PRELIMINARY PREPRINT VERSION – DO NOT CITE
This is not the version of record. The final version will appear in the Proceedings of IJCAI-ECAI 2026.

Constraining Generative Models: A Survey from the Constraint Programming Perspective

Alexandre Bonlarron[1] , **François Pachet**[2] , **Pierre Roy**[3] and **Jean-Charles Régin**[1]
[1]Université Côte d'Azur, CNRS, I3S, France
[2]Sorbonne Université, CNRS, LIP-6, France
[3]Soundtrap, France
{alexandre.bonlarron,pachet,roypie,jcregin}@gmail.com

Abstract

Generative models produce long and high probability sequences, yet they often fail to satisfy explicit constraints set by users. Over the past two decades, Constraint Programming (CP) has provided a complementary paradigm: combining generative models with a constraint solver to guarantee feasibility. This survey reviews the main concepts behind these CP-driven hybrid approaches, from enforcing ubiquitous structural rules (e.g., length and patterns) to preventing plagiarism. It synthesizes how learned models can be treated as constraints, compiled structures, or probabilistic factors. We highlight what has remained stable across applications, then discuss how these principles transfer to the Large Language Model era and outline open challenges for controllable and trustworthy generative systems.

1 Introduction

The ability to steer, control, and constrain generative models (i.e., *their ability to comply with explicit requirements*) has become a fundamental problem in AI. These notions can be viewed as a hierarchy from heuristic guidance to hard compliance. Steering and control generally direct generation towards desired preferences, while constrainability refers to the mandatory properties enforcement. In particular, recent cases involving large language models (LLMs) have triggered a lot of discussion. This is because they can easily generate long passages of fluent text, but often fail to meet explicit requirements [Garbacea and Mei, 2025]. In contrast, constraint-solving approaches, such as Constraint Programming (CP), are designed around the opposite ideal promise: guaranteed satisfaction (if any solutions are returned) of formally stated requirements to the solver [Dechter, 2003]. Internally, they rely on constraint propagation and, if necessary, backtracking searches to prove and provide a solution that meets the requirements. This distinction is important because, in contexts where violations are unacceptable, users may prefer the soundness and guarantees of a constraint solver over the likelihood of compliance of steering methods.

For instance, Constrained Text Generation is a very active topic, and several thorough surveys already cover strategies for controlling language-model outputs [Wang and Shu, 2025; Garbacea and Mei, 2025]. Yet, the underlying problem is not specific to text; it is a very general AI question about how to combine learned predictors with explicit and enforceable requirements (i.e., constraints) [Cappart *et al.*, 2025]. That said, the rest of the paper focuses on enforcing constraints on a discrete sequence of elements.

Interestingly, this idea of a CP as a hosting technology has a strong historical precedent in a creative setting. Almost 25 years ago, the very same tension arose in interactive music generation with the Continuator [Pachet, 2003]. The problem contains two dimensions: (1) an ill-posed part that is capturing the style of an artist; (2) a well-posed part inherited from several domain-specific requirements (e.g., obeying a musical treatise) or regulations (e.g., not copying a protected song). The main answer was to rely, on the one hand, on a learned model to capture the style of the author, and, on the other hand, to integrate it with CP paradigm to overcome the limitations of pure generative systems [Pachet and Roy, 2011]. This allows the design of the first leadsheet generation system [Papadopoulos *et al.*, 2016]. It is used in assisted musical composition and production in real-world scenarios at Sony[1][Pachet *et al.*, 2021]. This history shows that a solver can serve as a dedicated constraint mechanism wrapped around generative models, providing a principled way to represent and enforce requirements.

This survey aims to cover the CP view on constrained sequence generation. In other words, it is when the task consists in producing the sequences of discrete elements that must meet a set of mandatory requirements. The aim is to demonstrate that this lens is timely and still relevant for classical applications (e.g., code [Jiang *et al.*, 2022], text [Bonlarron *et al.*, 2025], music [Vassallo *et al.*, 2025; Sprockeels and Van Roy, 2025; Sprockeels and Van Roy, 2024], and molecules [Manibod *et al.*, 2025]. In addition, many ideas proposed in CP for so-called generative tasks have gained new relevance in the era dominated by LLM-driven generation, an era where the demand for control and guaranty is becoming very difficult to ignore (e.g., copyright infringement [Ahmed *et al.*, 2026]). For instance, CP provided a way to express and avoid plagiarism (an important component of copyright infringement) [Papadopoulos *et al.*,

[1]https://www.flow-machines.com/

IJCAI-ECAI 2026 PREPRINT

2014]). This survey also argues that CP provides a reasonable foundation for building a trustworthy architecture in modern generative architectures. More generally, many of these approaches can be seen as instances of a deeper connection between combinatorial constraints, probabilistic inference, and sampling: an idea that will reappear in the discussion. The paper is organized as follows: Section 2 briefly gives background on constraint programming, MDDs and belief propagation. Section 3 covers the Markovian Era. Section 4 explores the Current trends. Section 5 discusses future directions. Section 6 concludes the paper.

2 Preliminaries

This section recalls the CP and probabilistic concepts used throughout the survey, retaining only the notation needed to discuss constrained sequence generation.

2.1 Constraint Programming (CP)

Constraint Programming (CP) is a paradigm for solving combinatorial problems [Rossi *et al.*, 2006]. It represents a problem $\mathcal{P}$ using a Constraint Satisfaction Problem (CSP). It is a triplet $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$, where (i) $\mathcal{X}$ is the set of *variables* $X_{1..n}$, which denote the unknowns; (ii) $\mathcal{D}$ is a *domain mapping* that maps each variable X_i to its set of admissible values $D(X_i)$; and (iii) $\mathcal{C}$ is the set of *constraints* $C_\ell(S_\ell)$, each defined over a scope $S_\ell \subseteq \mathcal{X}$, which encode the mandatory requirements by restricting the allowed combinations of values.

CP involves two mechanisms: (i) a programmable backtracking search that takes decisions (variable assignments of X_i with respect to $D(X_i)$); (ii) constraint propagation that studies, within the search tree, the consequences of a variable assignment (detects and removes impossible values).

