

Controlling Creativity in Concept Space: A Framework for Generative Models

Dana Levin

Hebrew University of Jerusalem
dana.levin@mail.huji.ac.il

Dafna Shahaf

Hebrew University of Jerusalem
dshahaf@cs.huji.ac.il

Abstract

Generative models, such as text-to-image models, excel at taking textual prompts and producing semantically faithful outputs. However, the user’s ability to control the level of *creativity* of the output remains limited. We introduce Concept-Level Temperature (Concept- τ), a novel framework for controlling creativity in generative models. Unlike existing methods that operate on low-level features (tokens, noise, etc.), Concept- τ operates at a higher level of abstraction; it systematically identifies *decisions left underspecified* by the prompt and realizes them according to a creativity parameter, creating a *structured exploration space*.

In this work, we design general criteria for a creativity parameter, and implement an interpretable, model-agnostic method to satisfy them. We demonstrate our ideas on the text-to-image domain; empirical evaluation, both qualitative and quantitative, shows that Concept- τ significantly outperforms baseline methods in creative modulation and enhancement.

1 Introduction

Generative models have rapidly become tools for creative exploration, enabling users to produce images, text, and other outputs from simple prompts (Minaee et al., 2024; Ramesh et al., 2022; Podell et al., 2024; Nichol et al., 2022; Ding et al., 2022; Balaji et al., 2023; Rombach et al., 2022; Dhariwal and Nichol, 2021). Yet, a key limitation remains: while the user can specify *what* should appear in the output, they often cannot meaningfully specify how much *creativity* the model should apply.

This challenge arises because prompts are inherently underspecified. Each prompt leaves many semantic choices unresolved, creating a large space of possible outputs. However, current generative models provide little principled control over how this space is explored.

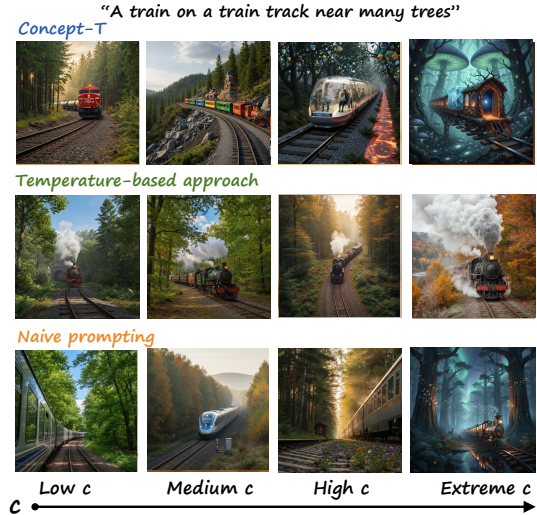


Figure 1: Images obtained via Concept- τ (top) vs. temperature-based and naive prompting approaches (Section 4), as the creativity parameter increases.

Naive prompting strategies, such as explicitly instructing the model to “be more creative” or including a numerical creativity score in the prompt, often fail to produce consistent or controllable variation, frequently collapsing to a small set of outputs (Han et al., 2025b; Mizrahi et al., 2026; Jentsch and Kersting, 2023).

Existing control mechanisms, such as temperature in language models (Zhang et al., 2024) and classifier-free guidance (CFG) in diffusion models (Ho and Salimans, 2021), modulate variability at the level of sampling (e.g., token distributions, noise scales or gradient magnitudes). Because these parameters operate on low-level representations, their effects on the semantic decisions that define creative variation are indirect and difficult to interpret, often leading to semantic drift, over-constraining or chaotic sampling regimes (Peepkorn et al., 2024; Shen et al., 2024; Biroli et al., 2024). Other lines of work attempt to directly optimize for creativity (Richardson et al., 2024;

Han et al., 2025b; Ismayilzada et al., 2025), but these methods are typically tied to specific architectures and provide limited interpretability, and many struggle with complex inputs. As a result, users still lack a principled way to control the level of creativity in outputs.

Creativity is commonly defined as a combination of **novelty** and **utility** (Boden, 1991; Mumford, 2003). This view provides a principled lens for our use case: creative control should meaningfully balance *originality* with *semantic coherence*.

In this work, we formalize general criteria for a **creativity parameter** for creative control of generative models, including monotonicity, diversity, utility and coverage. We introduce Concept-Level Temperature (Concept- τ), a novel framework for controlling creativity in generative models, and demonstrate it in the **text-to-image** (T2I) domain. See Figure 1 (top) for example outputs.

Unlike previous work, Concept- τ operates at a higher level of abstraction, **systematically identifying decisions left underspecified** by the prompt and **realizing them according to a controllable creativity parameter** (e.g., the color, type, or cargo of the train in the prompt “A train on a train track near many trees”, Figure 1). This produces a **structured exploration space** of possible outputs, from obvious to highly inventive. Importantly, our contribution is not in the idea of prompt rewriting per se, but in *how* these prompts are systematically constructed to allow modeling and controlling creativity through a parameter.

Concept-level controls align with human-like abstraction and compositional reasoning (Lake et al., 2017), enabling creativity to be modulated at the level where semantic decisions are actually made. We argue that this kind of parameter offers a more interpretable and effective lever for creative control than existing methods. Our contributions are:

- We introduce an interpretable, model-agnostic framework for controlling creativity in generative models via a **continuous parameter** operating over **concept-level semantic decisions**. We define key desiderata and an evaluation protocol.
- We demonstrate our ideas in the T2I domain. Extensive empirical evaluation combining quantitative, qualitative, and user studies show that our approach significantly outperforming baselines in creative modulation and enhancement.
- We show a proof-of-concept of adapting our method to another domain (recipes).

- We release all data and code¹.

2 Creativity Parameter: Desiderata

We start by describing our general setting. Let $M_G(d, \theta)$ be a generative model (for instance, a text-to-image model). The inputs to M_G include a textual prompt d and a vector of parameters θ . Let $f(c, d, \theta) = \langle d', \theta' \rangle$ be a (potentially non-deterministic) function that takes as input a *creativity parameter* c and modifies the inputs of M_G , controlling the creativity of the output $M_G(f(c, d, \theta))$.

The definition of f is deliberately flexible. In this work, we focus on **modifying the textual prompt** d , but other works might explore injecting creativity through different model parameters.

We now establish the criteria a creativity parameter should meet. In line with creativity literature (Boden, 1991; Mumford, 2003), creativity is commonly characterized as the combination of *novelty* (often operationalized as *surprise*) and *value* (or *utility*). Thus, we posit that the role of a creativity parameter is to increase novelty while maintaining utility of the outputs. Formally, **creativity parameter** c should satisfy the following properties:

P1. Monotonic Linear Novelty: The perceived novelty of the output should increase monotonically and approximately linearly with c . This is for ensuring a consistent, predictable scaling of novelty with respect to c .

In surprisal theory, novelty arises from accumulation of unexpected elements, each contributing novelty proportional to its self-information ($-\log P(x)$) (Shannon, 1948; Hale, 2001; Levy, 2008; Itti and Baldi, 2009). It is therefore natural to think of increasing c as progressively injecting additional surprising elements into the output. However, while in theory these contributions are additive, human perception of creativity exhibits diminishing returns (Cannan, 1892; Brue, 1993): successive surprises have progressively smaller impacts (see Figure 3; the impact of “glowing hair” is amplified when introduced into a simple scene, compared to a scene that already includes other surprising elements).

To preserve a roughly linear behavior, increments in c should therefore induce progressively more extreme surprise, compensating for saturation.

¹Code and data: https://github.com/LevinDana/concept_level_temp_code.

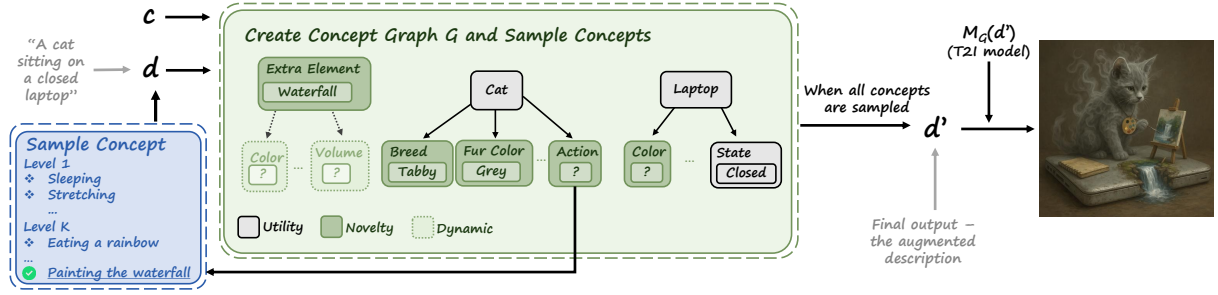


Figure 2: Concept-Level Temperature framework for image generation. We construct a concept graph G from the information missing in d , (§3.1) and assign each concept a creativity level (§3.2). For each concept, we generate candidate values across k creativity levels, conditioned on the iteratively updated d . We then select a sample based on the predefined creativity level (§3.3) and update d . After processing all concepts, the augmented description d' is passed to a T2I model to produce the final image.



Figure 3: Illustration of diminishing returns: for prompt ‘Girl in the forest’, adding *colorful glowing hair* has a much larger impact on A1 (plain scene) than on B1, which already has several surprising elements (the girl floats, trees are burning...)

- P2. **Wide Novelty Range:** Output created with different values of c should span a wide range of novelty, from the most predictable to highly surprising realizations of d .
- P3. **Diversity:** If f is non-deterministic, we expect outputs sampled with higher c ’s to exhibit more diversity than outputs sampled with lower c ’s, as creativity can take many forms.
- P4. **Utility Preservation:** Applying f does not degrade the utility of the outputs across the entire range of c . This ensures that the enriched prompt d' will remain faithful to the original prompt d (i.e., entailment: $d' \rightarrow d$). The definition of utility might also include domain-specific aspects (e.g., in recipe generation, ingredients are edible; see Section 5).

3 Method

In this work we focus on injecting creativity through the prompt d' ; we believe this is a natural choice for many generative models. We demonstrate our ideas in the domain of image generation. Given input prompt d (e.g., “A cat sitting on a closed laptop”) and creativity parameter $c \in [0, 1]$, our goal is to generate an enriched prompt d' in a way that adheres to the specifications of §2.

Our framework is based on the observation that the original prompt contains underspecified elements, delineating the space within which choices (more or less creative) can be made.

We define a structure we call a **Concept Graph** G . This graph captures concepts implied by the input prompt (cat, laptop), as well as concepts left underspecified (e.g., the cat’s breed or pose). This representation allows the model to modulate creativity at a high level of abstraction.

