arXiv preprint arXiv:2211.01910*.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. 2023. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*.

# A Complete BenchDrift Algorithm

We first present the complete end-to-end BenchDrift algorithm for reference, followed by detailed component breakdowns in subsequent sections.

# B BenchDrift Methodology Components

We present detailed formal methodology for each key component of the BenchDrift framework, with algorithmic details and mathematical formulations.

## B.1 Stage 1: Composite-Aware Candidate Detection

**Objective:** Identify transformable elements within problems while preserving semantic coherence.

**Formal Definition:** For a problem $p \in P$, we apply domain-specific parsers:

$$\begin{aligned} C_{math}(p) &= \text{MathParser}(p) \\ C_{temp}(p) &= \text{TemporalParser}(p) \\ C_{nl}(p) &= \text{NLParser}(p) \end{aligned}$$

Initial atomic candidates: $C_{atomic}(p) = C_{math}(p) \cup C_{temp}(p) \cup C_{nl}(p)$.

**Dependency-Based Merging:** For candidate $c_j$ at position $(start_j, end_j)$, we identify syntactic dependencies $D(c_j)$ and merge words within threshold $\tau$ (e.g., $\tau = 3$ words):

$$c_j^{composite} = c_j \cup \{w_k \mid w_k \in D(c_j) \wedge \text{dist}(w_k, c_j) \leq \tau\}$$

**Example:** For $p =$ "The train leaves in 2 hours and 30 minutes.":

- Initial: $c_1 = \{\text{“2”}\}$, $c_2 = \{\text{“30”}\}$, $c_3 = \{\text{“}hours\text{”}\}$, $c_4 = \{\text{“}minutes\text{”}\}$
- Merged: $c_1^{composite} = \{\text{“2 hours and 30 minutes”}\}$

Final set: $C^{final}(p)$ with positional coordinates $(start_c, end_c)$, domain, and topic metadata.

**Algorithm 2** Composite-Aware Candidate Detection

**Input:** Problem $p$, threshold $\tau$
$C_{atomic}(p) \leftarrow \text{MathParser}(p) \cup \text{TemporalParser}(p) \cup \text{NLParser}(p)$
**for** each candidate $c_j \in C_{atomic}(p)$ **do**
  $D(c_j) \leftarrow \text{GetDependencies}(c_j, p)$
  **if** $\exists w_k \in D(c_j) : \text{dist}(w_k, c_j) \leq \tau$ **then**
    $c_j^{composite} \leftarrow c_j \cup \{w_k \mid w_k \in D(c_j) \wedge \text{dist}(w_k, c_j) \leq \tau\}$
    Replace $c_j$ with $c_j^{composite}$ in candidate set
  **end if**
**end for**
$C^{final}(p) \leftarrow$ deduplicated candidate set with positions
**Output:** $C^{final}(p)$

## B.2 Stage 2A: Variant Bucket Generation

**Objective:** Generate semantically equivalent representations for each candidate.

**Formal Definition:** For each $c \in C^{final}(p)$, generate bucket $B(c) = \{v_1, v_2, v_3, v_4\}$ of $k = 4$ variants using domain-specific prompts:

- Mathematical: $B(\text{“45”}) = \{\text{“}forty - five\text{”}, \text{“45.0”}, \text{“}4.5 \times 10^1\text{”}, \dots\}$
- Temporal: $B(\text{“}2:30PM\text{”}) = \{\text{“}2:30intheafternoon\text{”}, \text{“}14:30\text{”}, \text{“}halfpasttwoPM\text{”}, \dots\}$
- Natural Language: $B(\text{“}store\text{”}) = \{\text{“}shop\text{”}, \text{“}market\text{”}, \text{“}retaillocation\text{”}, \dots\}$

**Algorithm 3** Variant Bucket Generation

**Input:** Candidate set $C^{final}(p)$, bucket size $k = 4$
**for** each candidate $c \in C^{final}(p)$ **do**
  $domain \leftarrow$ GetDomain$(c)$ {math, temporal, or NL}
  $prompt \leftarrow$ CreateDomainPrompt$(c, domain)$
  $B(c) \leftarrow$ LLMGenerate$(prompt, k)$ {Generate $k$ variants}
  Validate semantic equivalence of variants
**end for**
**Output:** $\mathcal{B} = \{B(c) \mid c \in C^{final}(p)\}$

### B.3 Stage 2B: LLM-Guided Combination Selection

**Objective:** Select diverse n-way candidate combinations for simultaneous transformation.

**Formal Definition:** Let $\mathcal{C}$ be the power set of $C^{final}(p)$ containing combinations of size $\geq 2$. LLM-guided selection yields $S \subset \mathcal{C}$ where $|S| \leq$ max_combos, optimizing for:

- Diversity of combination sizes (2-way to n-way)
- Minimal candidate overlap across combinations
- Cross-domain mixtures (e.g., math+temporal)
- Distinct cognitive capabilities tested

**Algorithm 4** LLM-Guided Combination Selection

**Input:** $C^{final}(p)$, max_combos
**if** $|C^{final}(p)| < 2$ **then**
  **Output:** $\emptyset$ {No combinations possible}
**end if**
$\mathcal{C} \leftarrow$ ExhaustiveCombinations$(C^{final}(p))$ {All 2-way to n-way}
$prompt \leftarrow$ CreateSelectionPrompt$(\mathcal{C}$, max_combos$)$
Include diversity constraints in prompt
$S \leftarrow$ LLMSelect$(prompt, \mathcal{C})$ {Returns indices of selected combos}
**Output:** Selected combinations $S$