In essence, solving a problem $\mathcal{P}$ in CP consists in identifying sub-problems $\mathcal{P}_{1,..m}$ such that $\bigcap_i^m \mathcal{P}_i = \mathcal{P}$ and mapping it to constraints in $\mathcal{C}$. CP associates each constraint with a filtering algorithm, also called a *propagator*, whose role is to eliminate values that are inconsistent with the constraint. Propagators play a key role in the efficiency and effectiveness of CP solvers. For instance, `All-Different` efficiently ensures that a subset of variables take different values [Régin, 1994].

2.2 Multi-Valued Decision Diagram (MDD)

MDDs are a compressed representation of solution sets. An ordered deterministic MDD [Castro *et al.*, 2022] is a layered directed acyclic graph (DAG), aligned with a given variable ordering in $\mathcal{X}$. It has a `root` node in the first layer and a `true terminal` node `tt` in the last layer. Each other layer corresponds to one variable X_i, and arcs go from layer i to layer $i{+}1$. Each arc is labeled by a value $a_k \in D(X_i)$. Any path from the `root` to `tt` defines an assignment $a = (X_1{=}a_1, \ldots, X_n = a_n)$.

A constraint C_i can be compiled into an MDD denoted by $\mathrm{MDD}(C_i)$ that represents the valid assignments of the constraint MDD-based propagators have been developed to integrate MDDs into CP Solver. Thanks to MDDs and their compression, we can represent constraints in extension that could not be represented by lists of allowed combinations. MDD compression often gains an exponential factor.

2.3 Belief Propagation (BP)

Belief Propagation (BP) is a general-purpose inference algorithm for probabilistic graphical models, originally introduced in the context of Bayesian networks [Pearl, 1986]. BP operates by exchanging local messages between variables and factors to compute marginal distributions. When the underlying factor graph is acyclic (i.e., a tree), BP is exact and runs in polynomial time. In loopy graphs, it becomes an approximate inference method, referred to as *loopy belief propagation*.

A factor graph represents the factorization of a joint probability distribution

$$p(X_1, \ldots, X_n) = \prod_{j=1}^{m} f_j(S_j),$$

where each factor f_j is defined over a subset S_j of variables. The graph is bipartite, with variable nodes and factor nodes, and edges indicating factor scopes. The sum–product algorithm computes, for each variable X_i, its marginal distribution $p(X_i = x_i) = \sum_{\{x_j | j \neq i\}} p(x_1, \ldots, x_n)$ by propagating messages that summarize the influence of the rest of the graph. Once marginals are computed, sampling can be performed easily, i.e., with no backtracks.

From a CP standpoint, BP can be interpreted as a probabilistic generalization of constraint propagation (see for intance CPBP [Pesant, 2019]). A CSP implicitly defines a uniform distribution over its solution set, and classical propagation algorithms can be seen as eliminating values with zero support. BP extends this idea by allowing *soft* supports: values are not only feasible or infeasible, but weighted by probabilities derived from factors. This connection has been studied extensively, notably through the correspondence between arc consistency and message passing on tree-structured constraint networks. BP is particularly attractive in hybrid settings where hard constraints and probabilistic models must coexist. BP preserves the declarative decomposition of constraints into local relations, while also supporting probabilistic reasoning and sampling. As a result, it provides a natural computational bridge between CP and statistical inference, a theme that will be exploited later in this survey.

2.4 Generative Model

A generative model is a probabilistic model that defines a distribution over data. It also enables the generation of plausible samples. In this survey, we are interested in generative models that produce sequences of discrete elements, most often learned from data. Formally, for a sequence of elements X_1^n from a finite vocabulary $\mathcal{V}$, a generative model defines a distribution $p_\theta(X_1^n)$, where θ is a parameter that can be learned. This includes well-known models such as Markov Models [Ross, 1995], Recurrent Neural Network (RNN/LSTM) [Hochreiter and Schmidhuber, 1997], Large Language Models [Vaswani *et al.*, 2017], and so on. While their implementations vary significantly over time, they all share the common objective of modeling and sampling from a sequence distribution. In many cases, this sequence distribution is computed at each position, with conditional distri-

IJCAI-ECAI 2026 PREPRINT

butions over $\mathcal{V}$,

$$p_\theta(X_1^n) = \prod_{k=1}^{n} p_\theta(X_k \mid X_1^{k-1}),$$

where X_1^0 means empty.

Variable-Length Sequences. Variable length is a recurring modeling issue because a CSP normally exposes a fixed set of variables. The works reviewed below handle it case by case: fixed horizons, padding or `EOS` symbols, duration/cost budgets, or repeated solves with increasing bounds. Another option is to solve a family of CSPs for increasing lengths.

3 The Markov+X Era: Style as a Constraint

Conceptually, the Markov+X paradigm introduced the following idea: generative models should no longer be generators to be sampled from, but constraints to be satisfied.

3.1 The Continuator

In 2002, The Continuator is introduced [Pachet, 2003], as an interactive music generation system. The Continuator leverages Markov chains to learn the style of a musician, in real time, and generate continuations of note sequences. Furthermore, the Continuator enabled musicians to "dialogue" with an algorithm that imitated their style. It gains recognition in jazz improvisation and music education contexts, because it produces stylistically coherent responses instantaneously[2].

The Continuator exposed a weakness of such a generative model: lack of control. Musicians could not reliably steer the output to meet explicit constraints (start and end with the same notes). This model generates what is most likely, even if that means unwanted notes. This insight motivated hybrid approaches: keeping Markov models for style and adding CP to enforce rules and structure.

3.2 Task Difficulty and CP Relevance

A natural question is why use CP here, beyond the usual benefits of declarative modeling and complete search. Sampling (or even enumerating) sequences from a generative model under constraints can become computationally intractable very quickly. Even for simple Markov models, adding a simple binary equality constraint, such as (with $i \neq j$):

$$C_{=}(X_i, X_j) : X_i = X_j \quad \text{over } (X_1, \ldots, X_n),$$

can already make constrained generation hard (i.e., #P-hard): one can no longer draw samples by local transitions independently of feasibility, because feasibility becomes an all-encompassing property with the constraints. Sampling under binary equality or under `regular` grammar constraint is #P-Complete [Rivaud *et al.*, 2016; Rivaud and Pachet, 2017].