Our framework includes three key steps (Figure 2). First, we construct the concept graph G from d (§3.1). Next, we assign a creativity level to each underspecified concept (§3.2). Finally, we traverse G and sample values for concepts according to their assigned creativity levels (§3.3).

3.1 Creating the Concept Graph

We construct the concept graph G by identifying unspecified concepts in d such as the cat’s breed. These concepts are organized into a structured graph representation $G = (V, E)$, where:

- Nodes V represent concepts. V includes both *utility* concepts meant to satisfy property P4, preserving faithfulness to d (e.g., “cat”, “laptop”), and *novelty* concepts, which are not explicitly determined by d and thus left for choosing (“breed”). Utility concepts whose value is mentioned in d (“closed laptop”) have this value captured in G (“state=closed”), otherwise the value is left open.

Importantly, concepts are dynamic and can expand to additional concepts. For instance, in Figure 2, when a waterfall is sampled, it spawns concepts such as “water color”, “volume”. Concepts can also be removed when prior assignments render them

Algorithm 1: Surprise Allocation Function
 $Assign(n, k, c)$

Input: n : number of concepts
 k : number of discrete surprise levels
 $c \in [0, 1]$: creativity parameter
Output: Number of concepts corresponding to each surprise level

Function $Assign(n, k, c)$:

```

if  $k = 1$  then
    Assign the value  $n$  to  $l_1$ ;
    return allocation vector
     $(l_k, l_{k-1}, \dots, l_1)$ ;
// num. concepts at level  $k$ 
 $m \leftarrow \lfloor h_k(c) \cdot n \rfloor$ ;
Assign  $m$  to  $l_k$ ;
// rescale  $c$  to next interval
 $[c_{\text{left}}, c_{\text{right}}] \leftarrow \{c' \mid \lfloor h_k(c') \cdot n \rfloor = m\}$ ;
 $c \leftarrow \frac{c - c_{\text{left}}}{c_{\text{right}} - c_{\text{left}}}$ 
Assign( $n - m, k - 1, c$ );

```

irrelevant (e.g., sampling “back to camera” as the figure’s position occludes its expression).

- Edges E denote concept hierarchy. For example, the edge (“cat”, “breed”) denotes that “breed” is a sub-concept of “cat”.

To obtain G , we leverage the broad world knowledge encoded in LLMs. Our prompt includes few-shot examples showing how a description d maps to a concept graph G . The examples also guide the LLM towards what constitutes *utility* and *novelty* in the domain. See Appendix A and code for implementation details.

3.2 Assigning Surprise Levels

After constructing the graph G , we assign a *surprise level* to each concept such that the overall surprise follows the properties of §2. We operate under the assumption that there are k surprise levels, and they are **quantitatively** different (i.e., a concept sampled from level $i+1$ contributes more surprise than any number of concepts at levels i or below). In practice, k is small (comparable to the values of a Likert scale).

Computing Allocation Vector. We design a surprise allocation function $Assign$, that takes c , number of concepts n and a number of surprise levels k as parameters. $Assign$ returns an allocation vector $(l_k, l_{k-1}, \dots, l_1)$, $\sum_i l_i = n$, determining the number of concepts in each level.

$Assign$ employs a *top-down hierarchical allocation scheme* (Algorithm 1): It starts at level k (most surprising), determining the portion of concepts assigned to it. The algorithm then continues recursively to distribute portions of the remaining concepts to lower levels until reaching level 1 (least surprising), which receives all remaining concepts.

$Assign$ is designed so that increases in c correspond to (approximately) linear increases in perceived surprise; to compensate for the concave nature of human perception of surprise, $Assign$ employs helper functions h_i , which are convex, surjective, non-decreasing functions in $[0, 1] \rightarrow [0, 1]$ to determine the portion of concepts for each level it reaches. In practice, x^2 and x^3 seem to be working well; see Appendix A for details. If concepts are dynamically added or removed from G , we recompute it dynamically.

Importantly, we design $Assign$ such that:

Theorem 1. *Let n be a positive integer. For any allocation $(l_k, l_{k-1}, \dots, l_1)$ such that $\sum_{i=1}^k l_i = n$, there exists a constant $c \in [0, 1]$ such that $Assign(n, k, c) = (l_k, l_{k-1}, \dots, l_1)$.*

Theorem 2. *For any $c_1, c_2 \in [0, 1]$, $c_1 > c_2$:*

$$Assign(n, k, c_1) \succeq_L Assign(n, k, c_2),$$

where $\succ_L$ denotes the lexicographic order.

In other words, $Assign$ has full coverage of the output space. In particular, when $c = 0$, all concepts occupy the lowest level (fully predictable); when $c = 1$, all are at the highest level. In addition, if $c_1 > c_2$, at the highest level where the assignments differ, the number of concepts in c_1 is strictly greater than c_2 . This design stems from our assumption about surprise levels being quantitatively different, with the top level dominating perceived surprise. See Appendix F for proofs.

Assigning Surprise Levels to Concepts. After computing the number of concepts to assign to each surprise level, we need to assign surprise levels to the actual concepts. We note that some concepts have a very limited outcome space (e.g., “season”) and thus have limited surprise potential. We assign concepts to levels at random according to the computed vector, while prioritizing nodes with higher surprise potential for the higher levels; concepts with limited surprise potential will be assigned higher levels of surprise only if required to satisfy the distribution (see Appendix A for details).

3.3 Sampling Concept values

We traverse G in a breadth-first manner, sampling parent concepts before their sub-concepts (as sub-concepts’ values might depend on the parents’).

For each concept, we first construct a candidate pool of values across k surprise levels, aiming to keep the levels **qualitatively** different. For instance, for concept “*Tree type*” and $k = 3$ levels, level 1 should include conventional options (*oak*), level 2 – unique yet plausible options (*Baobab*), and level 3 – unrealistic options (*tree made of ice*). The candidate pool is generated by prompting an LLM with the updated context (including already-sampled values) and few-shot examples demonstrating the creativity scale. See details in Appendix A. We chose not to rely on the model’s internal surprise estimates for reasons explained in §5.2.

4 Evaluation

To evaluate the effectiveness of Concept- τ for creativity modulation, we assess whether it empirically satisfies the four properties of §2.

4.1 Methods

Concept-level Temperature. In our Concept- τ implementation we use GPT-5.2 (Singh et al., 2026) to construct the graph G and sample values for concepts. For simplicity, we choose to keep G static. We set the number of creativity levels k to 3 (highly predictable values, surprising but plausible, out-of-the-box). See details in Appendix A.

Naive prompting. We prompt gpt-5.2 to generate a detailed T2I prompt at a creativity value in the range 0-10.

As noted earlier, temperature is the primary parameter used to modulate creativity in LLMs. Thus, we construct two natural baselines, which are temperature-based and model-agnostic.

Token Temperature (Token- τ). In this (straightforward) approach, an LLM with controlled temperature is tasked with adding details to the original prompt d . We use Llama-3-8B-Instruct (Grattafiori et al., 2024) since GPT models do not expose or guarantee a standard implementation of temperature, making comparisons unreliable (see Appendix C.5 for additional results on decoupling model and algorithm effects).

Structured Temperature (Structure- τ). To decouple the effect of our concept graph G from our

method of assigning values, we construct a baseline that takes G and uses an LLM (Llama-3-8B-Instruct) with controlled temperature to generate a value for each concept (e.g., The cat’s action is...). This baseline also serves as **ablation**.

See implementation details of all baselines in Appendix B.

4.2 Data

We randomly sampled 20 image descriptions² from the COCO dataset (Lin et al., 2014) (e.g., “Train on a track near many trees”). We use gemini-2.5-flash-image as the T2I generative model M_G . For each description d , we include:

- **Source image:** $M_G(d)$, image generated from d .
- **Concept- τ :** twelve enriched descriptions d' , (three images $\times$ four creativity levels). We select the creativity level values uniformly across the full operational range of c : $c_{\text{low}} = 0$, $c_{\text{mid}} = 0.33$, $c_{\text{high}} = 0.66$, $c_{\text{ext}} = 1$. Each d' is rendered into an image $M_G(d')$.
- **Temperature based methods:** for Token- τ and Structure- τ we follow the same procedure as Concept- τ but slightly modify the operational range: $c_{\text{low}} = 0.1$ (so we do not get deterministic behavior), $c_{\text{mid}} = 0.5$, $c_{\text{high}} = 1$. We set the highest level $c_{\text{ext}} = 1.5$ (with top- $k = 50$) as higher values produced incoherent outputs and significantly degraded utility (see Appendix B).
- **Naive Prompting** for Naive- τ we set $c_{\text{low}} = 0$, $c_{\text{mid}} = 3.5$, $c_{\text{high}} = 6.5$, $c_{\text{ext}} = 10$.

4.3 Distance

To perform a Quantitative Analysis, a basic building block we need to define is a distance metric. Since text-based embeddings are highly sensitive to linguistic features irrelevant to the visual scene (length, structure, etc.), and the enriched texts differ significantly in such features, we instead measure distances in the visual domain. Specifically, we compute cosine distance between CLIP image embeddings (Radford et al., 2021).

4.4 User Study Settings

To evaluate how Concept- τ compares to the baselines from a human perspective, we collect human annotations through Prolific.com, recruiting 50 participants from the US and UK with approval rate

²Datasets in the area of creative image generation are typically of comparable size (Richardson et al., 2024; Han et al., 2025b).

$\geq 98\%$, ≥ 500 submissions. We compensate participants at a rate of 1.2 GBP (12.4 GBP per hour).

Annotators are asked to rank images across four creativity levels, from 1 (*Low*) to 4 (*Extreme*) (see Appendix D); a screening task ensured they understood the instructions. We sampled a random description for each of the four methods (order of methods was randomized), showing the participant all 12 generated images for this method. In total, we collect 2,400 annotations. No PII was collected beyond Prolific’s basic demographics.

4.5 Results

We now review the results for each property of §2.

Monotonic Linear Novelty (P1) A creativity parameter should exhibit a monotonic, roughly linear relationship with the perceived novelty of the generated output. To evaluate whether this holds in our method, we compute Pearson and Kendall-tau correlations between human annotations and the creativity levels c from which they were derived.

Table 1 (left) shows that *Concept- τ* achieves **significantly higher linear and monotonic correlations** than all baseline methods. Results of Spearman correlations are similar (Appendix C).

Refer to Figure 1 and Appendix C to see how *Concept- τ* increases output novelty with c , *Token- τ* and *Structure- τ* exhibit stable (low) novelty, and *Naïve- τ* only increases novelty at the extreme level.

