MUSIC AS PROGRAMS: DISCOVERY OF MUSICAL STRUCTURE VIA PROGRAM INDUCTION

Anonymous Authors
Anonymous Affiliations
anonymous@ismir.net

ABSTRACT

Music is highly structured, in forms that reflect cultural traditions and human cognitive constraints. Building musical AI that explicitly models this structure can bring insight into music cognition, and enable more controllable and human-centered tools to empower musicians. To this end, we build on recent work on concept representation in cognitive science to model structured musical concepts as generative programs, and model reasoning about music structure as program induction. We leverage large language models (LLMs) as a backend for generating programs for tractable inference, where structure is represented by program-like primitives and their compositional transformations. In line with recent research on world-modeling with LLM-based program synthesis, we explore encoding these programs in a Turing-complete language, such as Python. We compare unconstrained program generation, and program generation constrained by a musical DSL of four operations (transpose, invert, retrograde, identity), finding that DSL constraints improve program discovery on unseen sequences. This result demonstrates the proof-of-concept validity and value of our approach toward building computational models of music cognition.

1. INTRODUCTION

Music is highly structured. This structure reflects in part the communicative pressure of transmitting music between practitioners [1–3], and in part the combinatorial nature of human thought [4, 5] reflected in the process of music creation. In this work we build on a core idea in cognitive science, that human learned representations are universally *combinatorial*, where complex concepts are built by combining simpler ones [4, 6, 7]. Based on this principle, we propose a new approach for engineering generative AI (GenAI) systems for music production grounded in human-like, combinatorial representations of musical structure. We give a proof-of-concept implementation of such a system, as a step toward building more controllable and human-centered tools for human-AI music co-creation.

Recent influential works in psychology have modeled mental representations of the world as generative programs. A common approach to modeling how mental programs are learned from experience is based on Program Induction (PI) – the process by which, when given a set of examples, learners hypothesize a latent generative rule that could have produced them. [8, 9]. These models demonstrate an impressive power of PI to explain human mental representations across reasoning domains including spatial [10, 11], visual [8, 12], and linguistic [13] concepts. Conversely, the creative process by which people come up with new samples from a given concept class can be modeled as Program Synthesis (PS) – the generative process that recombines and mutates simple compositional elements into more complex concepts. Given that the latent hierarchical structure in music is particularly well-suited for PI-based modeling, in this work we explore computational modeling approaches for flexibly encoding a variety of symbolic musical structures, as well as examine approaches that would make inference of the underlying structural representations more tractable.

We contribute: (1) proof-of-concept representations of musical structure as Turing-complete programs, (2) a tractable framework for discovery of musical structure through leveraging LLM-based synthesis of symbolic programs

2. APPROACH

Previous work on music representations Previous work has studied the generative nature of music via Context Free Grammars (CFGs) [14]. CFGs successfully model non-local hierarchical dependencies [15], the perception of structure in Western music [16], production of musical rhythm [17], and harmony [18]. However, CFGs are not Turing-complete, cannot grow by inventing new primitives and lack the flexibility to interactively adapt to human performers.

Music as Python programs We represent musical concepts as executable Python code. Unlike CFG or statistical models, Python code can encode causal transformations, recursion, and compositional hierarchies. Programs can be executed, inspected, and debugged, making them both highly expressive and human-interpretable.

LLMs as program generators A key challenge in program induction is the curse of compositionality, where there are infinitely many possible programs that can ex-

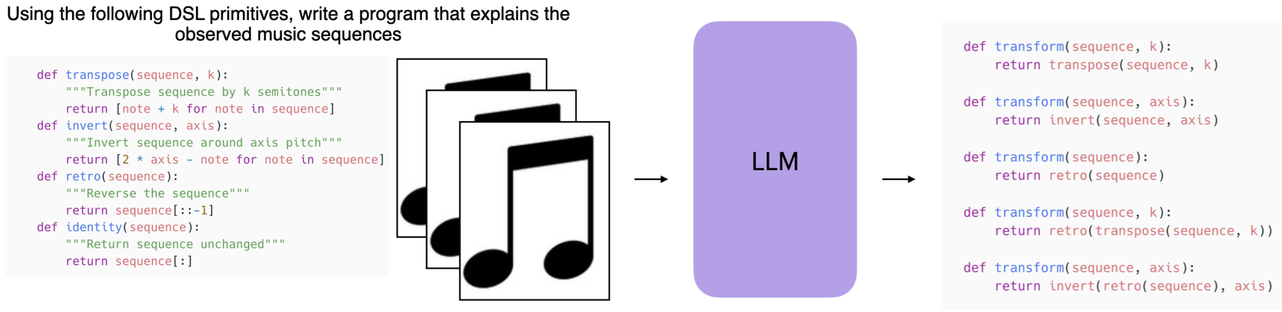

Figure 1. LLM-guided program induction for musical cognition. Given training examples and DSL constraints, LLMs generate candidate programs that are evaluated for compositional reasoning and generalization to novel sequences.