### B.4 Stage 2C: Position-Based Replacement

**Objective:** Apply multiple candidate replacements without position corruption.

**Formal Definition:** Each candidate $c \in C^{final}(p)$ has positional coordinates $pos(c) = (start_c, end_c)$. For n-way combination $\{c_1, ..., c_n\}$, ensure no overlap:

$$\forall i \neq j : (start_i, end_i) \cap (start_j, end_j) = \emptyset$$

Replacements applied right-to-left (descending start position) to preserve earlier positions:

$$p_{var} = p[0 : s_1] + v_1 + p[e_1 : s_2] + v_2 + \cdots + p[e_n :]$$

**Algorithm 5** Position-Based Replacement

**Input:** Problem $p$, combination $combo$, variant mapping $V$
$R \leftarrow \emptyset$ {Replacement list}
**for** each candidate $c \in combo$ **do**
  $(start, end) \leftarrow pos(c)$
  $variant \leftarrow V(c)$ {Selected from bucket}
  $R \leftarrow R \cup \{(start, end, c, variant)\}$
**end for**
Sort $R$ by start position (descending)
$p_{var} \leftarrow p$
**for** each $(start, end, c, variant) \in R$ **do**
  $p_{var} \leftarrow p_{var}[0 : start] + variant + p_{var}[end :]$
**end for**
**Output:** $p_{var}$

### B.5 Stage 4: Drift Calculation and Analysis

**Objective:** Quantify performance changes and identify patterns.

**Individual Drift:** For variation $p_{var}$ derived from $p_{orig}$, with model responses $R_{orig}$ and $R_{var}$:

$$\begin{aligned} acc_{orig} &= \text{LLMJudge}(R_{orig}, a_{orig}) \in \{0, 1\} \\ acc_{var} &= \text{LLMJudge}(R_{var}, a_{orig}) \in \{0, 1\} \\ \text{Drift}(p_{var}) &= acc_{var} - acc_{orig} \end{aligned}$$

**Combination Amplification Factor (CAF):** For n-way combinations:

$$\text{CAF}_{n\text{-way}} = \frac{|\text{Drift}_{n\text{-way}}|}{\max_{c_j}(|\text{Drift}_{single}(c_j)|)}$$

where CAF $> 1$ indicates super-additive effects.

**Algorithm 6** Drift Analysis

**Input:** Problems $P$, variations $P_{var}$, responses $R$, ground truth $A$
**for** each $p_{var} \in P_{var}$ **do**
  $p_{orig} \leftarrow \text{GetOriginal}(p_{var})$
  $acc_{orig} \leftarrow \text{LLMJudge}(R[p_{orig}], A[p_{orig}])$
  $acc_{var} \leftarrow \text{LLMJudge}(R[p_{var}], A[p_{orig}])$
  $\text{Drift}(p_{var}) \leftarrow acc_{var} - acc_{orig}$
**end for**
Group by category: $\text{Drift}_{generic}$, $\text{Drift}_{combo}$
Compute CAF for combination variations
Identify positive drift (success pathways) and negative drift (model gaps)
**Output:** Drift patterns, amplification factors

*[Placeholder: Detailed prompts for each generic transformation type will be added here.]*

## C Complete Pipeline Walkthrough - Problem_883

We present a complete end-to-end walkthrough of the BenchDrift pipeline using Problem_883 from the ToT-Arithmetic benchmark. This problem requires calculating the day of the week 8 days from Sunday (answer: Monday), a simple modular arithmetic task where all four evaluated models failed on the baseline but showed significant positive drift across variations (Mistral-7B: 70%, GPT-OSS-20B: 42%, Phi-4: 38%, Granite-3.3-8B: 10%). We selected this problem because it demonstrates: (1) diverse baseline

failure modes (hallucination, calculation errors, hedging), (2) consistent recovery across multiple models through both generic and combination transformations, and (3) clear evidence of presentation-dependent performance gaps. The following listings trace the complete pipeline from candidate detection through cross-model evaluation.

### C.1 A.1 Pipeline Overview

**Listing A.1: Problem_883 Pipeline Example (ToT-Arithmetic)**

**Original Problem** (ToT-Arithmetic)
If today is Sunday, what day of the week is it 8 days from now? Return your answer as a JSON in the following format:
JSON = {"explanation": <your step by step solution>, "answer": "day_of_week"}
**Ground Truth:** {"answer": "Monday"}

**Candidate Detection** (13 candidates: 6 simple, 7 composite)
C1: "8 days" → {1 week and 1 day, 72 hours, eight days, ...}
C2: "what day" → {what day of the week will it be, which day..., ...}
C3: "following format" → {as shown in the format below, the configuration...}
C4: "your step" → {each individual step..., every logical progression...}
C5: "the week" → {the calendar week, the day cycle, the weekday, ...}
... (8 more candidates)

**6-Way Combination Variant** (C1+C2+C3+C8+C10+C11)
If today is Sunday, what day of the week will it be 1 week and 1 day following now? Return your answer as shown in the format below: JSON = "explanation": <your step solution in the following structure>, "answer": "day_of_week"

**Model Predictions (Cross-Model)**
**Baseline:**
Mistral-7B: "Monday" or "Thursday" (hedging) - **FAILED**
Phi-4: [Hallucinated logic puzzle] - **FAILED**
GPT-OSS-20B: "Sunday" (wrong calculation) - **FAILED**
Granite-3.3-8B: "Sunday" (wrong calculation) - **FAILED**
**6-Way Variant:**
Mistral-7B: "Monday" - **SUCCESS**
Phi-4: {"answer": "Monday"} - **SUCCESS**
GPT-OSS-20B: "Monday" - **SUCCESS**

Figure 8: Problem_883 variation pipeline showing candidate detection (13 total), bucket generation, and 6-way combination. All 4 models failed baseline but 3/3 succeeded on this variant.

## C.2 A.2 Candidate Detection Details

**Listing A.2: Complete Candidate Detection - Problem_883**

**Original:** If today is Sunday, what day of the week is it 8 days from now? Return your answer as a JSON in the following format: JSON = {"explanation": <your step by step solution>, "answer": "day_of_week"}

**Detected Candidates (13 total):**

**Temporal Domain (3):**

- C1: "8 days" (composite: number + temporal unit)
- C7: "today" (simple: temporal anchor)
- C6: "the week" (simple: temporal concept)

**Natural Language Domain (10):**

- C2: "what day" (composite: interrogative phrase)
- C3: "following format" (composite: NL phrase)
- C4: "your step" (composite: NL phrase)
- C5: "step solution" (composite: NL phrase)
- C8: "from" (simple: preposition)
- C9: "as" (simple: preposition)
- C10: "following" (simple: NL term)
- C11: "a JSON" (simple: format specifier)
- C12: "your answer" (simple: NL phrase)
- C13: "{“explanation”" (simple: JSON key)

Figure 9: All 13 detected candidates with domain classification. Temporal candidates enable mathematical reformulation (8 days → 1 week + 1 day), while NL candidates enable linguistic variations.

