Large Language Model Meets Constraint Propagation

Alexandre Bonlarron , **Florian Régin** , **Elisabetta De Maria** and **Jean-Charles Régin**

Université Côte d'Azur, CNRS, I3S, France

alexandre.bonlarron@gmail.com, florian.regin06@gmail.com, elisabetta.de-maria@univ-cotedazur.fr,
jcregin@gmail.com

Abstract

Large Language Models (LLMs) excel at generating fluent text but struggle to enforce external constraints because they generate tokens sequentially without explicit control mechanisms. GenCP addresses this limitation by combining LLM predictions with Constraint Programming (CP) reasoning, formulating text generation as a Constraint Satisfaction Problem (CSP). In this paper, we improve GenCP by integrating Masked Language Models (MLMs) for domain generation, which allows bidirectional constraint propagation that leverages both past and future tokens. This integration bridges the gap between token-level prediction and structured constraint enforcement, leading to more reliable and constraint-aware text generation. Our evaluation on COLLIE benchmarks demonstrates that incorporating domain preview via MLM calls significantly improves GenCP's performance. Although this approach incurs additional MLM calls and, in some cases, increased backtracking, the overall effect is a more efficient use of LLM inferences and an enhanced ability to generate feasible and meaningful solutions, particularly in tasks with strict content constraints.

1 Introduction

The landscape of Large Language Models (LLMs) is evolving rapidly, with new features and capabilities emerging near-weekly. Despite their impressive fluency and versatility, current LLMs often struggle to adhere to specific rules or regulations during text generation [Yao *et al.*, 2024]. This limitation arises because LLMs are primarily designed as text predictors without built-in mechanisms to enforce constraints. This lack of guarantee has become an increasingly important, almost fundamental problem around the confident and safe usage of LLMs [Cardei *et al.*, 2025; Geh *et al.*, 2025]. As a result, recent Natural Language Processing (NLP) trends have explored augmenting LLMs with external control mechanisms. Examples include Retrieval-Augmented Generation (RAG) (e.g., integrating search engine results), grounding techniques (e.g., injecting relevant documents into the context window), external knowledge bases, and reasoning frameworks such as chain-of-thought [Wei *et al.*, 2022] and tree-of-thought [Yao *et al.*, 2023]. However, these methods often provide only partial solutions to the broader problem of *constrained text generation*—the task of generating text that meets mandatory requirements.

Structured methods from combinatorial optimization offer a promising alternative. For instance, heuristically-guided text generation utilizes techniques like Beam Search (BS) [Post and Vilar, 2018; Lu *et al.*, 2022] by exploring a subset of the search space. In contrast, Constraint Programming (CP) [Bonlarron *et al.*, 2023; Bonlarron and Régin, 2024a] has been used to formalizes the constrained generation task as a Constraint Satisfaction Problem (CSP), where:

- Variables correspond to text words,
- Domains represent the allowed vocabulary,
- and Constraints define the admissibility of word sequences.

Once the feasible solution set is found through an exhaustive search, the LLM performs a curation phase to rank these candidates based on criteria such as fluency and coherence. While effective in scenarios dominated by strong constraints, these CP-based approaches require multiple, complex processing steps and are parameter-sensitive. They suffer from the combinatorial explosion when constraints are weaker. This limitation arises from the loosely coupled two-step process, which separates CP and LLM, thereby hindering the direct integration of the LLM into the solving procedure. Ultimately, an LLM surrogate is employed to partially mitigate the combinatorial explosion, albeit while remaining sensitive to the cost of LLM inference in the curation phase (>>100 ms) [Bonlarron and Régin, 2024b].

A recent and up-and-coming line of research leverages LLMs to guide CP-based text generation: (GenCP) [Régin *et al.*, 2024], achieving 100% constraint satisfaction on recent benchmarks [Yao *et al.*, 2024] (sentence generation tasks). In this strongly coupled approach, the LLM first generates candidate tokens (using top-p or top-k sampling [Holtzman *et al.*, 2020]), which are then refined by a CP solver through a method that sequentially instantiates variables while maintaining local consistency in a backtracking search.

Unfortunately, the GenCP method is based on the modeling approach proposed by Bonlarron et al., which assumes a one-to-one correspondence between words and variables.
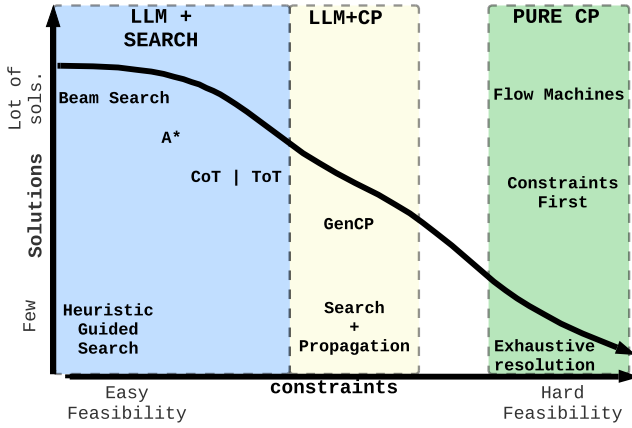
Figure 1: This diagram positions different methods along the x-axis based on feasibility difficulty for constrained text generation, from 'Easy Feasibility' to 'Hard Feasibility'. Methods such as Beam Search [Post and Vilar, 2018], A* [Lu *et al.*, 2022], and Tree of Thoughts (ToT) [Yao *et al.*, 2024], GenCP [Régin *et al.*, 2024], Pachet's ERC Flow Machines (Markov +X) [Pachet and Roy, 2011], and Constraints First [Bonlarron *et al.*, 2023] illustrate this progression.