To **quantitatively** reinforce these findings, we assign each output a *quantitative surprise score* which we correlate with c . Intuitively, outputs that deviate further from their source descriptions are more novel. We compute the distance (Section 4.3) between $M_G(d')$ and $M_G(d)$. Table 1 shows the correlation. The results closely mirror the human evaluation: *Concept- τ* exhibits **stronger correlations** with c than baselines across all metrics.

As a reality check, we verify the quantitative surprise scores correlate with human judgments. *Concept- τ* , *Token- τ* , *Structure- τ* , *Naïve- τ* achieved Pearson correlation of 0.73, 0.32, 0.04, 0.77 respectively (for *Structure- τ* , humans showed near-zero correlation with c , so this was expected).

Spanning a Wide Novelty Range (P2) To evaluate how effectively the creativity parameter covers the novelty range, we report the average human ratings for images generated with c_{low} , c_{mid} , c_{high} and c_{ext} (Figure 4, left). Ratings were on a scale of 1 (low) to 4 (extreme). *Concept- τ* ’s perceived surprise increases markedly with c , covering the

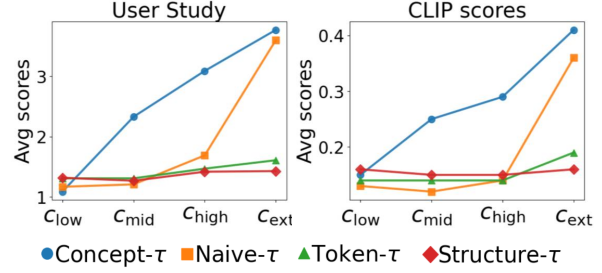


Figure 4: Averages of user scores and CLIP distances by creativity level. *Concept- τ* successfully covers a wide novelty range with high and low value at extremes. Scores are detailed in Appendix C.

entire range. Thus, our algorithm **achieves P2**. In contrast, *Token- τ* and *Structure- τ* achieve novelty scores only slightly higher than minimum. *Naïve- τ* achieves top and bottom levels but does not capture the intermediate range.

We corroborate this result **quantitatively** using the distance-based *quantitative surprise scores* introduced above (Figure 4, right).

Diversity (P3) To quantitatively measure diversity, we report average CLIP distances between all pairs of images of the same scene $M_G(d')$ (Table 1, middle). *CONCEPT- τ* exhibits **substantially higher diversity** than the baseline approaches.

We further examine how diversity varies with c . We expect that as c increases, both *intra-level* and *inter-level* diversity will rise. Figure 5 presents the average cosine distances between CLIP embeddings for all pairs of creativity levels (distances measured only between images of the same scene).

For *Concept- τ* , we note that inter-level diversity (off-diagonal) generally increases: images generated at more distant creativity levels differ substantially. As for intra-level diversity (diagonals), images at c_{low} are much more similar to each other than those at higher levels. In contrast, baselines exhibit lower and less structured diversity.

Utility (P4) We define utility for the T2I domain as prompt faithfulness: the output should satisfy d .

By design, our framework explicitly attempts to preserve the information in d . Consider the prompt “A pizza on a white plate.” Concepts specified in d (“plate color = white”) become *utility* concepts, and we do not sample their value. New concepts (“plate material”) become *novelty* concepts.

Novelty concepts may introduce contradictions; e.g., sampling “rubies” for material would contra-

Method	User Study		CLIP			Utility		
	τ	ρ	Diversity	τ	ρ	$\text{err}_{\text{prompt}}$	$\text{err}_{\text{image}}$	$\text{err}_{\text{quality}}$
Concept- τ	0.75 ± 0.02	0.83 ± 0.02	0.3 ± 0.0	0.59 ± 0.03	0.74 ± 0.03	0.02 ± 0.01	0.07 ± 0.02	0.02 ± 0.01
Token- τ	0.16 ± 0.04	0.17 ± 0.04	0.14 ± 0.0	0.14 ± 0.05	0.2 ± 0.06	0.03 ± 0.01	0.05 ± 0.01	0.93 ± 0.02
Structure- τ	0.1 ± 0.04	0.11 ± 0.04	0.12 ± 0.0	0.01 ± 0.05	0.03 ± 0.06	0.0 ± 0.0	0.02 ± 0.01	0.72 ± 0.03
Naïve- τ	0.66 ± 0.02	0.75 ± 0.02	0.21 ± 0.0	0.44 ± 0.04	0.64 ± 0.04	0.0 ± 0.0	0.01 ± 0.01	0.0 ± 0.0

Table 1: **Left:** Pearson (ρ) and Kendall- τ (τ) correlation between the user study results and the creativity levels used to generate the images. **Middle:** Pearson and Kendall- τ correlation using CLIP scores and diversity score (average CLIP distances). **Right:** Error rates for unfaithfulness in d' ($\text{err}_{\text{prompt}}$), unfaithfulness in $M_G(d')$ ($\text{err}_{\text{image}}$) and poor text quality ($\text{err}_{\text{quality}}$). Results in **bold** are best. Values following $\pm$ indicate standard errors. Concept- τ significantly outperforms the baselines in terms of correlation, diversity, and text quality, while exhibiting a marginally higher error rate in output faithfulness. Full correlations, including Spearman, are available in Appendix C

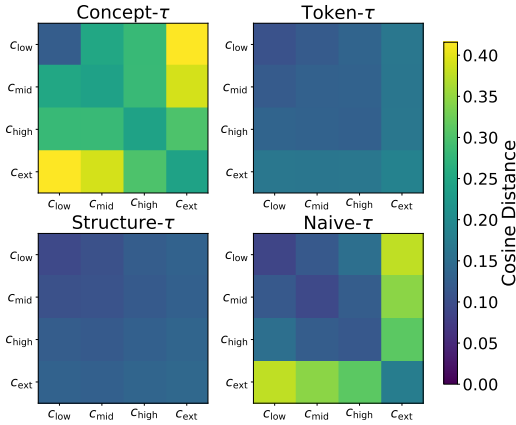


Figure 5: CLIP-based average cosine distances between creativity levels. Cell (i, j) shows distance between all images generated at levels i, j . Concept- τ outperforms the baselines in *intra-level* and *inter-level* diversity.

dict the white color³. To assess the prevalence and type of such violations, we manually annotated all outputs with respect to three types of potential errors: Unfaithfulness in d' – contradictions introduced in the augmented text; unfaithfulness in $M_G(d')$ – contradictions stemming from the text-to-image model M_G ; and low quality, where d' contains irrelevant information or gibberish.

From the results (Table 1, right) we observe that contradictions are rare across all approaches, particularly at the augmentation stage. As expected, Token- τ and Structure- τ produce a substantial number of poorly written augmentations. Additional details are provided in Appendix C.

³Sampling is done in context of known information, so it is unlikely to sample rubies for a white plate; however, LLMs make mistakes.

5 Discussion

5.1 Generalization to other domains

We have demonstrated our ideas on text-to-image, but the framework we have presented is **general**, and can be adapted to a range of creativity-driven domains such as short story, script, and recipe generation. We now demonstrate a proof-of-concept of adapting Concept- τ to recipe generation. Importantly, the overall algorithm remains consistent with that used for T2I generation. The primary differences are (1) different in-context examples, tailored to the culinary domain, and (2) the definition of *utility*, in addition to being faithful to the prompt, should reflect domain-specific aspects (i.e., the recipe must include only real, edible ingredients, and taste good). In Figure 6 we show examples of recipes generated for the prompt “burger”, sampled at different creativity levels (images generated for the sake of presentation; the outputs in this domain are purely textual). Although the formalization of “utility” warrants further refinement (particularly around the “tastes good” aspect), we find these outputs encouraging. See Appendix E.2 for more details on our proof-of-concept.

5.2 LLM internal probabilities

A natural question to ask is why we generate candidates of different levels of surprise using prompting instead of ranking candidates via self-information.

While appealing in theory, a core limitation of using LLM internal probabilities is that they are not trained to model *visual* surprise. For example, the table below shows self-information for several sentence completions. Numbers indicate $-\log(p)$, so candidates are sorted from less surprising to more surprising; colors indicate our approach’s judgment of surprise, where darker is more surprising. Thus, according to the LLM’s internal probabilities, find-



Figure 6: Proof-of-concept in a different domain: Examples of burger recipes generated at different creativity levels. Images generated for the sake of presentation; the outputs in this domain are purely textual.

ing a twig in a forest is less likely than a piano or a ship. While this may be true textually, it is counterintuitive in a visual context; in contrast, our approach correctly captures human intuition.

The girl's hair color is ____	The type of tree is ____	In the forest there is a ____
① brown 2.7	① oak 6.3	① tree 3.8
③ green 5.6	③ inverted 17.8	③ piano 13.5
③ rainbow 13.4	③ stone 18.0	③ ship 14.3
② almond 16.0	② cactus 18.4	① shrub 16.3
② cocoa 17.2	② joshua 21.4	① twig 16.5

6 Related Work

Image Creativity Enhancement Several works have attempted to enhance image creativity. ConceptLab (Richardson et al., 2024) aims to generate new members of a broad semantic category (e.g., *pet*), while C3 (Han et al., 2025b) directly amplifies feature maps during the denoising process. In contrast to Concept- τ , these approaches are architecture- and model-dependent and often fail to generalize to complex, constraint-laden scenes. POET (Han et al., 2025a) increases output diversity by expanding prompts along automatically discovered homogeneous dimensions derived from generated images. Although both approaches share a similar sub-task of detecting missing concepts, Concept- τ detects concepts that are comparable to POET while offering a simpler, more computationally efficient, and more interpretable framework. In addition, all of these approaches do not explicitly address creative modulation or the formulation of a controllable creativity parameter.

LLM Creativity Enhancement Previous work explored LLM creative enhancement through methods such as recombination (Mizrahi et al., 2026) (the process of combining several elements to create something new) or reinforcement learning with

“A train on a train track near many trees”

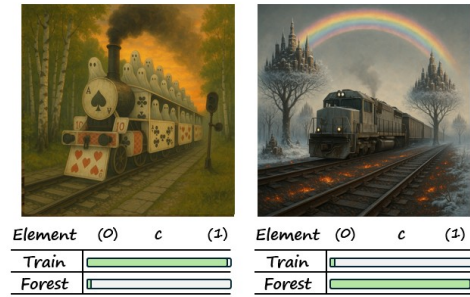


Figure 7: Concept- τ can control different aspects of an image at varying creativity levels. In this example we adjust the creativity of “train” and “forest” individually, rather than changing the global image creativity.

signals relating to surprise and diversity (CrPO (Ismayilzada et al., 2025)) or denial prompting (Lu et al., 2025). While these methods improve creative output quality, they do not explore mechanisms for controllable creative modulation. CrPO requires retraining the model to alter its creativity level, whereas our approach operates at inference time and enables direct control over creativity.