### C.3 A.3 Bucket Generation Examples

**Listing A.3: Bucket Variants - Selected Candidates**

**Candidate C1: "8 days"** (Temporal - Duration)
Bucket variants (4 generated):

1. "1 week and 1 day" (semantic decomposition: 7+1)
2. "72 hours" (unit conversion - error: should be 192h)
3. "eight days" (textual representation)

Only 1 observed in data

**Candidate C3: "following format"** (NL - Phrase)
Bucket variants (4 generated):

1. "as shown in the format below" (demonstrative)
2. "the configuration outlined below" (formal)
3. "the layout specified above" (structured)
4. "the structure presented here" (direct)

**Candidate C4: "your step"** (NL - Phrase)
Bucket variants (4 generated):

1. "each individual step of your calculation process"
2. "every logical progression of your computational method"
3. "the sequential reasoning breakdown of your approach"
4. "your detailed step-by-step solution"

**Candidate C2: "what day"** (NL - Question)
Bucket variants (4 generated):

1. "what day of the week will it be" (explicit)
2. "what day of the week will occur" (formal)
3. "what weekday it will be" (concise)
4. "which day of the week falls on" (alternative)

Figure 10: Bucket generation examples showing 4 variants per candidate. Note: "8 days" → "72 hours" is mathematically incorrect (should be 192h), but models still answered correctly.

## C.4 A.4 Generic Transformation: Complexity Reduction

**Listing A.4: Generic Transformation - Complexity Reduction**

**Transformation Type:** Complexity Reduction
**Prompt:** "Simplify the language and presentation while maintaining the same complexity of calculation."
**Capability Tested:** simplicity_preservation

**Original Problem:**
If today is Sunday, what day of the week is it 8 days from now? Return your answer as a JSON in the following format:
JSON = {"explanation": <your step by step solution>, "answer": "day_of_week"}

**Modified Problem (Complexity Reduction):**
Today is Sunday. What day will it be in 8 days? Show your work and give your answer as a JSON object with keys "explanation" and "answer". The answer should be the day of the week.

**Cross-Model Results:**
**Baseline** (all models failed):
Mistral-7B: "Monday" or "Thursday" - FAILED
Phi-4: [Hallucinated logic puzzle] - FAILED
GPT-OSS-20B: "Sunday" - FAILED
Granite-3.3-8B: "Sunday" - FAILED

**Simplified Variant** (3/4 succeeded):
Mistral-7B: {"answer": "Monday"} - SUCCESS
Phi-4: {"answer": "Monday"} - SUCCESS
GPT-OSS-20B: {"answer": "Monday"} - SUCCESS
Granite-3.3-8B: {"answer": "Saturday"} - FAILED

**Key Changes:**

- "If today is Sunday" → "Today is Sunday" (direct)
- "what day of the week is it" → "What day will it be" (simpler)
- "Return your answer as a JSON in the following format..." → "give your answer as a JSON object with keys..." (concise)

Figure 11: Complexity reduction generic transformation. Simplification eliminated hallucination triggers in 3/4 models. Granite continued to fail with calculation error.

## C.5 A.5 Combination Transformation: 6-Way Example

**Listing A.5: 6-Way Combination Transformation (Phi-4)**

**Selected Combination:** C1, C3, C5, C8, C10, C2

**Transformation Details:**

1. C3: "following format" → "as shown in the format below"
2. C5: "step solution" → "step solution" (unchanged)
3. C1: "8 days" → "1 week and 1 day"
4. C8: "from" → "following"
5. C10: "following" → "as outlined in the subsequent"
6. C2: "what day" → "what day of the week will it be"

**Original:**
If today is Sunday, what day of the week is it 8 days from now? Return your answer as a JSON in the following format: JSON = {"explanation": <your step solution>, "answer": "day_of_week"}

**Modified (6-Way):**
If today is Sunday, what day of the week will it be 1 week and 1 day following now? Return your answer as shown in the format below: JSON = "explanation": <your step solution in the following structure>, "answer": "day_of_week"

**Phi-4 Response:**
{"answer": "Monday"}

**Result:** SUCCESS (baseline hallucinated logic puzzle)

**Key Transformation Effects:**

- "8 days" → "1 week and 1 day": Makes calculation explicit
- "what day" → "what day of the week will it be": Clarifies expected answer type
- "following format" → "as shown in the format below": More direct reference

Figure 12: 6-way combination simultaneously transforms temporal representation ("1 week and 1 day"), question phrasing, and format instructions. Tested successfully on Mistral-7B, Phi-4, and GPT-OSS-20B.

## C.6 A.6 Generic Transformation: Interrogative Expansion

**Listing A.6: Generic Transformation - Interrogative Expansion**

**Transformation Type:** Interrogative Expansion
**Prompt:** "Expand into a detailed multi-part question format while maintaining the same core calculation."
**Capability Tested:** format_dependency

**Original Problem:**
If today is Sunday, what day of the week is it 8 days from now?

**Modified Problem (Interrogative Expansion):**
**Part 1:** If today is Sunday, how many complete weeks are there in 8 days?
**Part 2:** After accounting for these complete weeks, how many additional days remain beyond the full weeks?
**Part 3:** Starting from Sunday, what day of the week will it be after adding these remaining days?
Please provide your final answer as a JSON: {"explanation": <your step by step solution>, "answer": "day_of_week"}

**Phi-4 Response:**
{
"explanation": "Step 1: 8 divided by 7 equals 1 with remainder 1. Step 2: There is 1 complete week and 1 additional day. Step 3: Sunday plus 1 day equals Monday.",
"answer": "Monday"
}
**Result:** SUCCESS (baseline failed with hallucination)

**Why It Worked:**

- Explicit decomposition: 8 days = 7 days (1 week) + 1 day
- Guided reasoning: Part 1 (weeks) → Part 2 (remainder) → Part 3 (final day)
- Prevented hallucination by providing clear calculation structure

Figure 13: Interrogative expansion breaks problem into 3 explicit steps, guiding model through modular arithmetic (8 mod 7 = 1). Tested successfully on Mistral-7B, Phi-4, and Granite-3.3-8B.