However, this assumption does not hold in general for LLMs, as they process text at the token level, where tokens may correspond to entire words, subwords, or even individual characters, depending on the tokenizer [Geh *et al.*, 2024].

This modeling choice influenced how GenCP managed variables, ultimately limiting its ability to generate high-quality and feasible outputs. The original GenCP paper acknowledged the need for a token-based approach rather than a word-based one. In this work, we extend GenCP by introducing meta-variables, enabling it to manage tokens more effectively and improve both constraint satisfaction and text generation quality.

However, relying solely on autoregressive LLMs introduces a significant limitation: these models generate text left to right and do not give information about future tokens. In principle, if a filtering algorithm (propagator) can detect inconsistencies faster than the search procedure would otherwise uncover them, then early propagation of the constraints is highly beneficial. In other words, effective propagation reduces the search space by pruning inconsistent candidates before the solver suffers the cost of further search exploration. Since standard CP solvers rely on propagation, where constraints between distant variables help refine candidate domains, this unidirectional generation restricts the solver's ability to forecast future values (e.g., predicting the domain $D(X_8)$ solely on assignments X_1 to X_5). Consequently, the CP propagation mechanism becomes *starved* of necessary domain values, leading to redundant computations and search *thrashing* during backtracking.

To overcome this limitation, we propose leveraging Masked Language Models (MLMs), such as BERT [Devlin *et al.*, 2019], to enhance constraint propagation. Unlike autoregressive models, MLMs process entire sequences bidirectionally by predicting masked words based on both preceding and following context. For example, given the prompt "The city of London is the [MASK] of the United Kingdom," BERT correctly predicts *capital*. This bidirectional capability naturally aligns with CP solvers [Bessiere, 2006], as it enables constraints to be propagated across the entire sequence rather than solely in a left-to-right manner. In practice, if tokens X_1 through X_5 are assigned, an MLM can generate relevant candidate tokens for X_7, thereby refining the assignment decision for X_6 and reducing the likelihood of dead-ends and unproductive search efforts. Ultimately, while the backtracking search is guided by an autoregressive LLM in a left-to-right fashion, any constraint that involves long-range dependencies prompts the solver to leverage an MLM to enrich its candidate domains, thereby enabling improved filtering and more informed assignment decisions.

This article makes the following contributions:

- it introduces a novel integration of LLMs and CP that leverages bidirectional MLMs to enhance constraint propagation in text generation.

- it improves domain filtering by incorporating both past and future token assignments, thereby mitigating the limitations of autoregressive generation alone.

This article is organized as follows: Section 2 briefly introduces the necessary background on Constraint Programming and NLP techniques. Section 3 details our proposed approach. Section 4 presents experimental results, and Section 5 discusses limitations, open challenges, and future research directions. Section 6 concludes the paper.

2 Background

This section briefly reviews the key concepts and related work underpinning our approach, focusing on constrained text generation, LLM decoding, and GenCP.

2.1 Related Work in Constrained Generation

Early work in CP demonstrated its potential in creative domains such as text and music generation. For instance, the ERC Flow Machines led by François Pachet explored CP for style modeling [Pachet and Roy, 2011], lyric composition [Barbieri *et al.*, 2012], and many creative applications [Papadopoulos *et al.*, 2014; Papadopoulos *et al.*, 2015; Papadopoulos *et al.*, 2016]. More recent efforts have applied CP to standardized sentence generation [Bonlarron *et al.*, 2023; Bonlarron and Régin, 2024b; Bonlarron and Régin, 2024a] and even advanced musical composition [Sprockeels and Van Roy, 2024]. These contributions underscore CP's versatility and relevance to express complex requirements through constraints or exhaustively explore search space in generative setup.

2.2 Decoding Strategies in LLMs

Autoregressive LLMs generate text by predicting tokens sequentially. Common decoding strategies include:

- **Greedy Decoding:** Selects the most probable token at each time step but may miss globally optimal sequences.

- **Stochastic Sampling:** Top-k and top-p (nucleus) sampling [Holtzman *et al.*, 2020] introduce diversity by sampling from a subset of high-probability tokens.

- **Beam Search:** Maintains the k most likely sequences at each step [Hokamp and Liu, 2017; Post and Vilar, 2018]. Although beam search can improve overall results, it relies on local token probabilities and is tailormade to manage local constraints such as keyword constraints. It was extended to deal with constraints as logic predicates in conjunctive normal form [Lu *et al.*, 2021].

- **Beam Search + A*** Beam search augmented with an A* look-ahead [Lu *et al.*, 2022] is employed to enhance overall sequence probabilities in both unconstrained and constrained scenarios. This approach retains the top k candidate partial solutions that are closest to satisfying the constraints. In details, constraint violations are penalized according to their distance to the satisfaction and integrated into the objective. As a matter of fact, it steers the generation towards satisfaction.