This is precisely where CP is attractive: propagation and global constraints allow the solver to prune inconsistent assignments early and search directly in the space of feasible sequences, rather than relying on generate-and-test.

[2]https://youtu.be/cHKcq0D5EY4?si=a-KyTXxN8xPHC06B

3.3 The `Markov` Constraint

A major breakthrough for constraint-based generation of Markovian sequences is the `Markov` constraint [Pachet and Roy, 2011]. It provides a CP-friendly way of expressing a Markov chain by translating state-space transition as a global constraint over decision variables. In practice, this view allows the solver to search directly over sequences that satisfy Markovian transition structure, avoiding generate and test.

Since the `Markov` constraint turns the generative model into a constraint object within the CSP, the solver can find constrained sequences directly. Retroactively, this can be seen as one of the earliest successful integrations of a learned generative model into CP for constrained sequence generation. Compared with a purely grammar- or CP-based generator, Markov+X does not try to encode musical style entirely as formal rules. The Markov component preserves similarity to examples in the training corpus, while CP handles admissibility requirements that the learned transition model does not reliably enforce.

3.4 The 'X': Constraint Families

Once the Markov model is treated as a constraint, a natural question arises: how can additional requirements on the sequence be expressed? Constraint Programming provides a rich set of global constraints equipped with dedicated, efficient filtering algorithms, which have been extensively leveraged and extended in the works reviewed below. This section outlines the main families of such constraints. We review several classes of 'X' constraints to implement the following controls on the generated sequences: control on local elements, on temporal structure, on plagiarism, and on the distribution of elements in a sequence.

Local Constraints

The first attempt at enforcing hard control constraints on Markov sequence generation goes back to 2011 [Pachet *et al.*, 2011]. The authors cast controlled Markov generation as a CP task. They show that for a specific class of local constraints, it is possible to *compile* the constraints into another (non-homogeneous) Markov model. The key result is that the resulting model generates exactly the sequences satisfying the control constraints while preserving the original Markov distribution conditioned on feasibility, thereby enabling efficient sampling rather than mere solution construction. The approach is limited to two classes of constraints: (i) unary constraints and (ii) binary constraints between adjacent elements of the sequence. Note that these constraints involve elements denoted by their index position in the sequence. The case of temporal sequences will be addressed later.

Technically, their approach relies on enforcing arc-consistency on the induced CSP to eliminate infeasible values and transitions, followed by a non-trivial renormalization procedure that preserves the original probability distribution conditioned on constraint satisfaction. The resulting model generates exactly the feasible sequences with their correct relative probabilities using a simple random walk, without search, making it suitable for real-time applications.

The typical example is to generate melodies from a Markov model of note transitions with control on individual notes,

IJCAI-ECAI 2026 PREPRINT

e.g., sample from the Markov model a melody that ends with note C. A subsequent application to lyrics generation is given in [Barbieri *et al.*, 2012].

Temporal Sequences

Sequential generation models typically represent sequences as lists of discrete variables indexed by position. While this representation is natural for symbolic generation and is compatible with local constraints, as said above, it is intrinsically limited when temporal properties matter: in many domains, most notably music, but also text, the sequence is not fully defined by its index positions alone, but by the durations of and temporal positions of its elements.

In general, there is no direct relationship between *index*- and *time-based* positions in a sequence. This gives rise to a class of constraints that cannot be expressed or enforced locally: fixed total duration, alignment with external temporal references, or overlap relations between events. Such constraints depend on the global accumulation of durations and on implicit start and end times that are known only once the sequence is complete. As a consequence, generate-and-test approaches quickly become intractable, and dedicated global constraints are required.

The `Meter` constraint focuses on total duration and temporal accumulation, whereas the `Allen` constraint captures qualitative temporal relations between events. Note that standard sequencing or scheduling constraints are ill-suited to express such qualitative relations in a compact and reusable way.

`Meter` Constraint. Metrical (duration) constraints are ubiquitous in sequence generation whenever elements carry a length or cost: syllable counts in poetry or durations in music. The challenge is that the number of generated elements is unknown in advance, because different choices contribute different durations. Enforcing an exact total duration can be viewed as a summation constraint, closely related to subset-sum/knapsack-style reasoning. A practical way to handle it is dynamic programming (DP) over partial sums: i.e., compute reachable cumulative durations and represent feasible continuations as a layered automaton (or DAG) whose states are partial sums and whose transitions add a cost. Here, it leverages a similar idea, instead of managing partial cost and thus expanding the full DP state-space. Each state manages sumset of costs. This encoding has a polynomial size [Khovanskii, 1992], leading to a pseudo-polynomial complexity of the filtering algorithm [Roy and Pachet, 2013]. This constraint was reformulated using MDDs and recasted as `MDDMarkovProcess` [Perez and Régin, 2017]

`Allen` Constraint. The Allen global constraint [Roy *et al.*, 2016] uses interval algebra [Allen, 1983] to represent temporal relations between sequence events. Formally, the Allen constraint operates on a sequence of event variables, each associated with a duration and an implicit start time derived from the sequence. Given a reference time interval $[t_1, t_2]$ and an Allen relation (or disjunction of relations), the constraint enforces which events in the sequence satisfy this relation with respect to the reference interval. In addition, the constraint introduces set variables that collect (i) the indices of sequence positions whose corresponding events satisfy the relation and (ii) the set of events occurring at those positions. This design enables high-level temporal structure to be expressed declaratively and combined with other sequence-level constraints within a unified constraint model.

The Allen constraint considered here should not be confused with solving arbitrary qualitative temporal networks in the full Allen algebra. In our setting, intervals are induced by the generated sequence: event start times are determined by cumulative durations, and Allen relations are checked with respect to fixed reference intervals. Composite relations can be supported as disjunctions of admissible base relations, but only within this restricted global-constraint setting, not as an arbitrary network of pairwise interval relations.

Accordingly, the consistency notion used here is arc-consistency in the CP/MDD sense. Once the global constraint is compiled into an MDD, propagation removes values of sequence and set variables that do not occur on any accepting path. This should not be confused with path-consistency algorithms for general Allen networks.