7 Conclusion and Future Work

We introduced Concept- τ , a novel framework for creative modulation and enhancement in generative models, and demonstrated it in the text-to-image domain. Concept- τ explores the conceptual space left undefined by textual prompts, enabling explicit and interpretable control over creative generation.

We have defined properties that a creativity parameter should satisfy, and demonstrated that Concept- τ satisfies them, significantly outperforming baselines. We release data and code.

In the future, we envision applying the method to **other domains** (see Section 5.1). Another straight-

forward extension is to allow more **fine-grained control** via our structured representation G . Figure 7 demonstrates how we can easily use our approach to control creativity of specific parts of an image locally (train vs. forest), rather than globally.

Finally, the framework can also be **inverted** to compute a distance metric between a prompt and an output, quantifying the creativity of the output (see Appendix E.1 for more details on this direction).

More broadly, our framework aligns with the emerging paradigm of interactive creative exploration (Baek et al., 2023; Adamkiewicz et al., 2025; Brade et al., 2023). By enabling users to modulate creativity at the level of semantic decisions, it opens new possibilities for systematically exploring the space of outputs induced by a prompt. We hope this perspective will advance the understanding and control of creativity in generative models.

Limitations

Generalizability. Our framework is general but not domain-agnostic. To apply it to a new domain (e.g., stories, poems, recipes), one might need to tweak the few-shot examples to reflect the domain, as well as notions of surprise and utility (see Appendix E.2).

Inappropriate Attributes. LLM-generated attributes might not be appropriate, e.g., “Invisible” for “Cat fur”. Though rare, we believe such issues could be ameliorated with improved prompting, fine-tuning, or filtering.

Mode collapse. While our approach reduces mode collapse and increases diversity by sampling at the concept level (vs. sampling the whole scene), mode collapse still occurs, especially for concepts sampled at high surprise levels (repetition of clocks, galaxies, etc.). We believe this issue can be mitigated by inserting random external concepts for inspiration.

Reliance on Powerful Models. Generating candidates for each concept can be computationally expensive, as it relies on powerful LLMs. However, we believe that this cost can be substantially reduced through common techniques such as caching frequently used elements and attributes and distilling knowledge into smaller models.

Ethical Considerations

This study poses minimal direct ethical risks. However, the concept of novelty and creativity is cul-

turally dependent and socially constructed. Because our evaluation relies on outputs generated by a large language model, the notion of novelty reflected in the results may be influenced by biases present in the model’s training data.

We used a large language model (LLM) solely for paraphrasing purposes, and all generated outputs were carefully reviewed and verified for accuracy.

Acknowledgments

This work was supported by the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant no. 852686, SIAM).

References

- Krzysztof Adamkiewicz, Paweł W. Woźniak, Julia Dominiak, Andrzej Romanowski, Jakob Karolus, and Stanislav Frolov. 2025. [Promptmap: An alternative interaction style for ai-based image generation](#). In *Proceedings of the 30th International Conference on Intelligent User Interfaces, IUI ’25*, page 1162–1176, New York, NY, USA. Association for Computing Machinery.
- Seungho Baek, Hyerin Im, Jiseung Ryu, Juhyeong Park, and Takyen Lee. 2023. [Promptcrafter: Crafting text-to-image prompt through mixed-initiative dialogue with llm](#). *Preprint*, arXiv:2307.08985.
- Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Qinsheng Zhang, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, Bryan Catanzaro, Tero Karras, and Ming-Yu Liu. 2023. [ediff-i: Text-to-image diffusion models with an ensemble of expert denoisers](#). *Preprint*, arXiv:2211.01324.
- Giulio Biroli, Tony Bonnaire, Valentin De Bortoli, and Marc Mézard. 2024. Dynamical regimes of diffusion models. *Nature Communications*, 15(1):9957.
- Margaret A. Boden. 1991. *The creative mind: myths and mechanisms*. Basic Books, Inc., USA.
- Stephen Brade, Bryan Wang, Mauricio Sousa, Sageev Oore, and Tovi Grossman. 2023. [Promptify: Text-to-image generation through interactive prompt exploration with large language models](#). In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology, UIST ’23*, New York, NY, USA. Association for Computing Machinery.
- Stanley L. Brue. 1993. [Retrospectives: The law of diminishing returns](#). *Journal of Economic Perspectives*, 7(3):185–192.
- Edwin Cannan. 1892. [The origin of the law of diminishing returns, 1813-15](#). *The Economic Journal*, 2(5):53–69.

- Prafulla Dhariwal and Alex Nichol. 2021. Diffusion models beat gans on image synthesis. In *Proceedings of the 35th International Conference on Neural Information Processing Systems, NIPS '21*, Red Hook, NY, USA. Curran Associates Inc.
- Ming Ding, Wendi Zheng, Wenyi Hong, and Jie Tang. 2022. Cogview2: faster and better text-to-image generation via hierarchical transformers. In *Proceedings of the 36th International Conference on Neural Information Processing Systems, NIPS '22*, Red Hook, NY, USA. Curran Associates Inc.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. [The llama 3 herd of models](#). *Preprint*, arXiv:2407.21783.
- John Hale. 2001. [A probabilistic earley parser as a psycholinguistic model](#). In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics on Language Technologies, NAACL '01*, page 1–8, USA. Association for Computational Linguistics.
- Evans Xu Han, Alice Qian Zhang, Haiyi Zhu, Hong Shen, Paul Pu Liang, and Jane Hsieh. 2025a. Poet: Supporting prompting creativity and personalization with automated expansion of text-to-image generation. In *Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology*, pages 1–18.
- Jiyeon Han, Dahee Kwon, Gayoung Lee, Junho Kim, and Jaesik Choi. 2025b. Enhancing creative generation on stable diffusion-based models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 28609–28618.
- Jonathan Ho and Tim Salimans. 2021. [Classifier-free diffusion guidance](#). In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*.
- Mete Ismayilzada, Antonio Laverghetta Jr., Simone A. Luchini, Reet Patel, Antoine Bosselut, Lonneke van der Plas, and Roger E. Beaty. 2025. [Creative preference optimization](#). In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 9580–9609, Suzhou, China. Association for Computational Linguistics.
- Laurent Itti and Pierre Baldi. 2009. [Bayesian surprise attracts human attention](#). *Vision Research*, 49(10):1295–1306. Visual Attention: Psychophysics, electrophysiology and neuroimaging.
- Sophie Jentzsch and Kristian Kersting. 2023. [ChatGPT is fun, but it is not funny! humor is still challenging large language models](#). In *Proceedings of the 13th Workshop on Computational Approaches to Subjectivity, Sentiment, & Social Media Analysis*, pages 325–340, Toronto, Canada. Association for Computational Linguistics.
- Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. 2017. Building machines that learn and think like people. *Behavioral and brain sciences*, 40:e253.
- Roger Levy. 2008. [Expectation-based syntactic comprehension](#). *Cognition*, 106(3):1126–1177.
- Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *Computer Vision – ECCV 2014*, pages 740–755, Cham. Springer International Publishing.
- Yining Lu, Dixuan Wang, Tianjian Li, Dongwei Jiang, Sanjeev Khudanpur, Meng Jiang, and Daniel Khashabi. 2025. [Benchmarking language model creativity: A case study on code generation](#). In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2776–2794, Albuquerque, New Mexico. Association for Computational Linguistics.
- Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. 2024. Large language models: A survey. *arXiv preprint arXiv:2402.06196*.
- Moran Mizrahi, Chen Shani, Gabriel Stanovsky, Dan Jurafsky, and Dafna Shahaf. 2026. [Cooking up creativity: Enhancing LLM creativity through structured recombination](#). *Transactions of the Association for Computational Linguistics*, 14:418–441.
- Michael D. Mumford. 2003. [Where have we been, where are we going? taking stock in creativity research](#). *Creativity Research Journal*, 15(2-3):107–120.
- Alexander Quinn Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. 2022. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. In *International Conference on Machine Learning*, pages 16784–16804. PMLR.
- Max Peeperkorn, Tom Kouwenhoven, Dan Brown, and Anna Jordanous. 2024. Is temperature the creativity parameter of large language models? In *15th International Conference on Computational Creativity*. Association for Computational Creativity.
- Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. 2024. Sdxl: Improving latent diffusion models for high-resolution image synthesis. In *International Conference on Learning Representations*, volume 2024, pages 1862–1874.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiayi Yang, Jingren Zhou, and 25 others. 2025. [Qwen2.5 technical report](#). *Preprint*, arXiv:2412.15115.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, and 1 others. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. 2022. [Hierarchical text-conditional image generation with clip latents](#). *Preprint*, arXiv:2204.06125.

Elad Richardson, Kfir Goldberg, Yuval Alaluf, and Daniel Cohen-Or. 2024. Conceptlab: Creative concept generation using vlm-guided diffusion prior constraints. *ACM Transactions on Graphics*, 43(3):1–14.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Bjorn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10674–10685. IEEE Computer Society.

C. E. Shannon. 1948. [A mathematical theory of communication](#). *The Bell System Technical Journal*, 27(3):379–423.

Dazhong Shen, Guanglu Song, Zeyue Xue, Fu-Yun Wang, and Yu Liu. 2024. Rethinking the spatial inconsistency in classifier-free diffusion guidance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9370–9379.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, Akshay Nathan, Alan Luo, Alec Helyar, Aleksander Madry, Aleksandr Efremov, Aleksandra Spyra, Alex Baker-Whitcomb, Alex Beutel, Alex Karpenko, and 467 others. 2026. [Openai gpt-5 system card](#). *Preprint*, arXiv:2601.03267.

Shimao Zhang, Yu Bao, and Shujian Huang. 2024. [Edt: Improving large language models’ generation by entropy-based dynamic temperature sampling](#). *Preprint*, arXiv:2403.14541.

A Concept- τ Technical Details

Here we discuss the full technical details for the implementation of Concept- τ for image generation.

A.1 Constructing G

Concept Graph Construction For constructing the concept graph, we employ two in-context examples to probe the language model (gpt-5.2). In our adaptation for image generation, concepts correspond to *visual objects and attributes* – for example, *cat breed*, *fur appearance*, *age*, *action*, etc. We prompt the LLM to construct the graph of missing concept as a python file so we can parse it later. We start with an initial prompt and then chain the output to an additional editing prompt which checks for potential errors. We demonstrate below one of the python classes generated through the process as an example of G . The prompts themselves are available in the code files.

