Consistent Graph Model Generation with Large Language Models

Boqi Chen*
Electrical and Computer Engineering
McGill University
Montreal, Canada

***Abstract*—Graph model generation from natural language requirements is an essential task in software engineering, for which large language models (LLMs) have become increasingly popular. A key challenge is ensuring that the generated graph models are *consistent* with domain-specific well-formed constraints. LLM-generated graphs are often *partially correct* due to *inconsistency* with the constraints, limiting their practical usage. To address this, we propose a *novel abstraction-concretization* framework motivated by *self-consistency* for generating consistent models. Our approach first abstracts candidate models into a probabilistic partial model and then concretizes this abstraction into a consistent graph model. Preliminary evaluations on taxonomy generation demonstrate that our method significantly enhances both the consistency and quality of generated graph models.**

***Index Terms*—large language models, graph model generation, constraint optimization**

I. Introduction

Motivation. Graph models are an essential concept in software engineering (SE), used to represent core system concepts and their relationships. These models are often derived from natural language requirement specifications. Once constructed, they can enable many downstream applications, including model-based testing and automated code generation. Manually creating these models from requirements is often time-consuming and costly. To address this challenge, automated graph model generation aims to reduce the need for extensive manual effort. Traditional automated model generation approaches typically rely solely on formal specifications as input and are unable to process natural language requirements [1]. Recently, LLMs have gained increased interest in this task due to their impressive natural language understanding capability [2]. Still, these approaches remain exploratory and rely solely on LLMs during the generation process.

Problem. Different from typical LLM outputs, graph model generation requires LLMs to output textual representations of the underlying *graph*. This difference introduces *two challenges* for LLMs in this task: (1) *syntactic correctness* and *semantic consistency* with some well-formedness constraints over the metamodel [1] and (2) the generated graph models can be *partially* correct and contain hallucinated elements.

Background and related work. The syntax of LLM's output can typically be tackled with a filtering mechanism or constrained decoding [3]. The consistency of generated models and their impact on model quality has not yet been systematically investigated. Previously, consistent graph generation has been explored for other machine learning techniques [4].

Self-consistency improves LLM output quality by *selecting* the most frequent answer [5]. The core assumption is that if LLMs can solve a task, they should output the correct answer more often when asked repeatedly. Yet, graphs can be partially correct, violating such an assumption. The uncertainty in graph models can typically be represented using a partial model [6]. Such partial models have been used in traditional approaches for consistent graph model generation [1]. However, these methods rely solely on formal specifications and fail to account for natural language requirements.

* Work partially done during an internship at the Huawei Waterloo Research Center.

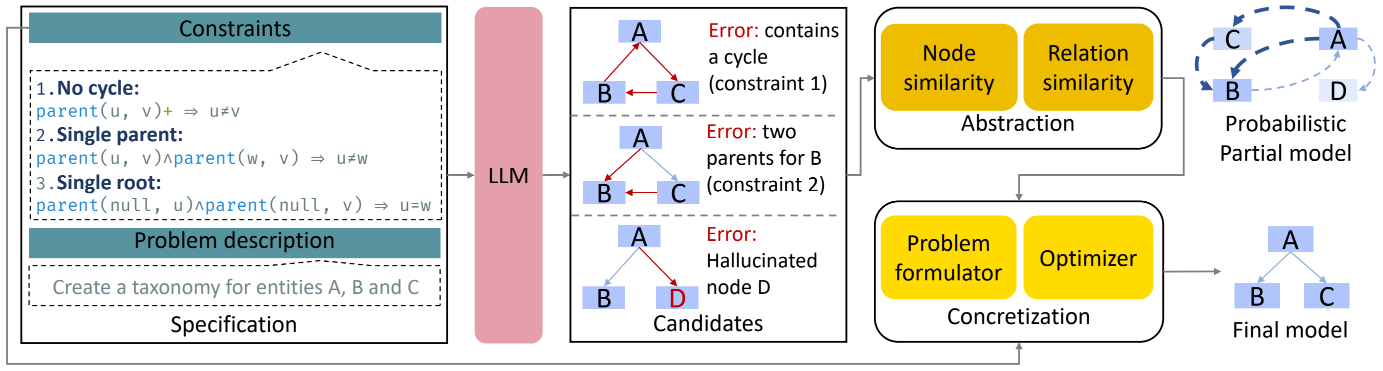
Constraints
1. No cycle:
parent(u, v)+ ⇒ u≠v
2. Single parent:
parent(u, v)∧parent(w, v) ⇒ u≠w
3. Single root:
parent(null, u)∧parent(null, v) ⇒ u=w
Problem description
Create a taxonomy for entities A, B and C
Specification
LLM
A
B
C
Error: contains a cycle (constraint 1)
Error: two parents for B (constraint 2)
D
Error: Hallucinated node D
Candidates
Node similarity
Relation similarity
Abstraction
Probabilistic Partial model
Problem formulator
Optimizer
Concretization
Final model

Fig. 1. Overview of the approach with the taxonomy generation example. Red elements represent errors in candidates. Color depths and weights represent probabilities in the partial model. Even if each individual candidate contains errors, the approach derives the correct model.

