Generating Domain-Specific Programs for Diagram Authoring with Large Language Models

Rijul Jain
rijul.jain@williams.edu
Williams College
USA

Wode Ni
woden@cs.cmu.edu
Carnegie Mellon University
USA

Joshua Sunshine
sunshine@cs.cmu.edu
Carnegie Mellon University
USA

Abstract

Large language models (LLMs) can generate programs in general-purpose languages from prose descriptions, but are not trained on many domain-specific languages (DSLs). Diagram authoring with Penrose, a diagramming system using three DSLs, exemplifies the utility of DSL program generation with LLMs, which enables diagram creation from prose. We provide methods to conceptualize and evaluate the structures of one-shot LLM prompts to generate error-free DSL programs and implement Penrose diagram creation from prose using LLMs. We will evaluate our LLM prompt structures by testing prompt variations across different diagramming domains and plan to run a user study to assess the ease of LLM-augmented Penrose diagramming over other tools.

CCS Concepts: • **Human-centered computing → Visualization systems and tools**.

Keywords: domain-specific languages, large language models, visualization

ACM Reference Format:
Rijul Jain, Wode Ni, and Joshua Sunshine. 2023. Generating Domain-Specific Programs for Diagram Authoring with Large Language Models. In *Companion Proceedings of the 2023 ACM SIGPLAN International Conference on Systems, Programming, Languages, and Applications: Software for Humanity (SPLASH Companion '23), October 22–27, 2023, Cascais, Portugal.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3618305.3623612

1 Problem and Motivation

Existing tools for authoring diagrams, from TikZ to Adobe Illustrator to Penrose [3], a system which uses three domain-specific languages (DSLs) to encode a high-level specification for its diagrams, all pose a learning curve to some extent.

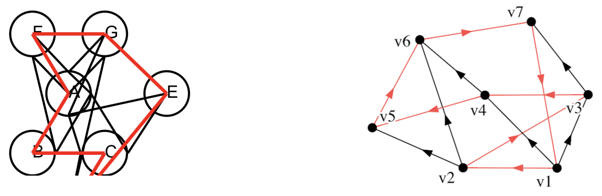


Figure 1. For a prose instruction, GPT-4 consistently fails to generate a legible SVG (left), and successfully generates high-quality diagrams in Penrose (right)

Since large language models (LLMs) can generate computer programs from natural language prompts, using LLMs to create diagrams by generating the corresponding programs in DSLs for graphics (e.g. SVG) from prose descriptions of desired diagrams has the potential to make diagram authoring accessible to a wider audience by greatly diminishing this learning curve. New diagrammers would be able to get started more quickly and see visual feedback more immediately than without a prose-to-diagram feature. However, as shown at left in Figure 1, which shows the resulting diagram when GPT-4 is asked to generate an SVG depicting "a Hamiltonian graph with 7 nodes and 13 edges with the cycle highlighted," LLMs do not perform well at producing relatively low-level text which translates directly to visuals, such as SVG and TikZ code. By contrast, with the methods we describe in this work, GPT-4 generates code in Penrose's higher-level Substance DSL that results in the Penrose diagram at right in Figure 1, which is correct and legible.

The main challenge with DSL program generation with LLMs is that while LLMs can often generate programs in general-purpose languages with zero-shot prompts containing only prose descriptions of the desired program, they are ill-equipped to generate programs in this way for many DSLs on which they may not be well trained, such as Substance. To generate DSL programs that will compile and that reflect the desired outcome using LLMs, it is necessary to provide thorough information to the model to "teach" it a DSL.

In this work, therefore, we introduce methods for conceptualizing the structures of one-shot LLM prompts for DSL program generation and evaluating prompt variations with GPT-4 to find those that best produce error-free DSL programs matching the prose descriptions; we also integrate

GPT-4 with Edgeworth [1], a diagrammatic problem authoring tool using Penrose, to introduce prose-to-diagram capabilities for users. We intend for the results of this work to illuminate the potential efficacy of LLM-augmented diagram authoring over other tools and best practices for DSL program generation with LLMs more generally.

2 Methods

Our prompt structures aim to effectively teach LLMs a new language for the end goal of diagram authoring; to this end, we pursue one-shot LLM prompts to minimize inference time, ensuring that a prose-to-diagram method would be practical and efficient to use. We conceptualize the possible elements of the structure of a one-shot prompt for DSL program generation as shown in the following section; variations in prompt structure will involve including, omitting, or modifying any of these elements.