While these methods are effective for generating fluent text, they do not inherently guarantee that generated sequences satisfy external constraints, motivating the integration of CP techniques. Figure 1 provides an overview of the feasibility challenges associated with different approaches.

2.3 Masked Language Models (MLMs)

Masked Language Models (MLMs), such as BERT [Devlin *et al.*, 2019], predict missing tokens in a text by considering both the left and right context. This bidirectional nature allows MLMs to capture deeper contextual relationships compared to autoregressive models. Typically, a portion of the input tokens is replaced by a special [MASK] token, and the model is trained to recover these tokens based on the surrounding context while maximizing:

$$\sum_{w_i \in M} \log P(w_i \mid \tilde{W}), \qquad (1)$$

where M is the set of masked positions in the modified sequence $\tilde{W}$. This capability is central to our approach, as it enables the propagation of constraints over the entire sequence rather than incrementally, thereby overcoming a key limitation of autoregressive generation .

2.4 Constraint Satisfaction Problems (CSPs) and GenCP

A Constraint Satisfaction Problem (CSP) is defined as a triplet $\langle X, D, C \rangle$, where:

- $\mathcal{X} = \{X_1, X_2, \dots, X_n\}$ is the set of variables,

- $\mathcal{D} = \{D(X_1), D(X_2), \dots, D(X_n)\}$ is the set of domains (each $D(X_i)$ is the set of values the variable X_i can take), and

- $\mathcal{C} = \{C_1, C_2, \dots, C_m\}$ is the set of constraints that specify allowable combinations of values.

A solution to a CSP is an assignment of values to all variables such that every constraint is satisfied.

Traditional CP approaches assume a fully defined CSP prior to the search. However, in constrained text generation, the structure of the problem is built incrementally as the text is generated. To address this, **GenCP** [Régin and De Maria, 2023; Régin *et al.*, 2024; Régin, 2024] introduces a dynamic, evolving CSP that is constructed on-the-fly. GenCP is governed by three core procedures:

- GenV: Dynamically introduces new variables.

- GenD: Defines the domain of each new variable using LLM predictions.

- GenC: Imposes constraints over variables to enforce desired properties during generation.

In summary, GenCP starts with no variables, domains, or constraints and constructs them incrementally during the search for solutions. Whenever a domain is defined via GenD, classical CP techniques, such as constraint propagation, search, backtracking, and domain pruning, are employed to ensure that the generated text adheres to the specified constraints. Figure 2 illustrates the entire process.

3 Future Variable Domain Generation with MLM

In this section, we describe our proposed method for integrating bidirectional Masked Language Models (MLMs) into the dynamic CP framework GenCP for constrained text generation. Our approach enhances constraint propagation by leveraging MLM-based domain generation and is organized into three main components: variable representation (GenV), domain generation (GenD), and constraint integration (GenC).

3.1 **GenV**: Variable Representation and Formatting

A key challenge in interfacing CP with language models is reconciling the granularity of decision variables with the token-based output of LLMs. While our formulation naturally models decision variables at the word level, LLMs operate on subword tokens. To bridge this gap, we introduce *meta-variables* X_i, where each meta-variable can comprise one or more decision variables $X_{i_1}, X_{i_2}, \dots$.

For example, generating the word "Using" might involve:

- X_{i_1}: "Us"

- X_{i_2}: "ing"

For instance, a current CSP assignment may be encoded as:

```
"Us;ing; a; transform;er;"
```

Here, spaces separate meta-variables, while semicolons distinguish individual decision variables. This flexible representation accommodates both token-level and word-level structure, and it can be extended to model larger textual units if needed.

Variable Formatting

As generation proceeds, most variables are already assigned. To generate the next variable, the MLM is provided with a
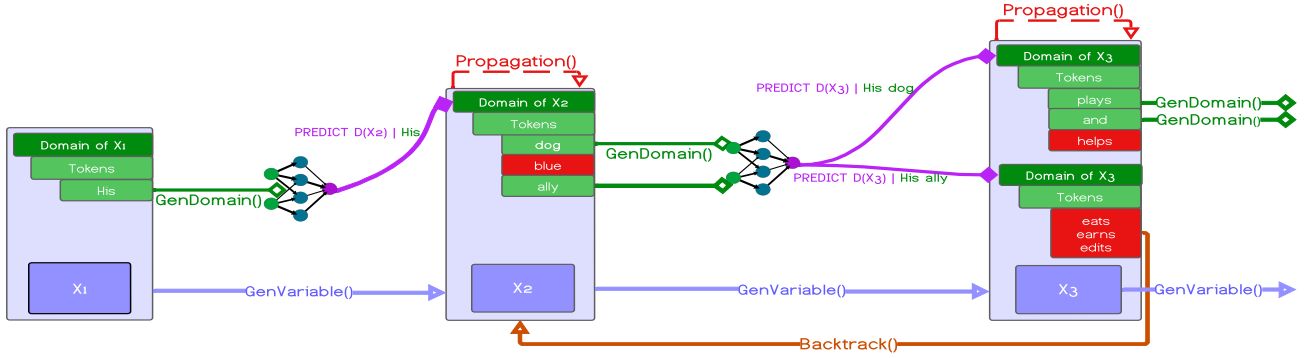
Figure 2: Illustration of GenCP in a toy example where the task is to generate a passage without the letter "e". The autoregressive LLM incrementally predicts candidate domains for variables, while GenCP enforces constraints dynamically. If no valid domain remains for a variable, backtracking is triggered.