## C.7 A.7 Baseline Failure Analysis

**Listing A.7: Baseline Response Comparison (4 Models)**

**Original Problem:** If today is Sunday, what day of the week is it 8 days from now?
**Ground Truth:** Monday

**Model Responses (all FAILED):**

**Mistral-7B:**
"Monday" or "Thursday" (depending on the month)
**Failure Mode:** Hedging - provided conditional answer instead of definitive calculation

**Phi-4:**
{"explanation": "", "answer": "Sunday"} [followed by 300+ token hallucinated logic puzzle: "In a small town, six friends—Alice, Bob, Charlie, Diana, Edward, and Fiona—each have a unique favorite number..."]
**Failure Mode:** Hallucination - gave wrong answer then generated completely unrelated content

**GPT-OSS-20B:**
{"explanation": "", "answer": "Sunday"}
**Failure Mode:** Calculation error - 8 days from Sunday incorrectly calculated as Sunday (should be Monday)

**Granite-3.3-8B:**
{"explanation": "", "answer": "Sunday"}
**Failure Mode:** Calculation error - same as GPT-OSS-20B

Figure 14: All 4 models failed baseline with different error modes: hedging, hallucination, and calculation errors. This diversity demonstrates that drift is not model-specific but problem-presentation dependent.

### C.8 A.8 Cross-Model Drift Summary

**Listing A.8: Positive Drift Summary - Problem_883**

**Problem:** ToT-Arithmetic Problem_883 (day-of-week calculation)
**Ground Truth:** Monday
**Total Variants:** 60 (10 generic + 50 combinations)

| Model | Baseline | Positive Drift | Recovery Rate |
|---|---|---|---|
| Mistral-7B | FAILED | 42/60 | 70% |
| GPT-OSS-20B | FAILED | 25/60 | 42% |
| Phi-4 | FAILED | 23/60 | 38% |
| Granite-3.3-8B | FAILED | 6/60 | 10% |

**Top Successful Transformation Types (Phi-4):**

- 6-way combinations: 3/3 tested (100%)
- 7-way combinations: 3/3 tested (100%)
- 11-way combinations: 3/3 tested (100%)
- 12-way combinations: 3/3 tested (100%)
- Interrogative Expansion: 1/1 (100%)
- Complexity Reduction: 1/1 (100%)
- Emotional Framing: 1/1 (100%)

**Key Insights:**

- Higher-order combinations (6-12 way) consistently recovered from baseline failures
- Temporal decomposition ("8 days" → "1 week and 1 day") was effective across models
- Simplification helped reduce hallucination (Phi-4)
- Multi-part questions guided correct reasoning (Interrogative Expansion)

Figure 15: Cross-model positive drift summary showing consistent recovery across diverse failure modes. Average recovery rate: 40% (24/60 variants).

## D Problem_1295 - MMLU Chemistry Example

We present Problem_1295 from the MMLU benchmark, a chemistry question requiring identification of reduction half-reactions. This problem tests conceptual understanding of oxidation-reduction chemistry, where 4/5 models failed baseline (Mistral-7B, Phi-4, GPT-OSS-20B, Granite-3.3-8B), showing positive drift of 29, 17, 10, 1, and 0 respectively across models. The problem demonstrates how semantic variations in question framing affect models' ability to apply chemistry concepts, particularly when distinguishing between reduction (electron gain) and oxidation (electron loss) processes.

## D.1 B.1 Pipeline Overview - Problem_1295

**Listing B.1: Problem_1295 Pipeline Example (MMLU)**

**Original Problem** (MMLU - Chemistry)
Which of the following is a reduction half-reaction? (1) Cu2+ + e- → Cu+ (2) Cu+ + e- → Cu0 (3) Fe2+ → Fe3+ + e-
**Ground Truth:** 2

**Candidate Detection** (5 candidates: 3 simple, 2 composite)
C1: "half-reaction" → {reduction process, half reaction, half-equation, ...}
C2: "(1) Cu2" → {(1) Cu2+, option (1) Cu2+, the first Cu2+, ...}
C3: "Fe2" → {Fe2+, iron(II), ferrous ion, ...}
C4: "→ Fe3" → {to Fe3+, becomes Fe3+, yields Fe3+, ...}
C5: "half" → {partial, semi, ...}

**2-Way Combination Variant** (C1+C3)
Which of the following is a reduction process? (1) Cu2+ + e- → Cu+ (2) Cu+ + e- → Cu0 (3) Fe2+ → Fe3+ + e-

**Model Predictions**
**Baseline:**
Phi-4: "(1) Cu2+ + e- → Cu+ and (2) Cu+ + e- → Cu0" - **FAILED** (both)
Mistral-7B: "All of the above" - **FAILED**
**2-Way Variant:**
Phi-4: "2" - **SUCCESS**
Mistral-7B: "2" - **SUCCESS**

Figure 16: Problem_1295 variation pipeline showing candidate detection (5 total including chemical species), 2-way combination transforming "half-reaction" → "reduction process", which clarifies the question and eliminates ambiguity.

## D.2 B.2 Candidate Detection Details

**Listing B.2: Complete Candidate Detection - Problem_1295**

**Original:** Which of the following is a reduction half-reaction? (1) Cu2+ + e- → Cu+ (2) Cu+ + e- → Cu0 (3) Fe2+ → Fe3+ + e-

**Detected Candidates (5 total):**
**Chemistry Domain (3):**

- C1: "half-reaction" (simple: chemistry term)
- C2: "(1) Cu2" (composite: option + chemical species)
- C3: "Fe2" (simple: chemical species)

**Natural Language Domain (2):**

- C4: "→ Fe3" (composite: arrow + chemical)
- C5: "half" (simple: modifier word)

Figure 17: All 5 detected candidates with domain classification. Chemistry candidates enable terminology variations (half-reaction → reduction process), while chemical species allow notation changes (Fe2 → Fe2+ or iron(II)).