2.1 Elements of the Prompt Structure

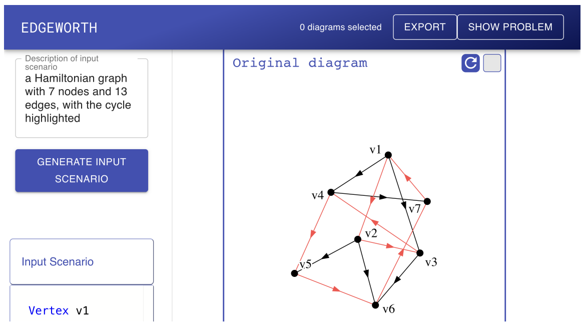
The first element consists of output instructions, which can contain a role instruction, i.e. "You are a code generator...", as well as instructions on comment syntax and solely outputting code; we vary this element by including or omitting either or both portions. Then, we provide a description of the DSL, which we either include or omit; for Substance, we explain Penrose and Substance's function within it. Next, we give a formal DSL specification: here, either a specialized BNF-style formal grammar [2] for the Substance language or a schema program written in Penrose's Domain DSL, both particular to the diagramming domain (i.e. graphs or geometry). We include either the grammar or the schema, either annotated or not, or omit both. For example, for a rule of a grammar such as `tname ::= "Vertex"`, we add a comment with a description and sample usage:

```
    // This type describes a vertex.
    // Example usage: 'Vertex v1, v2, v3'
```

Then, we give a standalone sample program demonstrating usage of all the types, functions, and predicates in the DSL; it is either included, commented or not, or omitted. Next, the prose description, which we modify by providing thorough, low-level prose or terse, high-level prose, describes the desired diagram. At the end, the final output instructions, which we either include or omit, repeat the comment syntax and only-code initial output instructions.

3 Evaluation

We performed initial tests on four variations with two different annotated formal DSL specifications (BNF-style grammar and Domain schema), either including or omitting standalone sample programs with all other prompt elements constant, where each variation was evaluated on 54 total programs generated from 18 unique prompts with different prose descriptions across 3 domains. The BNF-without-sample variation had a 96.29% compile rate while BNF-with-sample achieved 79.62%. Domain-with-sample had an 85.18% compile rate while Domain-without-sample achieved 68.51% (data here). We will further evaluate our LLM prompt structure by testing all variations in prompt structure on compile rate, diagram correctness, and inference time across diagramming domains (geometry, graphs, chemistry) and prose descriptions. We will test Substance generation with LLMs against LLM generation of other diagramming DSLs (SVG and TikZ).



a Hamiltonian graph with 7 nodes and 13 edges, with the cycle highlighted

```
Vertex v1
```

Figure 2. LLM-augmented Edgeworth UI

Figure 2 shows Edgeworth augmented with a prose-to-diagram feature using the BNF-without-sample prompt variation. We will evaluate the efficacy of LLM-augmented diagram authoring by designing and running a user study to examine the speed, ease of use, feature richness, and diagram quality in the user experience provided by Edgeworth versus that of other tools (Illustrator and Google Drawings).

4 Conclusion

We have introduced methods for one-shot DSL program generation for diagram authoring with LLMs and integrated LLMs with the diagrammatic problem authoring tool Edgeworth. Evaluating structural variations in LLM prompts will provide insight for DSL program generation in general, while enabling prose-to-diagram capabilities with Penrose is a step towards democratizing diagram authoring.

References

[1] Hwei-Shin Harriman. 2021. Edgeworth: Authoring Diagrammatic Math Problems Using Program Mutation. In *Companion Proceedings of the 2021 ACM SIGPLAN International Conference on Systems, Programming, Languages, and Applications: Software for Humanity* (Chicago, IL, USA) *(SPLASH Companion 2021)*. Association for Computing Machinery, New York, NY, USA, 22–24. https://doi.org/10.1145/3484271.3484978

[2] Bailin Wang, Zi Wang, Xuezhi Wang, Yuan Cao, Rif A. Saurous, and Yoon Kim. 2023. Grammar Prompting for Domain-Specific Language Generation with Large Language Models. arXiv:2305.19234 [cs.CL]

[3] Katherine Ye, Wode Ni, Max Krieger, Dor Ma'ayan, Jenna Wise, Jonathan Aldrich, Joshua Sunshine, and Keenan Crane. 2020. Penrose: From Mathematical Notation to Beautiful Diagrams. *ACM Trans. Graph.* 39, 4, Article 144 (aug 2020), 16 pages. https://doi.org/10.1145/3386569.3392375

Received 2023-08-15; accepted 2023-08-30