prompt consisting of the current assignments followed by one or more [MASK] tokens. For example, if:

$$X_1 = \text{"The"}, \quad X_2 = \text{"little"}, \quad X_3 = \text{"boy"},$$

The prompt becomes:

`"The little boy [MASK] [MASK] [MASK]..."`

The MLM then predicts candidate tokens for the masked positions, forming the candidate domains $D(X_4)$ and $D(X_5)$.

3.2 `GenD`: Domain Generation via MLM

The `GenD` procedure is responsible for instantiating candidate domains for unassigned variables. In our framework, if a variable X_i is unassigned and masked, the system queries an MLM rather than an autoregressive LLM to infer its domain.

MLM Domain Preview
Given a partial assignment of $k-1$ variables and a look-ahead depth d, the MLM is used to generate candidate domains for the next d variables:

$$\{X_k, X_{k+1}, ..., X_{k+d}\}.$$

After generating these candidate domains, relevant constraints are applied to the new variables. If any domain becomes empty, backtracking is triggered; otherwise, the next variable X_k is generated. Once the domain $D(X_k)$ is obtained (potentially via an autoregressive LLM), it is filtered by propagators associated with the current set of constraints $\mathcal{C}$. Consequently, the assignment $X_k = w$, where $w \in D(X_k)$, is determined based on the remaining consistent values within all variable domains:

$$\bigsqcup_{i=1}^{k+d} D(X_i).$$

This process can be iteratively applied to the next variable, X_{k+1}, considering now k assignments. Hence, it iteratively builds the complete text while ensuring that each assignment is consistent with past and anticipated future domains concerning the constraints.

In detail, following the generation of the domain for X_k via an autoregressive LLM, the filtering of $D(X_k)$ is conducted using the propagators associated with the current set of constraints $\mathcal{C}$.

3.3 `GenC`: Constraint Integration and Propagation

After candidate domains are established, constraint propagation ensures that future assignments adhere to task-specific requirements. For example, a length constraint can be expressed as:

$$C : \sum_{i=1}^{n} \#Char(X_i) = T,$$

where T is the target total length and $\#Char(X_i)$ denotes the number of characters in X_i. Such format constraints are ubiquitous in constrained text generation tasks from poetry generation, summarization or text style transfer[Garbacea and Mei, 2022].

Constraint-Specific Propagation
Once $k - 1$ variables have been assigned, the solver prepares to assign the kth variable by constructing a prompt:

$$\{X_1 = w_1, \ldots, X_{k-1} = w_{k-1}, X_k = [M], X_{k+1} = [M]\}.$$

The MLM then predicts candidate values for X_{k+1}, forming the domain $D(X_{k+1})$. Constraint propagation refines $D(X_{k+1})$ using the current assignments and the constraint C. For instance, considering a length constraint, the solver computes bounds L and U such that:

$$L \leq \#Char(X_{k+1}) \leq U,$$

and adjusts the domains of current and future variables accordingly to ensure that the cumulative length satisfies constraint C.

Pathological Case. In the worst-case, consider a sentence that must contain exactly ten words, with only two variables left to assign and ten characters remaining to satisfy the constraint C. Standard knapsack filtering [Trick, 2003] cannot be applied directly because the domains of the last two variables are unknown. As a result, the search would explore all possible combinations in a brute-force, generate-and-test manner, iterating over $D(X_9) \times D(X_{10})$ until it finds that, given $D(X_{10})$, for instance it was (7+3).

MLMs help by eliminating inconsistent combinations early while prioritizing feasible ones. Instead of exhaustively exploring every token combination in the penultimate and final domains, the solver can precompute the Cartesian product of admissible token sums for these positions based on the preview gave by the MLM that satisfy the current sum constraint. Consequently, the solver considers only those tokens that are coherent with the constraint, enabling more fine assignments. This domain previewing reduces unnecessary exploration, and since the rankings of the combination preserve the likelihood ordering of the autoregressive LLM, the method does not interfere with the left-to-right generation process while integrating bidirectional information to improve constraint satisfaction and more efficiently reach feasible solutions.

3.4 Implementation Workflow

The complete workflow for integrating MLM-based domain generation into GenCP is as follows:

1. **Domain Initialization:** Each variable X_i is initialized as empty or masked.

2. **MLM Querying:** The MLM predicts candidate tokens for masked positions, forming the candidate domain $D(X_i)$.

3. **Constraint Propagation (filtering):** The CP solver refines $D(X_i)$ based on previous assignments and task-specific constraints.

4. **Decision Making (assignment):** A value is selected from $D(X_i)$ to extend the partial assignment while ensuring consistency.

This integration ensures that generated text satisfies both linguistic coherence and strict problem-specific constraints, addressing the limitations of purely autoregressive generation.