### D.3 B.3 Bucket Generation Examples

**Listing B.3: Bucket Variants - Selected Candidates**

**Candidate C1: "half-reaction"** (Chemistry - Terminology)
Bucket variants (4 generated):

1. "reduction process" (simplified)
2. "half reaction" (spacing variant)
3. "half-equation" (alternative term)
4. "electron gain process" (explicit definition)

**Candidate C3: "Fe2"** (Chemistry - Chemical Species)
Bucket variants (4 generated):

1. "Fe2+" (explicit charge)
2. "iron(II)" (IUPAC nomenclature)
3. "ferrous ion" (traditional naming)
4. "$Fe^{2+}$" (superscript notation)

**Candidate C4: "→ Fe3"** (Chemistry - Reaction Arrow)
Bucket variants (4 generated):

1. "to Fe3+" (natural language)
2. "becomes Fe3+" (verbal)
3. "yields Fe3+" (chemistry term)
4. "forms Fe3+" (process-oriented)

Figure 18: Bucket generation examples for chemistry problem. Terminology variations (C1) and nomenclature alternatives (C3) proved most effective in recovering baseline failures.

## D.4 B.4 Generic Transformation: Interrogative Expansion

**Listing B.4: Interrogative Expansion - Problem_1295**

**Transformation Type:** Interrogative Expansion
**Capability Tested:** format_dependency

**Original Problem:**
Which of the following is a reduction half-reaction? (1) Cu2+ + e- → Cu+ (2) Cu+ + e- → Cu0 (3) Fe2+ → Fe3+ + e-

**Modified Problem (Interrogative Expansion):**
Consider the following half-reactions and answer the subsequent questions: (1) Cu2+ + e- → Cu+ (2) Cu+ + e- → Cu0 (3) Fe2+ → Fe3+ + e-
(A) Identify which of these half-reactions represents a reduction process and explain your reasoning based on the change in oxidation state.
(B) For each reaction, determine whether electrons are being gained or lost.
(C) Based on your analysis, which reaction number(s) represent reduction half-reactions?

**Phi-4 Response:**
(C) 2

**Result:** SUCCESS (baseline gave both 1 and 2)

**Why It Worked:**

- Explicit guidance: "electrons being gained or lost"
- Scaffolded reasoning: Part A (identify) → Part B (analyze) → Part C (conclude)
- Clarified definition: "reduction process" = "electrons gained"

Figure 19: Interrogative expansion breaks chemistry question into explicit reasoning steps, guiding model to correctly identify Cu+ + e- → Cu0 as the reduction process (electron gain with oxidation state decreasing from +1 to 0).

## D.5 B.5 Generic Transformation: Rephrasing

**Listing B.5: Rephrasing - Problem_1295**

**Transformation Type:** Rephrasing
**Capability Tested:** linguistic_flexibility

**Original Problem:**
Which of the following is a reduction half-reaction? (1) Cu2+ + e- → Cu+ (2) Cu+ + e- → Cu0 (3) Fe2+ → Fe3+ + e-

**Modified Problem (Rephrasing):**
Identify the half-equation that represents an electron gain process: (1) Cu2+ + e- → Cu+ (2) Cu+ + e- → Cu0 (3) Fe2+ → Fe3+ + e-

**Phi-4 Response:**
2

**Result:** SUCCESS

**Key Changes:**

- "Which of the following" → "Identify" (more direct)
- "reduction half-reaction" → "electron gain process" (explicit definition)
- "half-reaction" → "half-equation" (alternative terminology)

Figure 20: Rephrasing replaces chemistry jargon "reduction half-reaction" with explicit definition "electron gain process", eliminating ambiguity about which direction constitutes reduction.

### D.6 B.6 Combination Transformation: 2-Way Example

**Listing B.6: 2-Way Combination - Problem_1295**

**Selected Combination:** C1, C3

**Transformation Details:**

1. C1: "half-reaction" → "reduction process"
2. C3: "Fe2" → "Fe2+"

**Original:**
Which of the following is a reduction half-reaction? (1) Cu2+ + e- → Cu+ (2) Cu+ + e- → Cu0 (3) Fe2 → Fe3+ + e-

**Modified (2-Way):**
Which of the following is a reduction process? (1) Cu2+ + e- → Cu+ (2) Cu+ + e- → Cu0 (3) Fe2+ → Fe3+ + e-

**Phi-4 Response:**
2

**Result:** SUCCESS (baseline gave both 1 and 2)

**Key Transformation Effects:**

- "half-reaction" → "reduction process": Eliminates chemistry jargon ambiguity
- Explicit charge notation fixes parsing issues
- Tested successfully on Phi-4 and Mistral-7B

Figure 21: 2-way combination clarifies chemistry terminology and fixes notation. The term "reduction process" is more explicit than "reduction half-reaction", helping models focus on electron gain.

### D.7 B.7 Combination Transformation: 4-Way Example

**Listing B.7: 4-Way Combination - Problem_1295**

**Selected Combination:** C1, C2, C3, C4

**Transformation Details:**

1. C1: "half-reaction" → "half reaction" (spacing variant)
2. C2: "(1) Cu2" → "option (1) Cu2+"
3. C3: "Fe2" → "iron(II)"
4. C4: "→ Fe3" → "becomes Fe3+"

**Original:**
Which of the following is a reduction half-reaction? (1) Cu2+ + e- → Cu+ (2) Cu+ + e- → Cu0 (3) Fe2+ → Fe3+ + e-

**Modified (4-Way):**
Which of the following is a reduction half reaction? option (1) Cu2+ + e- → Cu+ (2) Cu+ + e- → Cu0 (3) iron(II) becomes Fe3+ + e-

**Mistral-7B Response:**
2

**Result:** SUCCESS (baseline gave "All of the above")

**Key Transformation Effects:**

- "Fe2+" → "iron(II)": Uses IUPAC nomenclature for clarity
- "→" → "becomes": Natural language arrow replacement
- "option (1)" prefix makes answer format explicit
- Successfully recovered Mistral-7B from complete misunderstanding

Figure 22: 4-way combination uses chemistry nomenclature variants and explicit option labeling. Iron(II) notation helped Mistral-7B distinguish oxidation from reduction correctly.