The main technical contribution is a filtering model based on MDDs. In this approach, the full extension of the Allen constraint is compiled into an MDD whose paths represent feasible sequences with respect to durations and temporal relations. The MDD is cleverly constructed by not incorporating all variables in order to maintain high compression. These variables are filtered by introducing channeling constraints between the MDD and them. Standard MDD-based propagation then enforces arc-consistency between the sequence variables and the associated set variables. A key result is that multiple Allen constraints defined on the same sequence can be merged into a single MDD, allowing arc-consistent propagation of their conjunction without a prohibitive blow-up in representation size. This makes the approach scalable to realistic problems with several interacting temporal constraints.

The Allen-constraint work sketches applications to structured music generation, such as multitrack audio synchronization task as well as lead-sheet generation constrained by chordal temporal structure. Overall, the Allen constraint shows how qualitative temporal reasoning can be integrated into sequence generation within a constraint-based framework, and illustrates the impact of MDD-based global constraints for enforcing high-level structural properties.

Avoiding Plagiarism

`MaxOrder` Constraint. While constrained sampling from Markov models can be made efficient, a key difficulty in creative domains (e.g., music) is verbatim reuse: as the Markov order increases, random-walk generation can reproduce long substrings of the training corpus, which may be perceived as plagiarism [Papadopoulos *et al.*, 2014]. The produced sequence must follow the order-k Markov transitions and must avoid any corpus substring of length L_0 (encoded as a set of NO-GOODS). This captures only an operational notion of verbatim reuse, not the whole musical plagiarism problem: transposition, rhythmic displacement, ornamentation, or motif transformation can preserve recognizable identity without reproducing the exact same substring. They enforce this constraint by building an automaton whose language is the set of Markovian sequences excluding those that contain any no-

IJCAI-ECAI 2026 PREPRINT

good g, i.e., $L(\mathrm{MO}) = L(\mathrm{Markov}) \cap \bigcap_{g \in N} \overline{L(A(g))}$, where $A(g)$ corresponds exactly to the sequences having g as a contiguous substring. Instead of performing the canonical intersection, they build the maximum-order automaton in time linear in the size of the no-goods input by using a trie (prefix tree) and then apply the `Regular` constraint [Pesant, 2004] to perform the filtering. This constraint has been reformulated in MDD-based fashion [Perez and Régin, 2015].

Spread and Distribution

`Voss` Constraint. It was introduced to enable the generation of more *natural* sequences, by enforcing 1/f (pink) noise, a statistical property ubiquitous in natural signals and human-generated sequences [Pachet *et al.*, 2015]. Such spectra reflect long-range correlations and a characteristic balance between regularity and variability, often associated with perceived naturalness. These properties are impossible to capture with finite-order Markov models, which by construction encode only local dependencies. Building on Voss's classic stochastic algorithm [Voss and Clarke, 1978], the authors introduce the Voss constraint, modeling 1/f behavior as a tree of ternary sum constraints that can be efficiently filtered. This formulation makes 1/f structure available as a global hard constraint, composable with Markov or other symbolic constraints. Applied to melody generation, the approach produces sequences whose spectra exhibit 1/f behavior independently of local constraints, illustrating how constraint-based frameworks can complement Markov models by enforcing global statistical structure. This work was later improved in [Perez *et al.*, 2018].

3.5 Exact Sampling with Belief Propagation (BP)

Beyond inference, BP can be used as an exact sampling mechanism when applied to tree-structured factor graphs. This property is central generation methods combining Markov models with hard constraints, where the goal is to produce sequences that both satisfy constraints and follow a target probability distribution.

[Papadopoulos *et al.*, 2015a] showed how constrained Markov sequence generation can be reformulated as sampling in a factor graph. The key idea is to compile the conjunction of a Markov model and Regular constraints into local unary and binary factors arranged in a linear chain. In this representation, non-zero probability assignments correspond exactly to sequences accepted by the automaton and consistent with the Markov transitions. In this setting, BP is applied in two phases: first, backward pass computes messages that summarize, for each position, how partial assignments can be extended to a full sequence satisfying all constraints; second, a forward pass then samples values sequentially, using these messages to draw each variable according to the correct marginal probability. The resulting sequence is guaranteed to be sampled exactly from the target distribution conditioned on the constraints without backtracking.

This approach has two major benefits. First, it avoids the combinatorial explosion associated with naive constrained sampling from Markov models, by exploiting a representation where global feasibility is enforced locally. Second, it cleanly separates feasibility (encoded by zero-probability factors) from preference (encoded by probabilistic weights).

From a CP perspective, the backward messages can be seen as a probabilistic equivalent of domain filtering: they specify which values can participate in a globally consistent solution, and with what weight.

Exact sampling with BP thus occupies a distinctive position between classical CP and generative modelling. Unlike CP-based generation, which typically enumerates or optimizes over feasible solutions, BP directly produces samples with the correct distribution. Unlike unconstrained generative models, it guarantees constraint satisfaction by construction.

4 Modern Trends, The Story is Repeating

Recently, CP-based approaches to constrained generation using CP can be positioned along a spectrum depending on the role played by constraints. At one extreme, constraints fully define the generative model, and solvers act as the main generators. At the other end, constraints are soft and simply guide generation. Finally, there is a compromise where the solver plays the role of a guardrail to the generative model, correcting and repairing the sequence on the fly. This compromise also covers sparse-but-hard requirements: a generated action sequence, code fragment, or structured answer may have only a few explicit rules, but these rules can be non-negotiable because they encode safety, admissibility, or consistency conditions. In such cases, constraints do not model the whole generative process; they constrain what the learned model is allowed to output. The overall landscape is summarized in Table 1.

4.1 Not Yet Coupled and Loosely Coupled

Constraints First: a Constrain-Then-Predict approach

Strongly constrained problems are problems in which feasibility is very hard to achieve. This happens when the constraints dominate the solution space. In that case, the task can be viewed primarily as a discrete combinatorial optimization problem, with quality being almost secondary. Unfortunately, quality cannot be entirely ignored during sequence generation (e.g., fluency in text).