4 Result

4.1 Experimental Conditions

The experiments were performed on a laptop with Windows 10 Professional, 32 GB RAM, and Intel 16 CPU cores. The new GenCP approach is implemented in Java 17. GenCP restarts the search each time a solution is found. A domain generation call (GenD) asks for 50 tokens (top-k) .

Benchmarks

Our benchmark suite consists of several task can be found in Tab. 1.

These benchmarks are inspired by the COLLIE Benchmark [Yao *et al.*, 2024], which exposes the limitations of LLMs in handling such tasks alone. The task sent-1 is retained to exemplify a fundamental counting task, illustrating the advantage of MLM-based domain preview. The remaining tasks integrate keyword usage and counting within paragraph generation, a challenge not addressed in the original GenCP approach [Régin *et al.*, 2024].

Task	#Sent	#W p Sent	#Char per Sent + Constraints
sent-1	1		=82
para-2	2	10-15	=60
para-3	3	≥ 15	
para-4	2	14	72-74
para-5	3		Start: "Dragons", "Kindgoms", "Barbarians"
para-6	4		No occurrence of "the", "and", "of"

Table 1: One sentence generation task, (sent-1) and 5 paragraph task (para-2-6). : #Sent = number of sentences, #Words/#Sent = word count per sentence, #Char/#Sent = character count per sentence with if any specific constraints.

sent-1	d	#LLMCalls	#MLMCalls	#bks	#sols
vanilla	0	444	0	375	3
metavar	0	456	0	211	4
previewMLM	2	**359**	32	170	**10**
para-2					
metavar	0	446	0	136	3
previewMLM	2	**341**	40	193	**6**
para-3					
metavar	0	824	0	0	13
para-4					
metavar	0	466	0	447	0
previewMLM	2	**422**	12	258	**2**
para-5					
metavar	0	833	0	0	21
para-6					
metavar	0	430	0	34	3

Table 2: The table reports, for each task which GenCP version was used (e.g., `vanilla`, `metavar`, `previewMLM`, the depth d for domain preview, the number of autoregressive LLM calls (#LLMCalls), the number of MLM calls (#MLMCalls), the number of backtracks (#bks), and the number of solutions (#sols) for sent-1 and paragraph tasks (para-2 to para-6).

LLM and MLM Choice

In the original GenCP paper, the model used was LLaMa Q4 7B, a lightweight variant of LLaMa quantized to 4-bit integers. The quantization level was such that it was unclear whether any loss of "meaning" was due to the search process or simply to poor token predictions. In contrast, in this paper we employ the autoregressive LLM `babbage-002` provided by OpenAI. This mid-sized model is not as large as the latest GPT-4 and can be considered as a basic model, lacking the additional refinements like InstructGPT. Thus, it is LLM that was not fine-tuned to follows instructions.

For the MLM component, computational considerations were less of an issue given that MLMs typically have fewer than 1 billion parameters. We used BERT as our MLM. Although more recent and larger MLM models are available, our objective is to gain insight into future values rather than to perfectly model language in this framework; therefore, employing a lightweight model is reasonable.

For left-to-right generation, we used `babbage-002`. The temperature was set to 0.8, it is considered as an high enough

task:
generated sample...

sent-1:
His eyes sparkled as he stared into the void, awaiting the arrival of an immortal.

para-2:
With an arm raised in triumph, he raised himself once again.
The sun had shifted its position, casting a shadow on earth.

para-3:
The warrior had been on the battlefield for at least a dozen years,
fighting and dying with the forces of the Empire
against the growing power of the Orcs.
He was the last in a line of warriors that had fought on the front.
The orcish king was dead, his son and grandson had been killed in a ruthless battle.

para-4:
The warrior was an ancient warrior, known to many as The Sword of Justice.
His name is known to all, but he is known, mostly, throughout the lands.

para-5:
Dragons raised into the sky, one of which bore a striking resemblance
to the dragons of myth, were beginning to circle in the sky above.
Kingdoms trembled, while the mighty nations of the world sat abed,
awaiting the day's end. Barbarians with swords and axes drew together,
their bloodlust and greed being satisfied one final time.

para-6:
He stood watching, waiting, as an eerie silence filled his mind.
His soul was trapped in a timeless battle, between
desire, truth, love, hate, fear, anger, lust, greed, envy,
cowardice, pride, ambition, pride, lust, revenge, forgiveness, jealousy, rage,
lust, hatred, ignorance, stupidity, ignorance, stupidity, etc.
He was battling for his very soul. This was his life.

Table 3: One generated example with GenCP with `previewMLM` for the tasks 1,2 and 4. While, GenCP with `metavar` only for the tasks 3,5 and 6.

value for creative task (i.e., less generic output). For domain preview, we used `bert-base-cased`.

Pre-prompt: Directly prompting the LLM or the MLM with an empty variable assignments often results in dull and unengaging outputs. To address this, we introduced thematic grounding to inspire more vivid and compelling sentences than typical legal text or generic blogging. For instance, using the pre-prompt: *"Amidst the crimson glow of a setting sun, a lone warrior, clad in battle-worn silver, stood atop the ancient ruins, his blade gleaming with the promise of legend."* This pre-prompt was employed to guide the LLM towards generating related content, enhancing engagement without the primary intention of controlling semantics or themes. While the main focus of this article is not to manipulate thematically through pre-prompts while enforcing constraints, this technique demonstrates the potential to evoke heroic fantasy elements in the generated text.