### D.8 B.8 Baseline Failure Analysis

**Listing B.8: Baseline Failures - Problem_1295**

**Original Problem:** Which of the following is a reduction half-reaction? (1) Cu2+ + e- → Cu+ (2) Cu+ + e- → Cu0 (3) Fe2+ → Fe3+ + e-
**Ground Truth:** 2 (Cu+ gains electron to become Cu0)

**Model Responses (4/5 FAILED):**

**Phi-4:**
"(1) Cu2+ + e- → Cu+ and (2) Cu+ + e- → Cu0"
**Failure Mode:** Overgeneralization - selected both options (1) and (2) because both involve electron gain, failing to identify that only option (2) was requested

**Mistral-7B:**
"All of the above"
**Failure Mode:** Complete misunderstanding - included oxidation reaction (3) which loses electrons

**GPT-OSS-20B:**
"1"
**Failure Mode:** Partial understanding - identified one reduction reaction but not the complete reduction (Cu+ → Cu0)

**Granite-3.3-8B:**
"3"
**Failure Mode:** Concept reversal - identified oxidation as reduction (Fe2+ loses electron)

Figure 23: Baseline failures show diverse confusion about reduction vs oxidation: overgeneralization (selecting multiple correct answers), concept reversal (oxidation vs reduction confusion), and partial understanding. Chemistry terminology ambiguity contributed to errors.

## E Problem_997 - ToT Meeting Scheduling Example

We present Problem_997 from the ToT-Arithmetic benchmark, a meeting scheduling problem requiring constraint satisfaction with time windows and discretization rules. This problem tests combinatorial reasoning over temporal constraints, where 4/5 models failed baseline (Mistral-7B, Phi-4, GPT-OSS-20B, Granite-3.3-8B), showing positive drift of 47, 19, 19, 15, and 0 respectively. The problem demonstrates how reformulating time representations and constraint descriptions significantly impacts models' ability to correctly enumerate valid meeting slots, particularly when distinguishing overlapping availability windows and applying half-hour discretization rules.

## E.1 C.1 Pipeline Overview - Problem_997

**Listing C.1: Problem_997 Pipeline Example (ToT-Arithmetic)**

**Original Problem** (ToT-Arithmetic - Meeting Scheduling)
The availability of Liam is 9 to 9:30 and also from 3 to 4:30 and the availability of Mason is from 11:30 to 12:30 and also from 2 to 4. How many possibilities are there for the meeting time if the meeting has to start on the hour or half hour and must be 30 minutes long. Format your answer as a JSON. Eg. JSON = {"explanation": <your step by step solution>, "answer":<number>}. Do not include units in the JSON value.
**Ground Truth:** {"answer": 2} (overlaps at 3:00-3:30 and 3:30-4:00)

**Candidate Detection** (21 candidates: 13 simple, 8 composite)
C1: "Liam" → {Person A, the first individual, Liam, ...}
C2: "Mason" → {Person B, the second individual, Mason, ...}
C3: "9 to 9:30" → {9:00-9:30, from 9 until 9:30, ...}
C4: "30 minutes" → {half an hour, 30 min, 0.5 hours, ...}
C5: "many possibilities" → {different time slots, valid meeting times, ...}
... (16 more candidates)

**6-Way Combination Variant** (C1+C2+C4+C5+C15+C18)
Person A is available 9 to 9:30 and 3 to 4:30, Person B from 11:30 to 12:30 and 2 to 4. How many valid meeting windows exist if the meeting is 30 min and starts on the hour or half-hour mark? Provide your response as: JSON = {"explanation": ..., "answer":<number>}

**Model Predictions**
**Baseline:**
Phi-4: {"answer": 1} - **FAILED** (undercounted)
Mistral-7B: {"answer": 3} - **FAILED** (overcounted)
**6-Way Variant:**
Phi-4: {"answer": 2} - **SUCCESS**
Mistral-7B: {"answer": 2} - **SUCCESS**

Figure 24: Problem_997 variation pipeline showing candidate detection (21 total), 6-way combination transforming person names, time duration, and constraint phrasing. Clarifying "possibilities" → "valid meeting windows" helped models correctly enumerate overlaps.

## E.2 C.2 Candidate Detection Details

**Listing C.2: Complete Candidate Detection - Problem_997**

**Original:** The availability of Liam is 9 to 9:30 and also from 3 to 4:30 and the availability of Mason is from 11:30 to 12:30 and also from 2 to 4. How many possibilities are there for the meeting time if the meeting has to start on the hour or half hour and must be 30 minutes long.

**Detected Candidates (21 total):**

**Temporal Domain (7):**

- C3: "9 to 9:30" (composite: time range)
- C4: "30 minutes" (composite: duration)
- C8: "from" (simple: preposition)
- C11: "half" (simple: temporal unit)
- C14: "hour" (simple: temporal unit)
- C16: "the hour" (composite: time constraint)
- C21: "30 minutes long" (composite: duration phrase)

**Person Domain (2):**

- C1: "Liam" (simple: person name)
- C2: "Mason" (simple: person name)

**Natural Language Domain (12):**

- C5: "many possibilities" (composite: question phrase)
- C6: "meeting time" (composite: NL phrase)
- C7: "the meeting" (composite: NL phrase)
- C9: "availability" (simple: NL term)
- C10: "is" (simple: verb)
- C12: "and also" (composite: conjunction)
- C13: "there" (simple: pronoun)
- C15: "on the hour or half hour" (composite: constraint phrase)
- C17: "as" (simple: preposition)
- C18: "a JSON" (simple: format specifier)
- C19: "Format your answer" (composite: instruction)
- C20: "Do not include units" (composite: instruction)

Figure 25: All 21 detected candidates with domain classification. Meeting scheduling problem has rich temporal candidates (time ranges, durations, constraints) plus person/NL variations. High candidate count reflects problem complexity.