Below is an example of G corresponding to the input $d = \text{“A train on a train-track near many trees.”}$

```
class Train:
    def __init__(self):
        self.type = get_options("train-type") #
        ↪ The train type is <MASK>. # Examples:
        ↪ (modern passenger train, Level1),
        ↪ (steam locomotive, Level2),
        ↪ (levitating maglev with no visible
        ↪ wheels, Level3). # train type
        self.appearance =
        ↪ get_options("train-appearance") #
        ↪ The train appearance is <MASK>. #
        ↪ Examples: (painted metal with
        ↪ windows, Level1), (graffiti-covered
        ↪ and patched, Level2), (made of
        ↪ stained glass with moving
        ↪ constellations inside, Level3). #
        ↪ train appearance
        self.condition = select(["New",
        ↪ "Well-maintained", "Worn", "Rusty",
        ↪ "Repaired"]) # The train condition
        ↪ is <MASK>. # train condition
        self.cargo_or_passengers = get_options("
        ↪ train-cargo-or-passengers") # The
        ↪ train carries <MASK>. # Examples:
        ↪ (commuters visible through windows,
        ↪ Level1), (stacked timber and crates,
        ↪ Level2), (a traveling greenhouse
        ↪ ecosystem in each car, Level3). #
        ↪ what the train carries
        self.extra_detail =
        ↪ get_options("extra-detail") # The
        ↪ train has an extra detail which is
        ↪ <MASK>. # Examples: (none, Level1),
        ↪ (a vintage brass nameplate and
        ↪ whistle, Level2), (a ribbon of
        ↪ floating lanterns orbiting the
        ↪ engine, Level3). # train extra detail

class TrainTrack:
    def __init__(self):
```

```

self.type = get_options("traintrack-type")
↪ # The traintrack type is <MASK>. #
↪ Examples: (standard steel rails on
↪ wooden sleepers, Level1), (old
↪ narrow-gauge track, Level2), (crystal
↪ rails that glow when the train
↪ approaches, Level3). # track type
self.condition = select(["New",
↪ "Maintained", "Overgrown",
↪ "Weathered", "Damaged"]) # The track
↪ condition is <MASK>. # track
↪ condition
self.surrounding_ground = get_options("t
↪ rack-surrounding-ground") # The
↪ ground around the track is <MASK>. #
↪ Examples: (gravel ballast and dirt,
↪ Level1), (wildflowers and tall grass,
↪ Level2), (soft moss that forms
↪ patterns like circuitry, Level3). #
↪ ground around track

class TreesArea:
    def __init__(self):
        self.tree_type = get_options("tree-type")
        ↪ # The trees are <MASK>. # Examples:
        ↪ (pines and oaks, Level1), (cherry
        ↪ blossoms, Level2), (towering trees
        ↪ with translucent bark and visible
        ↪ flowing sap-light, Level3). # tree
        ↪ type
        self.density = select(["Sparse",
        ↪ "Scattered", "Many", "Dense",
        ↪ "Impenetrable"]) # The tree density
        ↪ is <MASK>. # tree density
        self.understory =
        ↪ get_options("forest-understory") #
        ↪ The understory is <MASK>. # Examples:
        ↪ (ferns and bushes, Level1),
        ↪ (mushrooms and fallen logs, Level2),
        ↪ (bioluminescent vines forming
        ↪ archways, Level3). # understory

class Scene:
    def __init__(self):
        self.train = Train() # dictated by
        ↪ initial prompt "A train"
        self.traintrack = TrainTrack() # dictated
        ↪ by initial prompt "on a traintrack"
        self.trees_area = TreesArea() # dictated
        ↪ by initial prompt "near many trees"

```

Each line’s structure and associated comments are later used both to route the concept to its corresponding process and to serve as input for the candidate generation stage.

In our adaptation, the concept graph G is kept **static**. This choice reflects the practical limitations of current text-to-image models: excessive numbers of dynamically added concepts typically degrade image quality because fine-grained details cannot be reliably visualized.

A.2 Assign Parameters

Here we specify the parameters of the surprise allocation function *Assign*:

1. $k = 3$: the number of creativity levels;
2. Level 3 helper function: $h_3(x) = x^3$;

3. Level 2 helper function: $h_2(x) = x^2$.

Helper functions h_i should be convex, surjective, non-decreasing functions in $[0, 1] \rightarrow [0, 1]$ to determine the portion of concepts for each level it reaches. In practice, we observed that both x^2 and x^3 seem to be working well. We choose $h_3(x) = x^3$ and $h_2(x) = x^2$ to reflect the stronger qualitative jump in perceived surprise at level 3. Since level 3 represents the most surprising outcomes, we use the steeper function x^3 so that $h_3(c)$ stays small until c is sufficiently high. In contrast, the helper function x^2 allows level-2 concepts to appear earlier as c increases.

A.3 Candidate Generation

When we generate a candidate pool for a concept we route it into one of two types of processes:

1. Limited candidate mapping - In the construction of G some attribute concepts are inherently limited in number and creativity level of candidates. For instance, “Season” has only four potential values: Winter, Summer, Spring, and Fall. The candidates for such concepts are already generated when G is constructed. For such a concept we simply use a prompt to map the already established candidates into creativity levels.
2. Attribute generation - We generate a long list of options in $k = 3$ surprise levels using few shot learning.

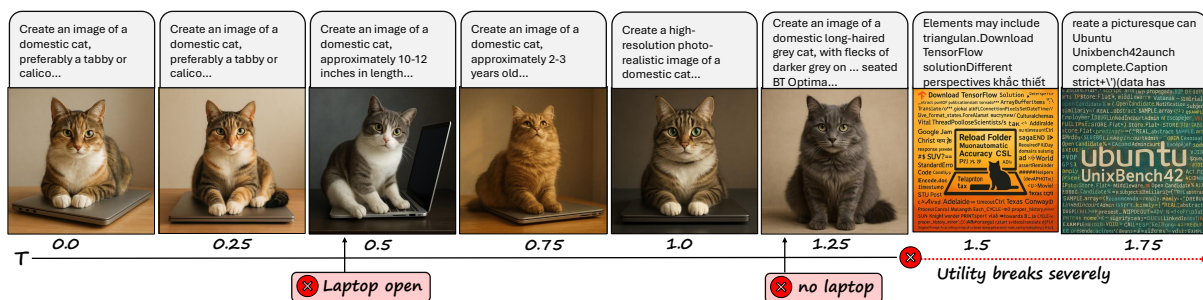
The routing to those processes is done using the representation of G . The full prompts are available in the code files.

B Baselines Implementation Details

B.1 Choice of Maximal Temperature

As we show in Figure 8, temperature tends to compromise text utility when set too high. To make sure all baseline outputs adhere to the utility constraint, we first set $k = 50$ so that sampled tokens would not be completely random (we try to avoid tokens such as Chinese characters which tend to appear when k is not limited). Secondly, we limit the maximal temperature to 1.5 as values set higher significantly harm legibility. To demonstrate we provide a full textual example of Token- τ for an output generated at temperature $t = 2$ (see Figure 9). For generation we use as the LLM Llama3-8b-instruct (same model we use for all results for Token- τ).

"A cat sitting on top of a closed laptop"



"A red-haired girl in the forest with her back to the camera"

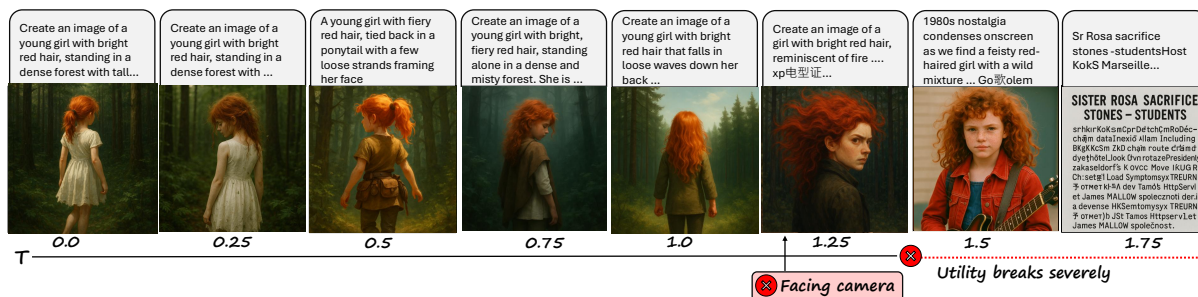


Figure 8: Demonstration of the limitations of token-level temperature using both the generated text (the beginning of the output) and the images derived from that text. In both examples, we observe that surprise does not increase substantially with t until t becomes too large, at which point utility is severely compromised. For instance, in the prompt "Cat on a laptop," increasing t initially leads to minor degradations – the laptop may be open when it should be closed, or may disappear entirely –until eventually, at sufficiently high t , the output collapses into gibberish. We observe similar behavior with "girl in the forest," where the subject may no longer face the camera or the forest may disappear entirely. This pattern is clearly visible in the text outputs themselves, which gradually become nonsensical as temperature increases.

Input: Augment the following prompt into a detailed text-to-image prompt: A pizza and cutter on a white plate and a drink.

Output: Create an image of a round white plate lying flat on a surface with 360 deg surroundings not visible. Place exactly one freshly cooked, partially cut juicy, mouth-watering-looking cheese covered vegetarian pizza on the center-upper quadrant of the plate. Arrange seven or eight large pepperocini mushroom-slices harmoniously evenly distributed on half of the pizza in a semi-grid like order. Scatter six cherry tomatoes on around six distinct spots without overlap across the other half, forming an attractive visually balance. To enhance the composition, add fresh basil leaves artfully drizzled on top, creating a gentle pattern reminiscent of the pepperocini distribution On the very center-top quarter area of the plate place ONE crisp-new rectangular glass glass-pint of clear ice-cold pure liquid drink on white plate on table, which has the slight reflection of LED-lensed lighting to catch eye. Show exactly on ONE side view of any angle facing 40+45-5 as its viewing direction and slightly leaning, almost-touching or brushing against table surface, because of its curved corner touching flat table, this creates a clean visual connection. It seems an informal-daily or take-out eating setting. Background must have pure white un-pattern-no-object as to visually contrast strong image's colors.

Figure 9: Textual example of Token- τ when the temperature is $t = 2$.

B.2 Token- τ Algorithm Details

In this (straightforward) approach, we use Llama-3-8B-Instruct to add details to the original prompt d , where we control its temperature. We use the following prompt:

Augment the following prompt into a detailed T2I prompt: "{d}"
Detailed prompt:

B.3 Structure- τ Algorithm Details

Obtaining the template T To implement Structure- τ , we first need to construct T from d .

In building G (represented as a Python file), we ensured that each concept is accompanied by a *comment containing a completion sentence*. This sentence explicitly encodes the concept with a <MASK> placeholder.

For example, for the concept *cat's fur color*, the associated context sentence is:

the cat's fur color is <MASK>.

To construct T we simply concatenate all context sentences.