Evaluation: Interacting with an LLM or an MLM often incurs high latency, typically on the order of hundreds of milliseconds. However, this is only a rough estimate since a wide range of hardware, from CPUs to GPUs, can run LLMs, and numerous optimization techniques (e.g., fast attention [Dao *et al.*, 2022], model quantization [Gholami *et al.*, 2021]) can significantly accelerate inference times. To quantify our results, we evaluate our approach in terms of LLM calls. And it offers a form of hardware independence. As shown in Table 2, we use the number of LLM calls to provide insight into the interaction between GenCP and the LLMs, and to enable comparisons across experiments conducted under different settings.

4.2 Result Analysis

Our experimental results, summarized in Table 2, provide insight into both the efficiency and quality of the different GenCP variants across the benchmark tasks. In the following, we analyze the performance and output quality of the approaches in detail.

Quality Analysis: Across all benchmarks in Table 1, both the `vanilla` and `metavar` approaches find a similar number of solutions. However, when a word is split into several tokens using meta-variables, the performance in terms of constraint satisfaction can sometimes decrease. This is due to the increased number of combinations that must be considered to assign a suitable value to a single meta-variable representing a word. Nevertheless, the improvement in text quality and diversity with meta-variables is substantial. Without meta-variables, GenCP effectively limits its search to a small subset of the LLM's vocabulary, typically very short and frequent tokens (e.g., "as," "they," "their," "a," "them"). In contrast, incorporating meta-variables expands the search space to cover a broader vocabulary, which, although it increases the search space, ultimately leads to richer and more diverse generated text. As the next subsection will details, in case where the constraint is not local but applied over the whole sequence of text, the use of the `PreviewMLM` to find more solutions in the same amount of times.

Performance Analysis: For the single-sentence generation task (sent-1), the `vanilla` approach required 444 LLM calls, encountered 375 backtracks, and generated 3 valid solutions. The `metavar` variant slightly increased the number of LLM calls to 456 while reducing backtracks to 211, yielding 4 solutions. In contrast, the `previewMLM` method, which leverages a domain preview (with depth $d = 2$) and incorporates an additional 32 MLM calls, significantly outperformed the other approaches by reducing LLM calls to 359 and backtracks to 170, while producing a notable 10 solutions. This clearly demonstrates that the domain preview strategy not only minimizes the reliance on expensive LLM calls but also improves the overall efficiency of the search.

A similar trend is observed in the paragraph-level tasks. In para-2, the `metavar` approach incurred 446 LLM calls and 136 backtracks to produce 3 solutions, whereas `previewMLM` reduced the LLM calls to 341 and, with 40 MLM calls and 193 backtracks, doubled the number of solutions to 6. For para-3, only the `metavar` variant was employed, requiring 824 LLM calls (with no backtracking) to yield 13 solutions. In para-4, the `metavar` method performed poorly with 466 LLM calls and a high backtracking cost of 447, failing to produce any valid solution; meanwhile, `previewMLM` achieved 422 LLM calls, 12 MLM calls, and 258 backtracks, resulting in 2 solutions. For para-5 and para-6, only the `metavar` method was used, with para-5 achieving 21 solutions from 833 LLM calls and para-6 yielding 3 solutions from 430 LLM calls and 34 backtracks.

These results highlight that integrating a domain preview (as in `previewMLM`) is particularly effective in reducing the number of LLM calls and reducing the search exploration, especially for tasks that impose strong constraints.

These observations suggest that while the non-preview approaches may be adequate for tasks with relatively weak constraints, the domain preview mechanism in previewMLM is particularly effective in more challenging scenarios. The additional guidance provided by the MLM preview seems to enable a more targeted search, thereby improving the feasibility capability of the approach to produces outputs under strict constraints.

Limitations: Since MLMs are trained to predict a limited fraction (typically about 15%) of masked tokens, they perform best when given rich contextual information. Consequently, it is important to note that, the depth d to define the preview of the domain is used only on the last two variables of any sentences. We tried further depth $d = 3$ and $d = 4$ and so on but unfortunately the number of solutions decrease for the same amount of running time while keeping the same quality. It seems the domain preview works best for the propagation of the sum over the last and before last variables to perform filtering and assignment.

5 Discussion

5.1 Thinking Fast and Slow AI

A fundamental challenge in constrained text generation is balancing intuitive language fluency with strict constraint satisfaction. This aligns with the dual-process theory described by Kahneman [Kahneman, 2011], which distinguishes between fast, intuitive reasoning (System 1) and slow, deliberate reasoning (System 2). Prior work [Booch *et al.*, 2021] has drawn parallels between this framework and AI systems, associating data-driven, heuristic-based approaches (e.g., LLMs) with System 1, while aligning structured reasoning methods (e.g., CP solvers) with System 2.