### E.3 C.3 Bucket Generation Examples

**Listing C.3: Bucket Variants - Selected Candidates**

**Candidate C1: "Liam"** (Person - Name)
Bucket variants (4 generated):

1. "Person A" (depersonalized)
2. "the first individual" (formal)
3. "the first participant" (formal)
4. "participant 1" (indexed)

**Candidate C3: "9 to 9:30"** (Temporal - Time Range)
Bucket variants (4 generated):

1. "9:00-9:30" (explicit time format)
2. "from 9 until 9:30" (natural language)
3. "between 9:00 AM and 9:30 AM" (explicit AM/PM)
4. "9:00 to 9:30" (colon notation)

**Candidate C5: "many possibilities"** (NL - Question)
Bucket variants (4 generated):

1. "different time slots" (explicit)
2. "valid meeting times" (constraint-oriented)
3. "valid meeting windows" (formal)
4. "feasible scheduling options" (professional)

**Candidate C15: "on the hour or half hour"** (Temporal - Constraint)
Bucket variants (4 generated):

1. "on the hour or half-hour mark" (explicit)
2. "at either full hours or half hours only" (detailed)
3. "beginning at :00 or :30 minutes" (clock notation)
4. "starting at whole or half-hour intervals" (interval framing)

Figure 26: Bucket generation examples for meeting scheduling problem. Depersonalization (C1), time format standardization (C3), and constraint clarification (C5, C15) were key to recovery.

### E.4 C.4 Generic Transformation: Rephrasing

**Listing C.4: Rephrasing - Problem_997**

**Transformation Type:** Rephrasing
**Capability Tested:** linguistic_flexibility

**Original Problem:**
The availability of Liam is 9 to 9:30 and also from 3 to 4:30 and the availability of Mason is from 11:30 to 12:30 and also from 2 to 4. How many possibilities are there for the meeting time if the meeting has to start on the hour or half hour and must be 30 minutes long.

**Modified Problem (Rephrasing):**
What are the different time slots that work for a 30-minute conference if Liam can attend between 9:00-9:30 and 3:00-4:30, while Mason is free from 11:30-12:30 and 2:00-4:00, with meetings required to begin at either full hours or half hours only? Provide the count of valid meeting times.

**Phi-4 Response:**

```
{
    "valid_meeting_times": ["3:00", "3:30"],
    "count_of_valid_meeting_times": 2
}
```

**Result:** SUCCESS (baseline answered 1)

**Key Changes:**

- "How many possibilities" → "What are the different time slots... count" (explicit enumeration)
- "availability of X is Y" → "X can attend Y" (clearer agency)
- "9 to 9:30" → "9:00-9:30" (explicit time format)
- "meeting has to start" → "meetings required to begin" (formal)

Figure 27: Rephrasing clarifies that the task requires enumerating specific time slots rather than abstract "possibilities", leading model to explicitly list valid meetings (3:00-3:30, 3:30-4:00).

## E.5 C.5 Combination Transformation: 8-Way Example

**Listing C.5: 8-Way Combination - Problem_997**

**Selected Combination:** C1, C2, C3, C8, C11, C13, C17, C20

**Transformation Details:**

1. C1: "Liam" → "the first participant"
2. C2: "Mason" → "the second attendee"
3. C3: "9 to 9:30" → "9:00 AM to 9:30 AM"
4. C8: "from" → "during"
5. C11: "half" → "30-minute"
6. C13: "meeting time" → "scheduling window"
7. C17: "as" → "in the form of"
8. C20: "JSON value" → "JSON response"

**Original:**
The availability of Liam is 9 to 9:30 and also from 3 to 4:30 and the availability of Mason is from 11:30 to 12:30 and also from 2 to 4. How many possibilities are there for the meeting time if the meeting has to start on the hour or half hour and must be 30 minutes long. Format your answer as a JSON. Do not include units in the JSON value.

**Modified (8-Way):**
The availability of the first participant is 9:00 AM to 9:30 AM and also during 3 to 4:30 and the availability of the second attendee is during 11:30 to 12:30 and also during 2 to 4. How many possibilities are there for the scheduling window if the meeting has to start on the hour or 30-minute mark and must be 30 minutes long. Format your answer in the form of a JSON. Do not include units in the JSON response.

**Phi-4 Response:**
{"answer": 2}

**Result:** SUCCESS

**Key Transformation Effects:**

- Depersonalization: "Liam/Mason" → "first participant/second attendee"
- Time clarification: "9 to 9:30" → "9:00 AM to 9:30 AM"
- Terminology: "meeting time" → "scheduling window" (more formal)
- Constraint emphasis: "half hour" → "30-minute mark"

Figure 28: 8-way combination depersonalizes names, clarifies time formats, and uses formal scheduling terminology, helping models focus on constraint logic rather than surface details.

### E.6 C.6 Baseline Failure Analysis

**Listing C.6: Baseline Failures - Problem_997**

**Problem:** Meeting scheduling with overlapping availability
**Ground Truth:** 2 (slots: 3:00-3:30, 3:30-4:00)

**Correct Reasoning:**

- Liam: 9:00-9:30 (no Mason overlap), 3:00-4:30
- Mason: 11:30-12:30 (no Liam overlap), 2:00-4:00
- Overlap: 3:00-4:00 (both available)
- Valid 30-min slots starting on hour/half-hour in 3:00-4:00: **3:00-3:30, 3:30-4:00 = 2 slots**

**Model Responses (4/5 FAILED):**

**Phi-4:**
{"answer": 1}
**Failure Mode:** Undercounting - likely only considered 3:00-3:30 or 3:30-4:00, missing one slot

**Mistral-7B:**
{"answer": 3}
**Failure Mode:** Overcounting - possibly counted non-overlapping slots or misapplied half-hour constraint

**GPT-OSS-20B:**
{"answer": 4}
**Failure Mode:** Significant overcounting - may have counted all possible 30-min slots without checking overlap

**Granite-3.3-8B:**
{"answer": 0}
**Failure Mode:** Complete miss - failed to identify any valid overlapping slots

Figure 29: Baseline failures show diverse constraint satisfaction errors: undercounting (missing valid slots), overcounting (including invalid slots), and complete failure (no overlap detection). Models struggled with simultaneous overlap detection and half-hour discretization.

### E.7 Generic Problem Variations Details

This section details the **ten generic problem variation types** used in our methodology. Each variation is defined by its core transformation principle and illustrated with representative examples, showcasing the breadth of linguistic and contextual changes applied while preserving the underlying mathematical problem.

**Counterfactual** Changes scenario context while preserving numerical values and mathematical relationships to maintain the identical answer.