plain observed behavior. This makes searching for programs computationally costly, a naive approach of enumerative search through all possible programs that satisfy given examples is computationally intractable [8, 9].

To make inference tractable, traditional PI frameworks assume that the underlying programs are written in a pre-defined Domain-Specific Language (DSL): a small set of primitives and transformations are hand-designed to constrain the search space, while capturing the concepts in the given domain. The need to hand-craft a DSL has traditionally limited applicability of PI to natural domains, such as music, where the DSL is inherently domain-open – new concepts can be invented as the system evolves.

Recent work addresses this challenge by using LLMs as stochastic proposal generators, guiding search toward plausible programs that can then be evaluated by Bayesian inference or symbolic execution [19, 20]. We adopt a similar strategy: an LLM proposes candidate programs consistent with observed musical excerpts, while a lightweight evaluator checks their validity and likelihood.

Domain-specific language constraints To test whether representational constraints improve program discovery, we compare unconstrained Python code generation with generation restricted to a minimal musical DSL. The DSL implements four primitive operations—transpose, invert, retrograde, and identity—each composable to form more complex transformations. Importantly, primitives can be composed, producing arbitrarily deep programs that capture hierarchical transformations. We hypothesized that constraining proposals to this DSL steers the model toward compositional generalization.

3. EXPERIMENTS

We conducted a proof-of-concept experiment testing whether LLMs can discover compositional structure in musical transformations when guided by representational constraints.

We first extracted symbolic note sequences using the `music21` library [21]. Each excerpt was represented as a sequence of MIDI pitch values (ignoring dynamics, timbre, or expressive timing). From these sequences we constructed input-output training pairs, where the output was derived from the input via a simple, musically interpretable transformation (e.g., transposition by n semitones, inversion about a reference pitch, retrograde reversal). These transformations instantiate examples of latent rules that human practitioners manipulate when producing music.

The task for the model was then: given a handful of input-output pairs, infer a program that explains the transformation. We compared two prompting techniques for LLM-based synthesis:

Freeform Python Generation: LLMs were allowed to produce arbitrary Python code. This setting provides maximal flexibility, with access to loops, conditionals, arithmetic, and even hardcoded lookup tables. However, such flexibility opens the door to degenerate "shortcut" solutions that fit training examples without revealing the underlying structure.

DSL-Constrained Generation: LLMs were restricted to our musical DSL. This forces the model to compose transformations from interpretable primitives. This approach introduces elements of traditional DSL-based program synthesis into LLM-based code generation, as a heuristic that constrains the program search space.

Both conditions were evaluated on their ability to generalize transformations to novel inputs (unseen excerpts drawn from the same underlying composition).

DSL-constrained generation achieved higher rates of generalization compared to freeform Python. Qualitative analysis of the generated programs shows unconstrained programs often defaulted to hardcoded memorization, while constrained generation discovered general transformation rules.

4. DISCUSSION

This work offers a new paradigm for computational cognitive science in creative domains. By combining neural generation with symbolic constraints and Bayesian reasoning, we aim to build AI systems that learn like humans: through interpretable, compositional programs that can be understood, debugged, and collaboratively extended. We argue that by building Gen AI grounded in human cognition, we take a step toward engineering systems that empower human creativity through interpretable, steerable tools that reason about music in fundamentally human-like ways, rather than replacing humans with black box AI.

5. REFERENCES

[1] M. Anglada-Tort, P. Harrison, and N. Jacoby, "Studying the effect of oral transmission on melodic structure using online iterated singing," in *Proceedings of the Annual Meeting of the Cognitive Science Society, 44 (44)*, 2022.

[2] M. Anglada-Tort, P. M. Harrison, H. Lee, and N. Jacoby, "Large-scale iterated singing experiments reveal oral transmission mechanisms underlying music evolution," *Current Biology*, vol. 33, no. 8, pp. 1472–1486, 2023.