Running Structure- τ To obtain d' using T we simply go over each sentence in T , remove the mask, obtain a completion from the LLM using the predefined temperature parameter, and augment d with the sentence and its completions.

Attribute Generation

Imagine the following image {d}.
Your goal is to choose a value for {value}.

Your goal is to complete the following
→ sentence: {masked sentence}
Important - Write ONLY the completion
→ without any explanations or comments.
Answer:

B.4 Naïve- τ Details

We use gpt-5.2 and the following prompt with temperature of 0.7:

You are an expert prompt engineer specializing in
→ text-to-image generation.

You will get two inputs:
1. A scene
2. a creativity level from 0-10, where 0
→ represents minimal creativity and 10
→ represents maximal creativity.

Your goal is to augment the input scene such that
→ the resulting text-to-image prompt will be at
→ the desired creativity level.

We define creativity to be a combination of
→ novelty and utility.
All outputs must maintain utility by being
→ faithful to the input scene.
The novelty level of the output prompts should
→ scale linearly with the specified creativity
→ level.

Minimal creativity: outputs should be highly
→ ordinary - "the first scene one would
→ imagine".
Medium creativity: outputs should be somewhat
→ novel while remaining reasonable.
Maximal creativity: outputs should be extremely
→ novel - "One would never imagine this image
→ given the source."

Guidelines:
Output must start with the input scene.
Output only the final text-to-image prompt.

Scene:
{d}
Creativity level:
{level}

C Additional Results

We present here additional results and details for the user study and computational studies.

C.1 Qualitative Comparison

We present in Figure 10 a qualitative comparison of the output images between Concept- τ and the baselines.

C.2 Additional computational scores

We include in Table 3 detailed results of the average novelty scores obtained at different creativity levels. Table 2 includes full correlation results (Pearson, Spearman and Kendall-tau).

C.3 Correlating Computational Surprise Scores

In our quantitative analysis we use quantitative novelty scores that measure cosine distance between outputs: $M_G(d)$ to $M_G(d')$ using CLIP. To justify using these quantitative scores, we verify in Table 4 that they correlate with human judgment scores. On Concept- τ , Naïve- τ and Token- τ , the table shows clear positive correlations, indicating that the metrics capture variation in surprise in a way that aligns with human intuition. For Structure- τ , the low correlation is expected: humans themselves showed near-zero correlation with c , suggesting that their labels were effectively random in those settings.

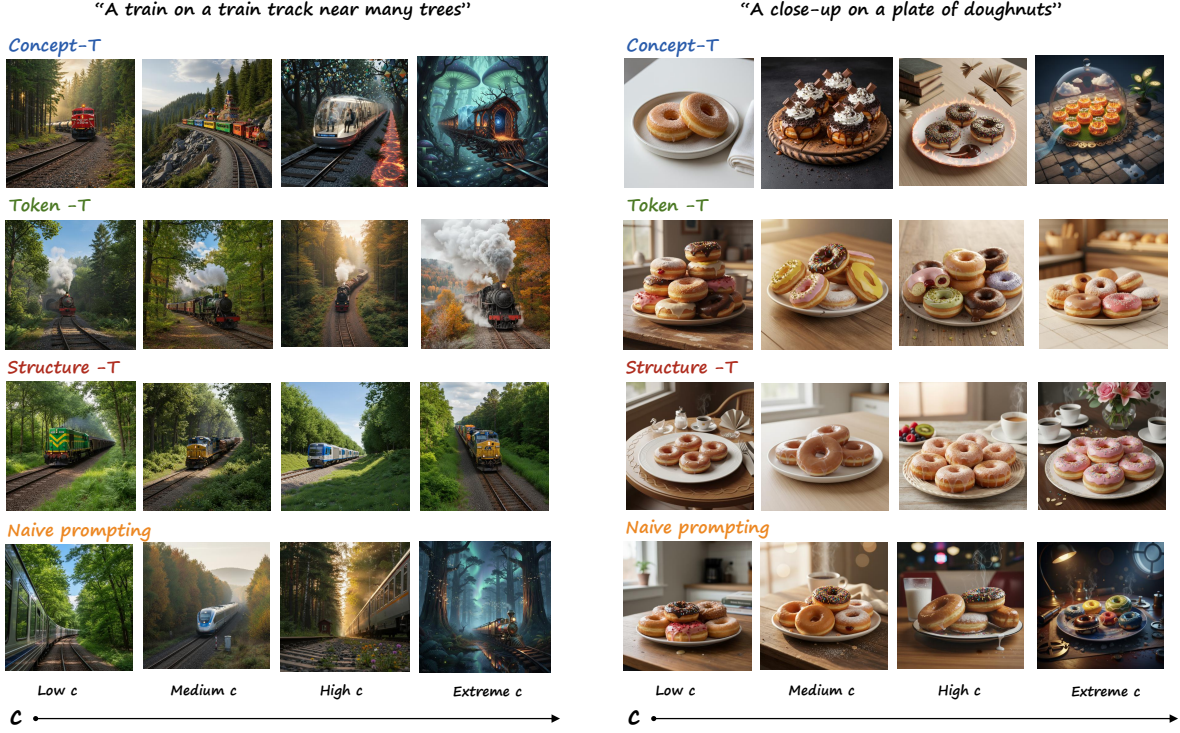


Figure 10: Comparison between Concept- τ , Token- τ , Structure- τ and Naïve- τ as the creativity parameter increases

Method	User Study			CLIP		
	τ	ρ	ρ_s	τ	ρ	ρ_s
Concept- τ	0.75 ± 0.02	0.83 ± 0.02	0.82 ± 0.02	0.59 ± 0.03	0.74 ± 0.03	0.74 ± 0.03
Token- τ	0.16 ± 0.04	0.17 ± 0.04	0.18 ± 0.04	0.14 ± 0.05	0.20 ± 0.06	0.19 ± 0.06
Structure- τ	0.10 ± 0.04	0.11 ± 0.04	0.09 ± 0.04	0.01 ± 0.05	0.03 ± 0.06	0.01 ± 0.06
Naïve- τ	0.66 ± 0.02	0.75 ± 0.02	0.74 ± 0.02	0.44 ± 0.04	0.64 ± 0.04	0.58 ± 0.05

Table 2: Pearson correlation (ρ), Spearman correlation (ρ_s) and Kendall- τ (τ) across user study and CLIP. Correlations are between user/CLIP scores and the creativity levels from which the images were generated. Concept- τ significantly outperforms the baselines.

Method	User Study				CLIP			
	c_{low}	c_{med}	c_{high}	c_{ext}	c_{low}	c_{med}	c_{high}	c_{ext}
Concept- τ	1.09 ± 0.03	2.33 ± 0.07	3.09 ± 0.07	3.77 ± 0.04	0.15 ± 0.01	0.25 ± 0.01	0.29 ± 0.01	0.41 ± 0.01
Token- τ	1.31 ± 0.05	1.31 ± 0.05	1.47 ± 0.06	1.61 ± 0.06	0.14 ± 0.01	0.14 ± 0.01	0.14 ± 0.01	0.19 ± 0.01
Structure- τ	1.32 ± 0.05	1.27 ± 0.05	1.42 ± 0.05	1.43 ± 0.06	0.16 ± 0.01	0.15 ± 0.01	0.15 ± 0.01	0.16 ± 0.01
Naïve- τ	1.17 ± 0.04	1.21 ± 0.04	1.69 ± 0.06	3.6 ± 0.05	0.13 ± 0.01	0.12 ± 0.01	0.14 ± 0.01	0.36 ± 0.01

Table 3: Comparison of methods across user study and CLIP average scores for c_{low} , c_{mid} , c_{high} and c_{ext} . Values following $\pm$ indicate standard errors. Concept- τ successfully covers a wide novelty range with high scores for extreme creativity level and low scores for low creativity level.

Metric	Concept- τ	Token- τ	Structure- τ	Naïve- τ
CLIP	0.73 ± 0.033	0.32 ± 0.07	0.036 ± 0.068	0.77 ± 0.029

Table 4: Pearson correlation (ρ) between human labels and CLIP novelty scores.

C.4 Utility Evaluation

We detail here our validation process for the utility metrics. All outputs were evaluated with respect to three types of potential errors:

- UMG — unfaithfulness in $M_G(d')$

- Ud' — unfaithfulness in d'
- Uquality — poorly written augmentations

For UMG, we manually validated all generated images. For Ud' and Uquality, we used gpt-5.2 with two validation prompts provided in the code-

Method	τ	ρ	ρ_s
Concept- τ	0.64 ± 0.02	0.8 ± 0.02	0.8 ± 0.02
Token- τ	0.08 ± 0.05	0.13 ± 0.06	0.1 ± 0.06
Structure- τ	0.04 ± 0.05	0.07 ± 0.06	0.05 ± 0.07

Table 5: Pearson correlation (ρ), Spearman correlation (ρ_s), and Kendall- τ (τ) across CLIP when using Qwen as the LLM for all approaches. The results show that even when compared over the same model, Concept- τ (with $k = 2$) significantly outperforms the temperature-based approaches.

base. To verify the reliability of this automated validation, we manually inspected a random sample of 200 images. The estimated error rate of the validation procedure was approximately 2

C.5 Decoupling Model and Algorithmic Effects

As discussed above, GPT models do not expose or guarantee a standard implementation of temperature. We therefore use Llama-3-8B-Instruct for the temperature-based baselines to ensure reliable temperature control. To decouple model and algorithmic effects, we additionally compare Concept- τ with the temperature-based baselines using the same open-source LLM, Qwen2.5-7B-Instruct (Qwen et al., 2025). We use Qwen because the task required by Concept- τ proved too challenging for the Llama models.

Generating the intermediate-surprise candidate pool required for $k = 3$ remained challenging even for Qwen. We therefore compare Token- τ and Structure- τ against a simplified variant of Concept- τ with $k = 2$. The corresponding prompt is provided in the released code. As shown in Table 5, Concept- τ significantly outperforms both Token- τ and Structure- τ under the same underlying model, indicating that its advantage stems from the algorithm rather than the choice of LLM.

D Human Study Details

For the user study we collect human annotations through Prolific, recruiting 50 participants from the US and the UK with an approval rate of over 98% over at least 500 submissions. Age and gender details are reported in Figure 12. We compensate participants at a rate of 1.2 GBP (12.4 GBP per hour). No PII was collected beyond Prolific’s demographics; annotators were informed of that through the consent form, approved by our institute.

Annotators are asked to rank images across four creativity levels, from 1 (*Low*) to 4 (*Extreme*). Sub-

jects had to successfully complete a screening task to ensure they understood the instructions. Then, we sampled a random description for each of the four methods (order of methods was randomized), showing the participant all 12 generated images for this method. In total, we collect 2400 annotations. The main stages of the process are demonstrated in Figure 11.