In this context, the Constraints First approach involves computing feasibility first and postponing quality preferences to a second stage [Bonlarron *et al.*, 2023]. A corpus-derived successor structure (e.g., n-grams inspired by `Markov` constraint) is used to enforce local transition, while the domain-specific rules are expressed as hard constraints. The CSP model is entirely computed and expressed with MDDs. It can leverage the MDD intersection and/or DP-formulation. From a very high-level, we have: $\mathrm{MDD}_{final} = \bigcap_{i=1}^{m} \mathrm{MDD}(C_i)$.

The resulting model enables the generation of a large set of candidate sequences that satisfy all constraints. An LLM (`LLM-as-an-Oracle`) is then used as a curation mechanism (perplexity scoring (PPL)) to select the most plausible candidates among the feasible ones. Formally, each solution in the MDD receives a PPL score and is sorted by this score.

Constraint First has been used to solve the constrained sentences generation task, where the constraints are strong and can be found in a well-known reading test specification in Vision Research (e.g., MNREAD [Legge, 2021]).

IJCAI-ECAI 2026 PREPRINT

Approach Representative paper	Learned model role	CP/solver role	Hard constraints live in	sat?
Markov + CP [Pachet and Roy, 2011]	Local score (n-gram / transitions)	Search + propagation + optimization	Solver (CP)	✓
Markov + MDD [Perez and Régin, 2017]	Arc costs on MDD (compiled Markov)	Compilation over MDD (generator)	Compiled structure (MDD)	✓
Markov + BP + CP [Papadopoulos *et al.*, 2015a]	Local factors (Markov, automaton)	Exact sampling on tree factor graph	Compiled structure (factor graph)	✓
MDD + LLM [Bonlarron *et al.*, 2023]	Scoring of feasible solutions	Compilation over MDD (generator)	Compiled structure (MDD)	✓
LLM + CP [Régin *et al.*, 2024]	Domain generator / expansion	Search + propagation	Solver (CP)	✓†
RNN + RL + CPBP [Lafleur *et al.*, 2022]	RL Policy + generator	Marginal guidance / steering	Guidance / violations model	×
LLM + CPBP [Manibod *et al.*, 2025]	Generator (next-token proposals)	Marginal guidance / steering	Guidance (greedy) / Solver	×/✓

✓: guaranteed hard-constraint satisfaction. ✓†: (✓) but in practice anytime. ×/✓: task/setting dependent.

Table 1: Overview of the methods: learned model/solver role, how hard constraints are enforced, and satisfaction guarantees.

Later, Bonlarron and Régin generalized Constraints First into a framework [Bonlarron and Régin, 2024a], introducing a CSP model for RADNER-like sentence specifications [Radner, 2017]. Furthermore, they highlight that several NLP requirements find a natural counterpart in CP.

Note: Retroactively, in both problems, the curation step leveraging `LLM-as-an-Oracle` could have been further refined by `LLM-as-a-Judge` [Zheng *et al.*, 2023].

`ngramMarkov`: `Markov` Constraint as LLM Surrogate

The two stages of Constraints First raise a question: can the LLM-based quality information (e.g., PPL) be injected earlier during CP search, rather than only after it?

Directly calling an LLM to score millions of partial assignments within the solver is slow, and optimizing the global LLM likelihood as a CP objective is not straightforward. A workaround is to approximate the global PPL score with an aggregation of local scores: sequences with low global PPL are often composed of locally high-probability transitions (although local and global quality do not align). This motivates a new constraint that leverages the `Markov` and `MDDMarkovProcess` filtering ideas, and uses LLM-derived transition scores [Bonlarron and Régin, 2024b].

In practice, one builds a Markov model where transitions are weighted using LLM probabilities. This defines an `ngramMarkov` constraint: at each step, candidate next tokens are evaluated given the current prefix, and continuations whose probability falls below a dynamically updated threshold are pruned (i.e., a gliding threshold).

Empirically, adding this constraint within *Constraints First* can substantially reduce the combinatorial explosion induced by complete search, by eliminating many low-likelihood branches early. The trade-off is that the local proxy may also prune some globally acceptable (or even very good) sequences.

4.2 Strongly Coupled

GenCP: Predict-and-Constrain (LLM+CP)

GenCP instantiates the `LLM-as-an-Oracle` idea for domain generation in CP. Instead of requiring the modeler to specify the domain $\mathcal{D}$, GenCP delegates part of this work to an LLM, which proposes candidate values for each variable, $\hat{D}_{\text{LLM}}(X_i)$, during search.

At solving time, this changes the classical CP loop. Rather than only removing unsupported values from $D(X_i)$, the solver can *query* the LLM for promising assignments for the current decision variable. The returned candidates form a (partial) domain $\hat{D}_{\text{LLM}}(X_i)$ (where the values are ordered and weighted by the $p_{\text{LLM}(.)}$) for the next branching decision. In this sense, GenCP acts like a CP decoding strategy: it explores LLM-proposed continuations while keeping CP backtracking and propagation to recover from dead ends [Régin *et al.*, 2024].

Later work adds a Masked Language Model (MLM) to reduce the strictly left-to-right limitations of autoregressive generation. This setup combines autoregressive proposals with MLM "look-ahead," using constraint propagation as the bridge: propagation detect inconsistencies early and enables backtracking, while the MLM preview helps spot dead ends sooner and improves assignement of $D_{\text{LLM}}(X_i)$ given an horizon d and $\hat{D}_{\text{MLM}}(X_{i+1}, ..., X_{i+d})$ [Bonlarron *et al.*, 2025]. In that study, the approach was evaluated only with summation propagation (e.g., `meter`-like constraint).

GenCP was introduced in 2024 for constrained sentence generation on the COLLIE benchmark [Yao *et al.*, 2024], i.e., a weakly constrained setting. Later work extended it to paragraph-level generation and reported better efficiency (in terms of the number of LLM calls), while also improving sentence-level results. In both cases, GenCP enforces constraints by construction: any output returned by the solver is guaranteed to satisfy the constraint set (the solver never outputs infeasible sequences). This highlights a key advantage of CP-based generation: hard satisfaction guarantees come from the paradigm itself, not from post-hoc filtering.