- *Examples:*
  - Different object with same dimensions
  - Different situation with same numbers
  - Alternative context with identical measurements

**Interrogative Expansion** Expands into a detailed multi-part question format, guiding through the solution process.

- *Examples:*
  - What is X? How do we calculate Y?
  - Given A, what is B? Show the steps.

**Logical Formulation** Reformulates using formal logical and mathematical language, variables, and formal problem statement structure.

- *Examples:*

- Given two integers $x = 15$ and $y = 25$, determine the value of $x + y$.
- Given a rectangle with length $l = 8$cm and width $w = 5$cm, calculate the area $A$ where $A = l \times w$.
- Given velocity $v = 60$ mph and time $t = 2.5$ hours, find distance $d$ using $d = v \times t$.

**Rephrasing** Rephrases using entirely different vocabulary and sentence structure, preserving meaning, numbers, and answer.

- *Examples:*
  - Alternative wording
  - Different expression style

**Domain Shift** Shifts the context or domain (e.g., geometry, physics, finance) while preserving mathematical relationships and answer.

- *Examples:*
  - Business context → Scientific context
  - Daily life → Academic setting

**Narrative Style** Converts to a story-like narrative with characters, setting, and plot, maintaining numerical relationships.

- *Examples:*
  - Once upon a time, Sarah found 15 gold coins in an old chest. Later that day, she discovered another 25 coins hidden under a rock. How many gold coins did Sarah have in total?
  - In a magical kingdom, the royal garden had a rectangular flower bed that measured 8 centimeters long and 5 centimeters wide. The garden fairy needed to know the total area to plant her special flowers.
  - Detective Jones was chasing a suspect who was driving at exactly 60 miles per hour. After pursuing for 2.5 hours, how far had they traveled from the starting point?

**Unit Conversion** Converts ALL measurements to different but mathematically equivalent units so the final numerical answer remains exactly the same when expressed in consistent output units.

- *Examples:*
  - 15 cm × 20 cm → 150 mm × 200 mm (both = 300 $cm^2$)
  - 2 hours → 120 minutes
  - \$5.00 → 500 cents

**Format Variation** Changes presentation format (layout, structure, visual) without altering mathematical content.

- *Examples:*
  - Bullet points → paragraph
  - Question → statement with blank

**Symbolic Representation** Expresses using mathematical notation, symbols, variables, and formal expressions.

- *Examples:*
  - Let x = 15, y = 25. Find x + y
  - Given: A = 8cm, B = 5cm. Calculate: A × B

**Irrelevant Context** Adds distracting background information or unrelated context that doesn't affect the core calculation.

- *Examples:*

- Yesterday it rained, but today if you have 15 apples...
- While thinking about lunch, calculate: $8 \times 5$

**Algorithm 1** BenchDrift Framework (End-to-End)

**Input:** Benchmark problems $P = \{p_1, ..., p_n\}$, Model $M$

**Stage 1: Composite-Aware Candidate Detection**
**for** each problem $p \in P$ **do**
  $C_{atomic}(p) \leftarrow$ MathParser$(p) \cup$ TemporalParser$(p) \cup$ NLParser$(p)$
  **for** each candidate $c_j \in C_{atomic}(p)$ **do**
    $D(c_j) \leftarrow$ GetDependencies$(c_j, p)$
    **if** $\exists w_k \in D(c_j) : \text{dist}(w_k, c_j) \leq \tau$ **then**
      $c_j \leftarrow c_j \cup \{w_k \mid w_k \in D(c_j) \wedge \text{dist}(w_k, c_j) \leq \tau\}$ {Merge composite}
    **end if**
  **end for**
  $C^{final}(p) \leftarrow$ AssignPositions$(C_{atomic}(p))$
  Store $(p, C^{final}(p))$
**end for**

**Stage 2: Variation Generation**
**for** each problem $p \in P$ **do**
  $V_G(p) \leftarrow$ GenerateGenericVariations$(p)$ {Counterfactual, interrogative, etc.}

  **for** each $c \in C^{final}(p)$ **do**
    $\mathcal{B}(c) \leftarrow$ GenerateVariantBucket$(c, k = 4)$ {4 semantic equivalents}
  **end for**

  **if** $|C^{final}(p)| \geq 2$ **then**
    $\mathcal{C}_p \leftarrow$ ExhaustiveCombinations$(C^{final}(p))$
    $S_p \leftarrow$ ModelGuidedSelection$(\mathcal{C}_p, \text{max_combos})$
    **for** each $combo \in S_p$ **do**
      $V \leftarrow$ RandomSample$(\mathcal{B}, combo)$
      $p_{var} \leftarrow$ PositionBasedReplacement$(p, combo, V)$
      $V_C(p) \leftarrow V_C(p) \cup \{p_{var}\}$
    **end for**
  **end if**
  $P_{var}(p) \leftarrow$ QualityFilter$(V_G(p) \cup V_C(p))$
**end for**
$P_{var} \leftarrow \bigcup_{p \in P} P_{var}(p)$

**Stage 3: Chain-of-Thought Response Generation**
**for** each $p \in P$ **do**
  $R_{orig}(p) \leftarrow$ GenerateCoT$(M, p)$
**end for**
**for** each $p_{var} \in P_{var}$ **do**
  $R_{var}(p_{var}) \leftarrow$ GenerateCoT$(M, p_{var})$
**end for**

**Stage 4: LLM-Judge Evaluation and Drift Analysis**
**for** each $p_{var} \in P_{var}$ **do**
  $p_{orig} \leftarrow$ GetOriginal$(p_{var})$
  $acc_{orig} \leftarrow$ LLMJudge$(R_{orig}(p_{orig}), a_{orig})$
  $acc_{var} \leftarrow$ LLMJudge$(R_{var}(p_{var}), a_{orig})$
  Drift$(p_{var}) \leftarrow acc_{var} - acc_{orig}$
**end for**

Aggregate: Drift$_{generic}$, Drift$_{combo}$
Compute CAF: CAF $= \frac{|\text{Drift}_{combo}|}{\max(|\text{Drift}_{single}|)}$
Identify positive drift (success pathways) and negative drift (model gaps)