E Concept- τ Extensions

E.1 Concept- τ as a distance metric

Here we detail how to invert our Concept- τ procedure for obtaining a distance between a textual image description d and its generated image $M_G(d')$.

Given d and $M_G(d')$, we infer the creativity value c by reversing our original process through the following steps:

1. Extract the structured representation G from d .
2. Fill in the novelty concepts in G using information derived from the image $I(d)$. For example, if a novelty concept in the updated graph G is *fur appearance*, and the image depicts a cat whose fur is gray, the concept can be completed as “Gray fur”.
3. Use an LLM to classify each novelty concept into a surprise level in the range $(1, k)$.
4. Use the inverse of *Assign* to obtain $[c_l, c_r]$ (an interval or c values).

The resulting interval acts as a conceptual surprise based distance metric between the prompt and the generated image, quantifying the degree of human-perceived surprise introduced during the generation process.

E.2 Concept- τ for Other Domains

We present an example of how the Concept- τ framework can be applied to other domains, specifically recipe generation. The overall process remains consistent with that used for text-to-image (T2I) generation, with the primary differences being the use of in-context examples tailored to the culinary domain and a domain-specific definition of *utility*.

In our adaptation, we assume that d is a general dish request with missing information (e.g., “sandwich”). In this context we define utility to have the following properties:

1. **Ingredients must be edible.** All components must be safe and recognized as food.

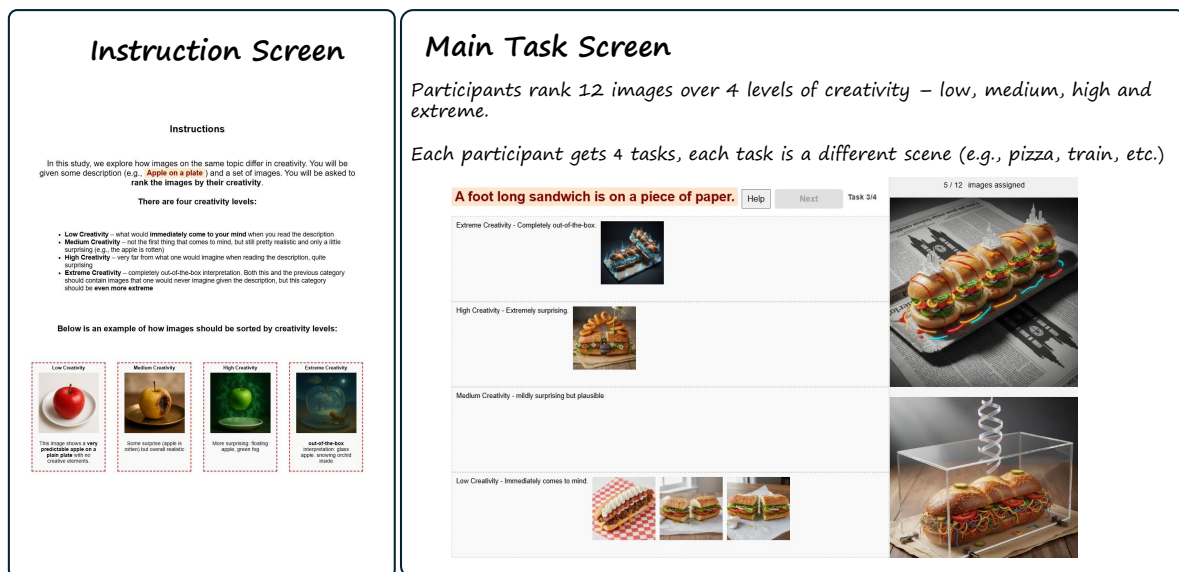


Figure 11: Left: Instruction screen, Right: Main task screen (in this example the participant sorts images from Concept- τ “Sandwich” scene.)

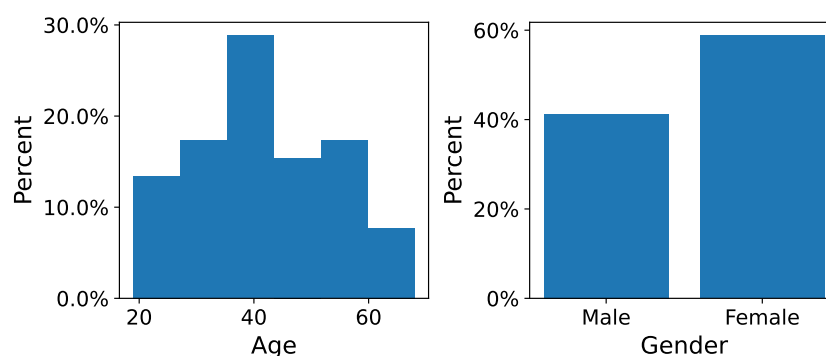


Figure 12: Age and gender distribution of participants in the user study.

- Ingredients must be real.** Recipes should remain grounded in reality – no fictional or impossible items.
- The preparation procedure must be feasible.** Instructions must describe steps that a cook could plausibly follow in a real kitchen.
- The resulting dish must taste good.**

To construct G in a manner that adheres to the new utility constraints, we first note that recipes naturally consist of two components: *ingredients* and *instructions*.

Ingredients. This component offers the greatest potential for creativity. Ingredient choice, substitution, and combination provide opportunities for innovation.

Instructions. In contrast, instructions are highly technical. They demand precision, reproducibility, and feasibility. While small creative variations are

possible (e.g., in presentation techniques or preparation order), many “creative” deviations risk being

- unnoticeable in the finished dish (e.g., “Wash the tomatoes while humming the Swedish anthem.”),
- impractical to execute, (e.g., “stir the sauce for six days.”)
- or harmful to the quality of the final dish (e.g., “Toss the bread out the window for extra texture.”).

Therefore, in our design of G , all unspecified ingredient-based concepts are treated as *novelty concepts*, while instructions are later obtained deterministically.

When constructing G , we follow the same procedure as in the image-generation adaptation and identify all unspecified or missing ingredients. For example, given the request “sandwich,” the missing ingredients include the type of bread, cheese, sauce,

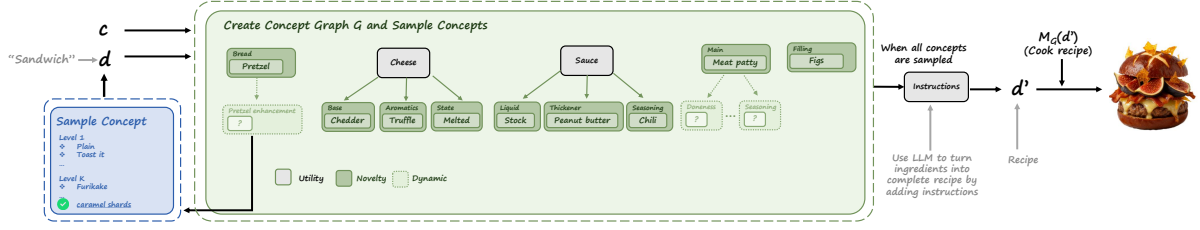


Figure 13: Concept-Level Temperature framework for recipe generation. We construct a concept graph G from the information missing in d , (§3.1) and assign each concept a creativity level (§3.2). For each concept, we generate candidate values across k creativity levels, conditioned on the iteratively updated d . We then select a sample based on the predefined creativity level (§3.3) and update d . After processing all novelty (ingredient) concepts we use an LLM to generate coherent instructions how to use them (utility). We now get the augmented description d' - a detailed recipe which adheres to d and is at c creativity level.

filling, and main component. If we further decompose the concept of “sauce,” its missing ingredients (i.e., its constituent elements) are the liquid, thickener, and seasoning.

We then apply the standard procedure of using *Assign* and generating a pool of candidates for all *novelty concepts* (ingredient concepts). By adapting the in-context examples to the culinary domain, we prevent the LLM from generating unrealistic or inedible ingredients (e.g., Lego bread). Once the ingredient set is determined, we generate the *instructions* deterministically with an LLM, conditioned on the selected ingredients. This provides us with a space to explore creativity while maintaining the utility of practical instructions. See Figure 13 for a diagram of the suggested procedure and how G could look in this domain. Full prompts are available in the code base.

F Correctness of Assign Function

Notation 1. We define the following notations to be used in the theorems of this section:

1. We recall $\text{Assign}(n, k, c)$ accepts as input c the creativity parameter, n the number of concepts and k the number of surprise levels. It returns an allocation vector which we denote $\text{Assign}(n, k, c) = (l_k, l_{k-1}, \dots, l_1)$ where l_i denotes the number of concepts allocated to level i .
2. We denote n_i the number of concepts that is provided to *Assign* as input at step i .
3. We denote c_i the rescaled creativity parameter c provided to *Assign* as input at step i .
4. $[c_{i-\text{left}}, c_{i-\text{right}}] \leftarrow \{c' \mid \lfloor h_i(c') \cdot n_i \rfloor = l_i\}$ the continuous interval which *Assign* computes at step i in order to rescale c_i , where h_i is the helper function at step i .
5. When we distinguish between several creativ-

ity parameter values (e.g., c_1, c_2) we append an identifier in superscript to all related notations. For instance for c_i we write $c_i^{(1)}, c_i^{(2)}$, for l_i we write $l_i^{(1)}, l_i^{(2)}$ and so on.

Lemma 1. Let c_1, c_2 be two creativity parameters such that $c_1 \geq c_2$

For any $0 < i < k$, if:

$$(l_k^{(1)}, l_{k-1}^{(1)}, \dots, l_{i+1}^{(1)}) = (l_k^{(2)}, l_{k-1}^{(2)}, \dots, l_{i+1}^{(2)}),$$

Then

$$c_i^{(1)} \geq c_i^{(2)}$$

Proof. We prove the claim by backward induction on i .