IJCAI-ECAI 2026 PREPRINT

Additional Coupling Patterns

Replacing Constraint Propagation with BP. Pesant proposed CPBP [Pesant, 2019], a CP solver in which constraint propagation is replaced by BP. By estimating marginal probabilities over each variable's domain (via solutions counting-based algorithm). Pesant shows that these marginals can also guide the search, for example by inducing a branching heuristic on max marginal. It is then important to mention briefly a third integration pattern, where CPBP marginals are combined with an external generative model during inference time. Ideally, the generative model proposes a distribution for the next decision token, and the CPBP machinery should reshape that distribution based on CSP constraints so that it better reflects the "true" distribution of the constrained samples.

RNN+RL-Informed CPBP. This line is illustrated by CPBP+RNN in [Lafleur *et al.*, 2022]. It targets melody-line generation, i.e., generating sequences of pitch notes. It revolves around three components: (i) an RNN trained on Bach, (ii) a Reinforcement Learning agent that chooses the current note, and (iii) a CPBP model together with a classical CP model (called the violations model).

One could almost say, this is a quartet rather than a trio, in the sense that there is indeed a standard CP model whose role is to count violations, i.e., to infer a "distance" to satisfaction. In short, this architecture trains an RL agent to pick the next note so as to generate music in the style of Bach while still respecting musical rules coming from counterpoint theory. To do so, the RL reward function is computed as a combination of: the probability assigned by the RNN to the current note, the marginal probabilities produced by CPBP, and finally the violation cost. Unfortunately, this approach does not guarantee satisfaction, but the authors report a positive effect on it (82.4% of constraint satisfaction rates vs 77.2% for the closest baseline).

LLM-Informed CPBP. The same general idea can be reused with a task-specific LLMs [Manibod *et al.*, 2025]. The probability distribution produced by an LLM at each step is combined with marginal probabilities computed from a CPBP model encoding the problem structure. In practice, it leverages the same principle as the domain generation of GenCP: $\hat{D}_{\text{LLM}}(X_i)$. They cast it as a constraint named $\texttt{Oracle}(X_i, \hat{D}_{LLM}(X_i))$. Thanks to the BP mechanism, it allows the LLM distribution to communicate with the others contraints of the problem thanks to the `Oracle` constraint. This approach is evaluated on two tasks: SMILES molecule generation and Chord-conditioned Melody Generation. Notably, for molecule generation, they achieve a constraint satisfaction rate of 92%, with an average perplexity (PPL) score of 8.35. In contrast, the baseline CPBP without Oracle achieves a constraint satisfaction rate of 100%, but with an average PPL score of over 1200. These results highlight a common challenge throughout the survey once again: the difficulty of reconciling constraint satisfaction with learned distributions.

Related Work

What CP Adds Beyond Decoder-Side Control. A large adjacent literature studies constrained generation, especially in NLP: grid or lexically constrained beam search and predicate-logic decoding methods [Wang and Shu, 2025]. These approaches operate at the model interface. At each step, they reshape the set of legal next tokens, or rank beams so that a required phrase, grammar, or logical condition is eventually realized. This is often exactly the right tool when the requirement has a compact sequential representation. The limitation is that control remains organized around the decoder prefix: constraints must be exposed as token admissibility, greedy policy, or local lookahead. The CP viewpoint starts from the opposite direction. A generated object is first a set of decision variables linked by constraints; the probabilistic model is one source of preference or proposals among others. GenCP or GeAI BlanC are a useful bridge case: the LLM proposes a partial domain for the next variable, much like a decoding policy, while constraint or belief propagation and backtracking decide which continuations remain globally feasible. This makes it natural to combine requirements that live at different levels, such as meter, temporal intervals, plagiarism automata, global cardinalities, and harmonic structure, and to use propagation and backtracking when early choices make later feasibility impossible. In essence, CP targets a different bottleneck: heterogeneous, non-local requirements whose interaction is the problem itself.

Broader Context. While constrained generation is also studied in areas such as story generation, automated planning, program synthesis, grammar-based generation, and formal-methods-inspired decoding, this survey focuses on the Constraint Programming perspective. Our goal is not to cover all forms of constrained generation, but to isolate what CP contributes specifically: first-class constraint modeling, propagation, backtracking search, compact compiled structures such as automata and MDDs, and hard satisfaction guarantees. These adjacent traditions are complementary, but they usually rely on different computational mechanisms, such as plan-space search, grammar expansion, decoder masking, or domain-specific symbolic representations.

5 Open Directions

Many problems involving constrained text or music generation remain out of reach for current approaches. We mention a few here because they are interesting not only in themselves, but also because they exemplify more advanced problem-solving skills that still need to be developed.

5.1 Palindromes and Variations

Palindromes are sequences that read identically forwards and backwards. They have been studied in the past, in particular in [Norvig, 2002]. These studies show that it is possible to generate either meaningful short palindromes (e.g., *"A man, a plan, a canal: Panama"*) or meaningless long ones, such as the example reported in [Papadopoulos *et al.*, 2015b] with length 84,541 words. However, results suggest that current generation algorithms fail to simultaneously enforce strong global structural constraints and rich semantic content. In particular, the ability to control semantics in long palindromic sequences remains out of reach.

IJCAI-ECAI 2026 PREPRINT

A similar limitation appears in music generation. To the best of our knowledge, no approach is able to generate long musical pieces that are both palindromic and musically coherent. The same holds for fugues, a highly-structured compositional form in which musical motifs are repeated and transformed across the entire piece.

5.2 Controlling Sequence Generation with Complex Features

We give here an example of a natural yet particularly difficult problem encountered in music generation, for which no solution has been found so far. The task is to generate a sequence of chords according to a probabilistic model (e.g., a Markov model) that constrains transition probabilities between chords, while constraining global harmonic features of the sequence. For example, enforcing that the sequence *modulates* a given number of times (e.g., exactly two) or explores a fixed, small number of tonalities.

A sequence is said to modulate if its harmonic analysis yields more than one underlying tonality. Such an analysis typically requires DP and can only be performed on the whole generated sequence. The exact computation is actually NP-hard as it involves both the minimization of a total cost (reducing number of modulations) but also the minimization of the total number of different tonalities [Pachet, 2026].