[3] M. Lumaca and G. Baggio, "Cultural transmission and evolution of melodic structures in multi-generational signaling games," *Artificial Life*, vol. 23, no. 3, pp. 406–423, 2017.

[4] J. A. Fodor, *The language of thought*. Harvard university press, 1975, vol. 5.

[5] E. Schulz, J. B. Tenenbaum, D. Duvenaud, M. Speekenbrink, and S. J. Gershman, "Compositional inductive biases in function learning," *Cognitive Psychology*, vol. 99, pp. 44–79, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010028517301743

[6] B. M. Lake and S. T. Piantadosi, "People infer recursive visual concepts from just a few examples," *Computational Brain & Behavior*, vol. 3, no. 1, pp. 54–65, 2020.

[7] B. Pitt, S. Ferrigno, J. F. Cantlon, D. Casasanto, E. Gibson, and S. T. Piantadosi, "Spatial concepts of number, size, and time in an indigenous culture," *Science Advances*, vol. 7, no. 33, p. eabg4141, 2021.

[8] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum, "Human-level concept learning through probabilistic program induction," *Science*, vol. 350, no. 6266, pp. 1332–1338, 2015.

[9] K. Ellis, C. Wong, M. Nye, M. Sablé-Meyer, L. Morales, L. Hewitt, L. Cary, A. Solar-Lezama, and J. B. Tenenbaum, "Dreamcoder: Bootstrapping inductive program synthesis with wake-sleep library learning," in *Proceedings of the 42nd acm sigplan international conference on programming language design and implementation*, 2021, pp. 835–850.

[10] M. Kryven, C. Wyeth, A. Curtis, and K. Ellis, "Cognitive maps are generative programs," *Cognitive Sciences Society*, 2025.

[11] S. Sharma, A. Curtis, M. Kryven, J. Tenenbaum, and I. Fiete, "Map induction: Compositional spatial submap learning for efficient exploration in novel environments," *International Conference of Learning Representations*, 2022.

[12] L. Tian, K. Ellis, M. Kryven, and J. Tenenbaum, "Learning abstract structure for drawing by efficient motor program induction," *Advances in Neural Information Processing Systems*, vol. 33, pp. 2686–2697, 2020.

[13] T. Verhoef, S. Kirby, and B. De Boer, "Emergence of combinatorial structure and economy through iterated learning with continuous acoustic signals," *Journal of Phonetics*, vol. 43, pp. 57–68, 2014.

[14] M. Rohrmeier, "Towards a generative syntax of tonal harmony," *Journal of Mathematics and Music*, vol. 5, no. 1, pp. 35–53, 2011. [Online]. Available: https://doi.org/10.1080/17459737.2011.573676

[15] S. Koelsch, M. Rohrmeier, R. Torrecuso, and S. Jentschke, "Processing of hierarchical syntactic structure in music," *Proceedings of the National Academy of Sciences*, vol. 110, no. 38, pp. 15 443–15 448, 2013.

[16] Z. Ren, X. Guan, and M. Rohrmeier, "A computational cognitive model for processing repetitions of hierarchical relations," *arXiv preprint arXiv:2504.10065*, 2025.

[17] M. Rohrmeier, "Towards a formalization of musical rhythm." in *ISMIR*, 2020, pp. 621–629.

[18] M. Rohrmeier, W. Zuidema, G. A. Wiggins, and C. Scharff, "Principles of structure building in music, language and animal song," *Philosophical transactions of the Royal Society B: Biological sciences*, vol. 370, no. 1664, 2015.

[19] H. Tang, D. Key, and K. Ellis, "Worldcoder, a model-based llm agent: Building world models by writing code and interacting with the environment," 2024. [Online]. Available: https://arxiv.org/abs/2402.12275

[20] W. T. Piriyakulkij, Y. Liang, H. Tang, A. Weller, M. Kryven, and K. Ellis, "Poe-world: Compositional world modeling with products of programmatic experts," 2025. [Online]. Available: https://arxiv.org/abs/2505.10819

[21] M. S. Cuthbert and C. Ariza, "Music21: A toolkit for computer-aided musicology and symbolic music data." in *ISMIR*, J. S. Downie and R. C. Veltkamp, Eds. International Society for Music Information Retrieval, 2010, pp. 637–642. [Online]. Available: http://dblp.uni-trier.de/db/conf/ismir/ismir2010.html#CuthbertA10