In our framework, the domain generation process implemented via GenD using LLMs or MLMs act as System 1, quickly proposing candidate tokens based on heuristic evaluation. In contrast, the CP component, responsible for filtering and assigning future values, operates as System 2, carefully enforcing constraints and guiding the overall generation process. Here, the CP solver maintains the primary control loop and invokes the heuristic domain generation only when needed, ensuring that the generated text remains both fluent and compliant with the desired constraints.

This integration effectively bridges the gap between intuitive language generation and rigorous constraint enforcement, harnessing the strengths of both fast, heuristic reasoning and slow, deliberate decision-making.

5.2 Meaningful Content Constraint

Our approach leverages the LLM's expertise in producing meaningful text to manage the domains of the CSP. In this framework, the LLM serves two roles. First, it implicitly defines a "meaningful content constraint" by generating an initial set of candidate values for each decision variable, in a similar way as Lazy Arc Consistency [Schiex *et al.*, 1996]. Second, it provides a good branching heuristic based on likelihood that guides the solver's decisions. In other words, assign the values (tokens) in the same ordering as traditional decoding strategies. As a result, when a the domain of a variable

is determined, the solver assigns values that not only satisfy the explicit constraints but also maintain high semantic quality, with backtracking employed to recover from inconsistent or low-likelihood assignments.

5.3 Backtracking with LLM
Strengths of Incorporating Backtracking
Backtracking, as a complete search strategy, is traditionally considered less efficient than incomplete methods; however, a backtracking search with good heuristic often brings with few backtracks a viable solution (e.g., for scheduling problems [Baptiste *et al.*, 2001]). This is the case for the experiments in task para-6 and sent-1. This limited backtracking can capture high-quality, constrained text that a greedy procedure might overlook [Régin *et al.*, 2024]. The inherent completeness of backtracking provides an attractive guarantee: even when free text generation may lead to harmful or incoherent outputs, backtracking may be a great tool to enforce those constraints robustly. Furthermore, employing a restart-based strategy after finding the first solution: as demonstrated in creative applications [Sprockeels and Van Roy, 2024], can yield both diversity and improved quality without sacrificing efficiency.

Potential Dangers and Mitigation Strategies
The complete nature of backtracking search can drive the LLM to explore low-likelihood regions of the search space. While this might increase the feasibility (higher satisfaction rates), it risks degrading overall text quality. In other words, a solution that meets all constraints may still be semantically or stylistically "suboptimal" if it resides in an unlikely region of the LLM's output distribution. This empirical observation underscores the importance of balancing constraint satisfaction with likelihood. Therefore, we advocate for two mitigation strategies: (1) initiating search restarts once a predefined number of acceptable solutions (e.g., k solutions) are found, and (2) continuously tracking the likelihood of the current assignment and backtracking when this likelihood falls below a set threshold. Such measures can help maintain the semantic quality of generated text while still benefiting from the robustness of backtracking.

6 Conclusion

This paper addresses the limitations of autoregressive LLMs in constrained text generation tasks, thanks to an enhanced GenCP that integrates MLMs for domain previewing. Our approach balances token-level predictions (metavar) and structural constraint enforcement, resulting in more reliable constraint management during generation. Through our empirical evaluations on COLLIE benchmarks, we demonstrated that incorporating an MLM-based domain preview (previewMLM) significantly improves GenCP's performance in tasks with strict content constraints in terms of feasibility. Even though, GenCP uses additional MLM calls, it optimizes the use of LLM inferences, leading to more feasible and meaningful constrained text outputs. By refining this interplay between CP and LLM decoding, we aim to broaden the potential for these systems to generate text when language fluency and constraint satisfaction are paramount.

Acknowledgments

References

[Baptiste *et al.*, 2001] Philippe Baptiste, Claude Le Pape, and Wim Nuijten. *Constraint-based scheduling: applying constraint programming to scheduling problems*, volume 39. Springer Science & Business Media, 2001.

[Barbieri *et al.*, 2012] Gabriele Barbieri, François Pachet, Pierre Roy, and Mirko Degli Esposti. Markov constraints for generating lyrics with style. In *Proceedings of the 20th European Conference on Artificial Intelligence*, ECAI'12, page 115–120, NLD, 2012. IOS Press.

[Bessiere, 2006] Christian Bessiere. Constraint propagation. In *Foundations of Artificial Intelligence*, volume 2, pages 29–83. Elsevier, 2006.

[Bonlarron and Régin, 2024a] Alexandre Bonlarron and Jean-Charles Régin. Intertwining cp and nlp: The generation of unreasonably constrained sentences. In Kate Larson, editor, *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pages 7600–7608. International Joint Conferences on Artificial Intelligence Organization, 8 2024. AI, Arts & Creativity.

[Bonlarron and Régin, 2024b] Alexandre Bonlarron and Jean-Charles Régin. Markov constraint as large language model surrogate. In Kate Larson, editor, *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pages 1844–1852. International Joint Conferences on Artificial Intelligence Organization, 8 2024. Main Track.

[Bonlarron *et al.*, 2023] Alexandre Bonlarron, Aurélie Calabrèse, Pierre Kornprobst, and Jean-Charles Régin. Constraints first: A new mdd-based model to generate sentences under constraints. In Edith Elkind, editor, *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, pages 1893–1901, 2023.