This makes the problem challenging for CP-based generation. If the harmonic analysis is performed only after a full sequence has been generated, the method degenerates into generate-and-test. But if the analysis is integrated into the search, the solver must propagate through an optimization procedure over latent variables. In CP terms, this resembles combining a probabilistic sequence model with an NValue-like constraint over hidden states and a shortest-path or Viterbi-style global constraint. The example therefore illustrates a broader open problem: how to generate sequences while constraining global features that are not directly observable, but are computed by a non-trivial inference procedure over the whole generated object. Therefore, the only viable approach we currently know is generate-and-test. More generally, this example illustrates the difficulty of efficiently integrating global features computed on the whole sequence directly into the search process.

6 Discussion & Conclusion

A key problem for AI today is to build systems able to combine efficient search with guarantees of correctness (traditionally the domain of CSP and SAT solvers) with commonsense inference and statistical generalization, which are the strengths of learned generative models and, more recently, LLMs. Achieving this unification requires principled mechanisms to mix gradient-based learning with filtering and propagation in the sense of constraint programming. As illustrated in this survey, strong results have already been obtained for Markov models, where learned distributions can be turned into constraints, compiled structures, or probabilistic factors. Extending these ideas to models with larger contexts, such as transformers, remains open in the general case. The approaches reviewed in this paper are part of a deeper and well-established connection between statistical physics, combinatorial search, and information theory. This connection has been extensively developed in the statistical physics of disordered systems, notably in the work of Mézard and Montanari [Mézard and Montanari, 2009], where CSPs are analyzed as energy landscapes and solution sets as Gibbs distributions. From this perspective, inference, counting, and sampling are not peripheral tasks but central objects of study. Belief Propagation, in particular, emerges as a unifying algorithmic principle: originally derived as an approximation of marginal probabilities in graphical models, it can also be interpreted as a message-passing relaxation of constraint propagation, and as a tool for sampling solutions of CSPs according to well-defined distributions. The use of BP for exact or approximate sampling under constraints, as described in this survey, is therefore not an isolated technique but one concrete instantiation of this deep correspondence. We believe that many other bridges between CP, information-theoretic inference, and statistical physics remain to be explored, and that pursuing them is likely to yield both new algorithms and new theoretical insights into constrained generative modeling.

To make progress, we argue that it is essential to ground future research in concrete, non-artificial problems that arise from real practice. The techniques surveyed here were not developed in abstraction, but in response to genuine application-driven difficulties: interactive music generation, tonal harmony modeling, plagiarism avoidance, temporal reasoning, or constrained sentence generation for vision research. In all these cases, the difficulty of the task was not designed for benchmarking purposes; it emerged naturally from the requirements of the domain.

As Leslie Lamport once remarked [Lamport, 1987], some of the most interesting theoretical problems arise only once systems are implemented and confronted with reality. Artificial benchmarks, while useful, often fail to expose the true complexity of constraint interaction, propagation limits, or the tension between feasibility and quality. In contrast, non-artificial benchmarks, rooted in actual use cases, tend to be both more demanding and more revealing.

Looking forward, we believe that the field would benefit from embracing even more challenging, practice-inspired problems, involving global semantic properties, long-range structure, or constraints that can only be evaluated on complete sequences. Problems such as controlled modulation in music, long-form palindromic structures with semantic coherence, or deeply nested structural constraints in text remain largely out of reach. Rather than avoiding such problems, we see them as essential stress tests for future hybrid architectures. As history has repeatedly shown, it is by confronting AI systems with real, difficult, and sometimes uncomfortable requirements that the most durable theoretical and algorithmic advances are made.

IJCAI-ECAI 2026 PREPRINT

Acknowledgments

This work has been supported by the French government, through the 3IA Côte d'Azur Investments in the Future project managed by the National Research Agency (ANR) with the reference number ANR-19-P3IA-0002.

References

[Ahmed *et al.*, 2026] Ahmed Ahmed, A. Feder Cooper, Sanmi Koyejo, and Percy Liang. Extracting books from production language models, 2026.

[Allen, 1983] James F. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843, November 1983.

[Barbieri *et al.*, 2012] Gabriele Barbieri, Francois Pachet, Pierre Roy, and Mirko Degli Esposti. Markov constraints for generating lyrics with style. *Frontiers in Artificial Intelligence and Applications*, 242, 08 2012.

[Bonlarron and Régin, 2024a] Alexandre Bonlarron and Jean-Charles Régin. Intertwining cp and nlp: The generation of unreasonably constrained sentences. In *IJCAI*, 2024.

[Bonlarron and Régin, 2024b] Alexandre Bonlarron and Jean-Charles Régin. Markov constraint as large language model surrogate. In *IJCAI*, 2024.

[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 *IJCAI*, 2023.

[Bonlarron *et al.*, 2025] Alexandre Bonlarron, Florian Régin, Elisabetta De Maria, and Jean-Charles Régin. Large language model meets constraint propagation. In James Kwok, editor, *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25*, pages 10036–10044, 2025.

[Cappart *et al.*, 2025] Quentin Cappart, Tias Guns, Michele Lombardi, Gilles Pesant, and Dimos Tsouros. Combining constraint programming and machine learning: From current progress to future opportunities. *Journal of Artificial Intelligence Research*, 84, 2025.

[Castro *et al.*, 2022] Margarita P. Castro, Andre A. Cire, and J. Christopher Beck. Decision diagrams for discrete optimization: A survey of recent advances. *INFORMS Journal on Computing*, 34(4):2271–2295, 2022.

[Dechter, 2003] Rina Dechter. *Constraint processing*. Elsevier, 2003.

[Garbacea and Mei, 2025] Cristina Garbacea and Qiaozhu Mei. Why is constrained neural language generation particularly challenging? *Transactions on Machine Learning Research*, 2025.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997.

[Jiang *et al.*, 2022] Nan Jiang, Maosen Zhang, Willem-Jan Van Hoeve, and Yexiang Xue. Constraint reasoning embedded structured prediction. *Journal of machine learning research*, 23(345):1–40, 2022.

[Khovanskii, 1992] Askold Georgievich Khovanskii. Newton polyhedron, hilbert polynomial, and sums of finite sets. *Functional Analysis and Its Applications*, 26(4):276–281, 1992.