Base case ($i + 1 = k$). If $(l_k^{(1)}) = (l_k^{(2)})$, since $n = n_k^{(1)} = n_k^{(2)}$ then the intervals computed by *Assign* are identical:

$$[c_{k-\text{left}}^{(1)}, c_{k-\text{right}}^{(1)}] = [c_{k-\text{left}}^{(2)}, c_{k-\text{right}}^{(2)}]$$

We recall the next step in *Assign* is to rescale c before submitting it as input to the next step $k - 1$. To recall the rescaling is as such:

$$c_{k-1}^{(1)} = \frac{c_k^{(1)} - c_{k-\text{left}}^{(1)}}{c_{k-\text{right}}^{(1)} - c_{k-\text{left}}^{(1)}},$$

$$c_{k-1}^{(2)} = \frac{c_k^{(2)} - c_{k-\text{left}}^{(1)}}{c_{k-\text{right}}^{(2)} - c_{k-\text{left}}^{(2)}},$$

Since:

$$c_{k-\text{left}}^{(1)} = c_{k-\text{left}}^{(2)}, c_{k-\text{right}}^{(1)} = c_{k-\text{right}}^{(2)}$$

we get:

$$c_{k-1}^{(1)} \geq c_{k-1}^{(2)}$$

Induction step. Assume that for some $i + 1 \leq k$, if

$$(l_k^{(1)}, \dots, l_{i+1}^{(1)}) = (l_k^{(2)}, \dots, l_{i+1}^{(2)}),$$

Then

$$c_i^{(1)} \geq c_i^{(2)}$$

We will now prove the claim for i . We therefore assume:

$$(l_k^{(1)}, \dots, l_i^{(1)}) = (l_k^{(2)}, \dots, l_i^{(2)}),$$

Since $l_i^{(1)} = l_i^{(2)}$ we get:

$$c_{i-left}^{(1)} = c_{i-left}^{(2)}, c_{i-right}^{(1)} = c_{i-right}^{(2)}$$

We recall, $c_{i-1}^{(1)}, c_{i-1}^{(2)}$ are computed at step i as such:

$$c_{i-1}^{(1)} = \frac{c_i^{(1)} - c_{i-left}^{(1)}}{c_{i-right}^{(1)} - c_{i-left}^{(1)}}, c_{i-1}^{(2)} = \frac{c_i^{(2)} - c_{i-left}^{(2)}}{c_{i-right}^{(2)} - c_{i-left}^{(2)}}, \text{ then } n_{i-1}^{(1)} = n_{i-1}^{(2)}.$$

Since by the assumption $c_i^{(1)} \geq c_i^{(2)}$ we get

$$c_{i-1}^{(1)} \geq c_{i-1}^{(2)}$$

□

Theorem 1. For any $c_1, c_2 \in [0, 1]$ such that $c_1 > c_2$:

$$\text{Assign}(n, k, c_1) = (l_k^{(1)}, \dots, l_1^{(1)}) \succeq_L (l_k^{(2)}, \dots, l_1^{(2)}) \\ = \text{Assign}(n, k, c_2),$$

where $\succ_L$ denotes the lexicographic order.

Proof. Let $h_i : [0, 1] \rightarrow [0, 1]$ be the nondecreasing assign helper function for level i . For any $c_1, c_2 \in [0, 1]$ with $c_1 > c_2$, define for each index i :

$$l_i^{(1)} = \lfloor h_i(c_i^{(1)}) \cdot n_i^{(1)} \rfloor, \quad l_i^{(2)} = \lfloor h_i(c_i^{(2)}) \cdot n_i^{(2)} \rfloor,$$

We now prove that the claim holds for $i = 1$ by backward induction and show that for every i :

$$(l_k^{(1)}, \dots, l_i^{(1)}) \succeq_L (l_k^{(2)}, \dots, l_i^{(2)}),$$

Base case ($i = k$). Since h_k is nondecreasing and $c_k^{(1)} > c_k^{(2)}$ ($c_1 = c_k^{(1)}, c_2 = c_k^{(2)}$), we have $h_k(c_k^{(1)}) \geq h_k(c_k^{(2)})$. Since $n = n_k^{(1)} = n_k^{(2)}$ we get:

$$l_k^{(1)} = \lfloor h_k(c_k^{(1)}) \cdot n_k^{(1)} \rfloor \geq \lfloor h_k(c_k^{(2)}) \cdot n_k^{(2)} \rfloor = l_k^{(2)}.$$

Hence, $(l_k^{(1)}) \succeq_L (l_k^{(2)})$.

Induction step. Assume that for some $1 < i \leq k$,

$$(l_k^{(1)}, \dots, l_i^{(1)}) \succeq_L (l_k^{(2)}, \dots, l_i^{(2)}).$$

We now consider index $i - 1$.

Case 1: The tails are strictly ordered. If

$$(l_k^{(1)}, \dots, l_i^{(1)}) \succ_L (l_k^{(2)}, \dots, l_i^{(2)}),$$

then the same ordering immediately holds for the full sequences:

$$(l_k^{(1)}, \dots, l_{i-1}^{(1)}) \succeq_L (l_k^{(2)}, \dots, l_{i-1}^{(2)}).$$

Case 2: The tails are equal and the current step is $i > 2$. If

$$(l_k^{(1)}, \dots, l_i^{(1)}) =_L (l_k^{(2)}, \dots, l_i^{(2)}),$$

In addition, by Lemma 1 we get that $c_{i-1}^{(1)} \geq c_{i-1}^{(2)}$. Therefore, since h_{i-1} is nondecreasing we get that $l_{i-1}^{(1)} \geq l_{i-1}^{(2)}$, and consequently

$$(l_k^{(1)}, \dots, l_i^{(1)}) \succeq_L (l_k^{(2)}, \dots, l_i^{(2)}).$$

Case 3: The tails are equal. and the current step is $i = 2$. If

$$(l_k^{(1)}, \dots, l_2^{(1)}) =_L (l_k^{(2)}, \dots, l_2^{(2)}),$$

Then by the construction of Assign $l_1^{(1)} = l_1^{(2)} = n - \sum_{t=k}^2 l_t^{(1)}$, therefore we get:

$$(l_k^{(1)}, \dots, l_1^{(1)}) =_L (l_k^{(2)}, \dots, l_1^{(2)}),$$

By induction, the property holds for all $i \in \{1, \dots, k\}$.

Hence for $i = 1$,

$$\text{Assign}(n, k, c_1) = \\ (l_k^{(1)}, \dots, l_1^{(1)}) \succeq_L (l_k^{(2)}, \dots, l_1^{(2)}) \\ = \text{Assign}(n, k, c_2)$$

which proves the theorem. □

Theorem 2. Let n be a positive integer. For any allocation $(l_k, l_{k-1}, \dots, l_1)$ such that $\sum_{i=1}^k l_i = n$, there exists a constant $c \in [0, 1]$ such that $\text{Assign}(n, k, c) = (l_k, l_{k-1}, \dots, l_1)$

Proof. For each step i , we show the following:

1. There exists a point $c \in [0, 1]$ s.t. $\text{Assign}(n, k, c)_{[k:i]} = (l_k, l_{k-1}, \dots, l_i)$
2. For every point $c \in [0, 1]$ for which $\text{Assign}(n, k, c)_{[k:i]} = (l_k, l_{k-1}, \dots, l_i)$, for every point $x_i \in I_i$ (the interval computed by Assign at step i) there is a corresponding point at step k denoted $s_k(x_i) \in [0, 1]$ s.t:

$$\text{Assign}(n, k, s_k(x_i))_{[k:i]} = (l_k, l_{k-1}, \dots, l_i)$$

Base case ($i = k$).

Since $h_k : [0, 1] \rightarrow [0, 1]$ is continuous, surjective and non-decreasing, the function

$$x \mapsto h_k(x) \cdot n$$

is also continuous, surjective and non-decreasing on $[0, 1]$, with range $[0, n]$. Therefore, there exists a point $x'_k \in [0, 1]$ s.t:

$$\lfloor h_k(x'_k) \cdot n \rfloor = l_k$$

and therefore:

$$\text{Assign}(n, k, x'_k)_{[k]} = (l_k)$$

Let there be a point c for which: $\text{Assign}(n, k, c)_{[k]} = (l_k)$. By definition, every point $x_k \in I_k$ (the computed interval at step k) obtains:

$$\lfloor h_k(x_k) \cdot n \rfloor = l_k$$

Since $I_k \subseteq [0, 1]$, then trivially for every point $x_k \in I_k$ there is a point $s_k(x_k) = x_k$ for which $\text{Assign}(n, k, s_k(x_k))_{[k]} = (l_k)$ (the same point). We therefore show claim 2.

Induction step

We assume that:

1. There exists a point $c \in [0, 1]$ s.t. $\text{Assign}(n, k, c)_{[k:i]} = (l_k, l_{k-1}, \dots, l_i)$
2. For every point $c \in [0, 1]$ for which $\text{Assign}(n, k, c)_{[k:i]} = (l_k, l_{k-1}, \dots, l_i)$, for every point $x_i \in I_i$ (the interval computed by Assign at step i) there is a corresponding point at step k denoted $s_k(x_i) \in [0, 1]$ s.t:

$$\text{Assign}(n, k, s_k(x_i))_{[k:i]} = (l_k, l_{k-1}, \dots, l_i)$$

We now examine index $i - 1$.

Let c be a point s.t. $\text{Assign}(n, k, c)_{[k:i]} = (l_k, l_{k-1}, \dots, l_i)$

At step i of $\text{Assign}(n, k, c)$, I_i is rescaled into a continuous range of values $[0, 1]$ which is the domain of the input to Assign at step $i - 1$.

Since $x \mapsto h_{i-1}(x) \cdot n$ is continuous, non-decreasing and with range $[0, n_i]$, there exists $x'_{i-1} \in [0, 1]$ s.t

$$\lfloor h_{i-1}(x'_{i-1}) \cdot n_{i-1} \rfloor = l_{i-1}$$

By construction, every point $x_{i-1} \in [0, 1]$ in step $i - 1$ maps to a specific point in I_i , denoted $s_i(x_{i-1})$.

Specifically, $s_i(x'_{i-1}) \in I_i$

By our assumption (2), and since $s_i(x'_{i-1}) \in I_i$, there exists a corresponding point $s_k(s_i(x'_{i-1}))$ such that

$$\text{Assign}(n, k, s_k(s_i(x'_{i-1})))_{[k:i]} = (l_k, \dots, l_i)$$

Therefore, there exists a point $s_k(s_i(x'_{i-1})) \in [0, 1]$ s.t

$$\begin{aligned} &\text{Assign}(n, k, s_k(s_i(x'_{i-1})))_{[k:i-1]} \\ &= (l_k, \dots, l_i, l_{i-1}) \text{ (Claim 1)} \end{aligned}$$

We now show the correctness of the second second claim. Let there be a point $c \in [0, 1]$ for which

$$\text{Assign}(n, k, c)_{[k:i-1]} = (l_k, l_{k-1}, \dots, l_{i-1})$$

By definition, every point $x_{i-1} \in I_{i-1}$ obtains $\lfloor h_{i-1}(x_{i-1}) \rfloor = l_{i-1}$.

Since every such point x_{i-1} maps to a specific point in I_i : $s_i(x_{i-1})$ and using our assumption (2), there exists a point $s_k(s_i(x'_{i-1})) \in [0, 1]$ s.t:

$$\begin{aligned} &\text{Assign}(n, k, s_k(s_i(x'_{i-1})))_{[k:i-1]} \\ &= (l_k, l_{k-1}, \dots, l_{i-1}) \end{aligned}$$

□