[Booch *et al.*, 2021] Grady Booch, Francesco Fabiano, Lior Horesh, Kiran Kate, Jonathan Lenchner, Nick Linck, Andreas Loreggia, Keerthiram Murgesan, Nicholas Mattei, Francesca Rossi, and Biplav Srivastava. Thinking fast and slow in ai. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(17):15042–15046, May 2021.

[Cardei *et al.*, 2025] Michael Cardei, Jacob K Christopher, Thomas Hartvigsen, Brian R. Bartoldson, Bhavya Kailkhura, and Ferdinando Fioretto. Constrained discrete diffusion, 2025.

[Dao *et al.*, 2022] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

[Devlin *et al.*, 2019] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019.

[Garbacea and Mei, 2022] Cristina Garbacea and Qiaozhu Mei. Why is constrained neural language generation particularly challenging? *arXiv preprint arXiv:2206.05395*, 2022.

[Geh *et al.*, 2024] Renato Geh, Honghua Zhang, Kareem Ahmed, Benjie Wang, and Guy Van Den Broeck. Where is the signal in tokenization space? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 3966–3979, 2024.

[Geh *et al.*, 2025] Renato Lui Geh, Zilei Shao, and Guy Van den Broeck. Adversarial tokenization, 2025.

[Gholami *et al.*, 2021] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. *CoRR*, abs/2103.13630, 2021.

[Hokamp and Liu, 2017] Chris Hokamp and Qun Liu. Lexically constrained decoding for sequence generation using grid beam search. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1535–1546, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[Holtzman *et al.*, 2020] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[Kahneman, 2011] Daniel Kahneman. *Thinking, fast and slow*. Farrar, Straus and Giroux, New York, 2011.

[Lu *et al.*, 2021] Ximing Lu, Peter West, Rowan Zellers, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. NeuroLogic decoding: (un)supervised neural text generation with predicate logic constraints. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4288–4299, Online, June 2021. Association for Computational Linguistics.

[Lu *et al.*, 2022] Ximing Lu, Sean Welleck, Peter West, Liwei Jiang, Jungo Kasai, Daniel Khashabi, Ronan Le Bras, Lianhui Qin, Youngjae Yu, Rowan Zellers, Noah A. Smith, and Yejin Choi. NeuroLogic A*esque decoding: Constrained text generation with lookahead heuristics. In *Proceedings of the 2022 Conference of the North American*

Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 780–799, Seattle, United States, July 2022. Association for Computational Linguistics.

[Pachet and Roy, 2011] François Pachet and Pierre Roy. Markov constraints: Steerable generation of markov sequences. *Constraints*, 16(2):148–172, apr 2011.

[Papadopoulos *et al.*, 2014] Alexandre Papadopoulos, Pierre Roy, and François Pachet. Avoiding plagiarism in markov sequence generation. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, page 2731–2737. AAAI Press, 2014.

[Papadopoulos *et al.*, 2015] Alexandre Papadopoulos, Pierre Roy, Jean-Charles Régin, and François Pachet. Generating all possible palindromes from ngram corpora. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, page 2489–2495. AAAI Press, 2015.

[Papadopoulos *et al.*, 2016] Alexandre Papadopoulos, Pierre Roy, and François Pachet. Assisted lead sheet composition using flowcomposer. In Michel Rueher, editor, *Principles and Practice of Constraint Programming*, pages 769–785, Cham, 2016. Springer International Publishing.

[Post and Vilar, 2018] Matt Post and David Vilar. Fast lexically constrained decoding with dynamic beam allocation for neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1314–1324, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[Régin, 2024] Florian Régin. *Generative constraint programming*. Theses, Université Côte d'Azur, December 2024.

[Régin and De Maria, 2023] Florian Régin and Elisabetta De Maria. Using on-the-fly model checking to improve constraint programming for dynamic problems. In *2023 IEEE 35th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 393–398, 2023.

[Régin *et al.*, 2024] Florian Régin, Elisabetta De Maria, and Alexandre Bonlarron. Combining Constraint Programming Reasoning with Large Language Model Predictions. In *30th International Conference on Principles and Practice of Constraint Programming (CP 2024)*, 2024.

[Schiex *et al.*, 1996] Thomas Schiex, Jean-Charles Régin, Christine Gaspin, and Gerard Verfaillie. Lazy arc consistency. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 1, pages 216–221, 1996.

[Sprockeels and Van Roy, 2024] Damien Sprockeels and Peter Van Roy. Expressing musical ideas with constraint programming using a model of tonal harmony. In Kate Larson, editor, *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pages 7753–7761. International Joint Conferences on Artificial Intelligence Organization, 8 2024. AI, Arts & Creativity.

[Trick, 2003] Michael Trick. A dynamic programming approach for consistency and propagation for knapsack constraints. *Annals of Operations Research*, 118:73–84, 2003.

[Wei *et al.*, 2022] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[Yao *et al.*, 2023] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: deliberate problem solving with large language models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

[Yao *et al.*, 2024] Shunyu Yao, Howard Chen, Austin W. Hanjie, Runzhe Yang, and Karthik R Narasimhan. COLLIE: Systematic construction of constrained text generation tasks. In *The Twelfth International Conference on Learning Representations*, 2024.