[Lafleur *et al.*, 2022] Daphné Lafleur, Sarath Chandar, and Gilles Pesant. Combining reinforcement learning and constraint programming for sequence-generation tasks with hard constraints. In *CP 2022*, 2022.

[Lamport, 1987] Leslie Lamport. My writings. https://lamport.azurewebsites.net/pubs/pubs.html, 1987. Accessed: 2026-06-08.

[Legge, 2021] Gordon E. Legge. *Psychophysics of Reading in Normal and Low Vision*. CRC Press, 2 edition, 2021.

[Manibod *et al.*, 2025] Virasone Manibod, David Saikali, and Gilles Pesant. Constrained sequential inference in machine learning using constraint programming. In James Kwok, editor, *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25*, 2025.

[Mézard and Montanari, 2009] Marc Mézard and Andrea Montanari. *Information, Physics, and Computation*. Oxford University Press, Oxford, UK, 2009.

[Norvig, 2002] Peter Norvig. World's longest palindrome sentence. http://norvig.com/palindrome.html, 2002.

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

[Pachet *et al.*, 2011] Francois Pachet, Pierre Roy, and Gabriele Barbieri. Finite-length markov processes with constraints. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence, IJCAI'11*, pages 635–642. AAAI Press, 2011.

[Pachet *et al.*, 2015] François Pachet, Pierre Roy, Alexandre Papadopoulos, and Jason Sakellariou. Generating 1/f noise sequences as constraint satisfaction: The voss constraint. In *Proceedings of IJCAI 2015*, pages 2482–2488, 2015.

[Pachet *et al.*, 2021] François Pachet, Pierre Roy, and Benoit Carré. Assisted music creation with flow machines: towards new categories of new, 2021.

[Pachet, 2003] Francois Pachet. The continuator: Musical interaction with style. *Journal of New Music Research*, 32(3):333–341, 2003.

[Pachet, 2026] François Pachet. Tonal parsimony in chord-sequence analysis: combining modulation cost and tonal vocabulary, 2026.

[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.*, 2015a] Alexandre Papadopoulos, François Pachet, Pierre Roy, and Jason Sakellariou. Exact sampling for regular and markov constraints with belief propagation. In Gilles Pesant, editor, *Principles and Practice of Constraint Programming*, pages 341–350, Cham, 2015. Springer International Publishing.

[Papadopoulos *et al.*, 2015b] Alexandre Papadopoulos, Pierre Roy, Jean-Charles Régin, and François Pachet.

IJCAI-ECAI 2026 PREPRINT

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.

[Pearl, 1986] Judea Pearl. Fusion, propagation, and structuring in belief networks. *Artificial Intelligence*, 29(3):241–288, 1986.

[Perez and Régin, 2015] Guillaume Perez and Jean-Charles Régin. Efficient operations on MDDs for building constraint programming models. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2015.

[Perez and Régin, 2017] Guillaume Perez and Jean-Charles Régin. MDDs: Sampling and probability constraints. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming*, page 226–242, 2017.

[Perez *et al.*, 2018] Guillaume Perez, Brendan Rappazzo, and Carla P. Gomes. Extending the capacity of 1/f noise generation. In *CP 2018*, 2018.

[Pesant, 2004] Gilles Pesant. A regular language membership constraint for finite sequences of variables. In Mark Wallace, editor, *Principles and Practice of Constraint Programming – CP 2004*, pages 482–495, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[Pesant, 2019] Gilles Pesant. From support propagation to belief propagation in constraint programming. *J. Artif. Intell. Res.*, 66:123–150, 2019.

[Radner, 2017] W. Radner. Reading charts in ophthalmology. *Graefe's Archive for Clinical and Experimental Ophthalmology*, 255(8):1465–1482, August 2017.

[Régin, 1994] Jean-Charles Régin. A filtering algorithm for constraints of difference in csps. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (Vol. 1)*, page 362–367, USA, 1994. American Association for Artificial Intelligence.

[Rivaud and Pachet, 2017] Stephane Rivaud and François Pachet. Sampling markov models under constraints: Complexity results for binary equalities and grammar membership, 2017.

[Rivaud *et al.*, 2016] Stephane Rivaud, François Pachet, and Pierre Roy. Sampling markov models under binary equality constraints is hard. In *Journées Francophones sur les Réseaux Bayésiens et les Modèles Graphiques Probabilistes*, Clermont-Ferrand, France, 2016.

[Ross, 1995] Sheldon M Ross. *Stochastic processes*. John Wiley & Sons, 1995.

[Rossi *et al.*, 2006] Francesca Rossi, Peter Van Beek, and Toby Walsh. *Handbook of constraint programming*. Elsevier, 1st edition, 2006.

[Roy and Pachet, 2013] Pierre Roy and Francois Pachet. Enforcing meter in finite-length markov sequences. *Proceedings of the AAAI Conference on Artificial Intelligence*, 27(1):854–861, Jun. 2013.

[Roy *et al.*, 2016] Pierre Roy, Guillaume Perez, Jean-Charles Régin, Alexandre Papadopoulos, F. Pachet, and M. Marchini. Enforcing structure on temporal sequences: the Allen constraint. In *International conference on principles and practice of constraint programming*, page 786–801. Springer, 2016.

[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.

[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, 2024.

[Sprockeels and Van Roy, 2025] Damien Sprockeels and Peter Van Roy. Towards a practical tool for music composition: Using constraint programming to model chord progressions and modulations. In James Kwok, editor, *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25*, pages 10171–10179, 2025.

[Vassallo *et al.*, 2025] Juan S. Vassallo, Örjan Sandred, and Julien Vincenot. Neuralconstraints: integrating a neural generative model with constraint-based composition. *Frontiers in Computer Science*, Volume 7 - 2025, 2025.

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

[Voss and Clarke, 1978] Richard F. Voss and John Clarke. "1/f noise" in music: Music from 1/f noise. *The Journal of the Acoustical Society of America*, 63(1):258–263, 01 1978.

[Wang and Shu, 2025] Haoran Wang and Kai Shu. Make every token count: A systematic survey on decoding methods for foundation models, 01 2025.

[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.

[Zheng *et al.*, 2023] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

IJCAI-ECAI 2026 PREPRINT