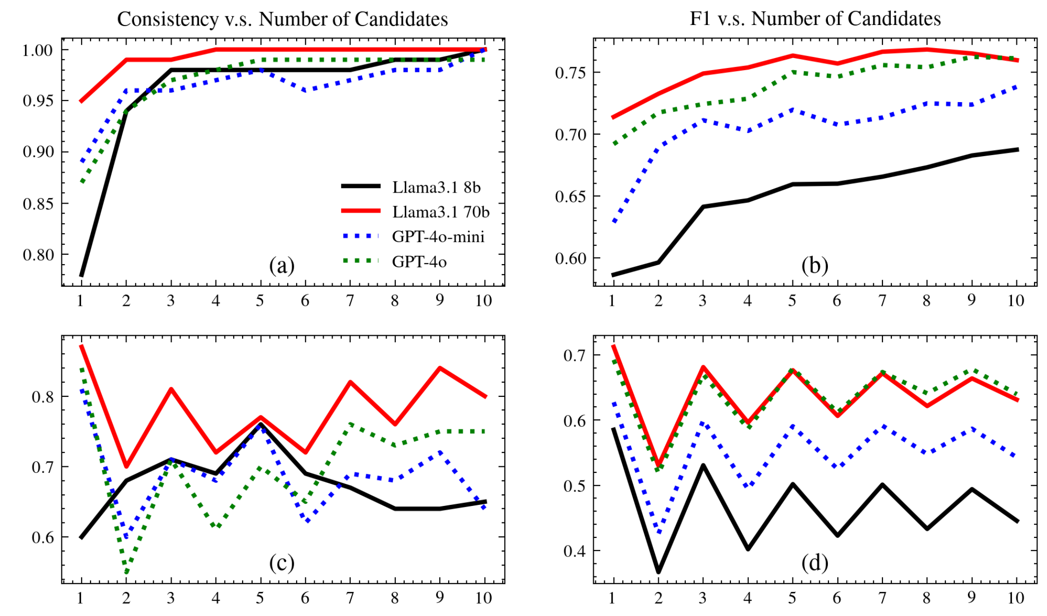

Fig. 2. The trend of metric values by increasing the number of candidates for concretization with constraints (a, b) and without constraints (c, d)

II. Approach and Uniqueness

We are the **first to tackle both challenges** of LLMs in automated graph model generation by proposing a *novel abstraction-concretization* framework motivated by self-consistency that guarantees graph consistency. The core idea of the approach is to leverage multiple outputs from LLMs. By sampling multiple outputs from LLMs, we construct a *probabilistic partial model* that captures the likelihood of the generated models. This partial model is then concretized into the most likely consistent model.

Figure 1 shows an **overview of the approach** and an example. Given a specification, including problem descriptions and constraints, the LLM first generates *k candidates*. In the example, due to inherent limitations, the LLM generates two partially correct solutions and a hallucinated solution.

Abstraction incrementally adds candidates to a *probabilistic partial model* using graph similarity guided by (1) node similarity measured using embedding on node attributes and (2) relation similarity measured by graph edit distance. The resulting partial model consists of all candidates while each element is associated with a probability representing how likely it appears in one of the candidates.

Concretization searches for the most promising model from the partial model using a *constraint optimization problem*. It selects edges and nodes from the partial model, providing constraints in logic form. With the Naive Bayes assumption (probabilities of nodes and edges are independent), the objective is set to maximize cross-entropy. The final model is given by solving this optimization with a constraint optimizer, which provides consistency guarantee if the problem is solvable.

III. Evaluation

Setup. We evaluate this approach on the WordNet[1] taxonomy generation using LLMs with different architectures and sizes. Taxonomies are models representing parental relations between concepts. We evaluate the generated taxonomies using F1 scores over ancestral relations and their consistency with constraints. We randomly sample 3 taxonomies as few-shot examples and 100 taxonomies for evaluation. We compare concretization with and without constraints using the CBC optimizer[2].

Results and contribution. *With 10 candidates, our approach enables three LLMs to achieve full consistency* as shown in Figure 2(a). This is not achieved even by the best-performing LLM with a single output. *The improvement in consistency also significantly benefits the model quality*. As shown in Figure 2(b), with constraints during concretization, F1 scores of all LLMs generally increase as the candidate number increases, especially in weaker LLMs. Notably, the F1 score of GPT-4o-mini with 10 candidates beats the best-performing model Llama3.1 70b with a single output. By contrast, Figure 2 (c) and (d) show *these benefits disappear when constraints are not considered*, highlighting the importance of constraints.

References

[1] O. Semeráth, A. S. Nagy, and D. Varró, "A graph solver for the automated generation of consistent domain-specific models," in *ICSE*, 2018.

[2] B. Chen, F. Yi, and D. Varró, "Prompting or fine-tuning? a comparative study of large language models for taxonomy construction," in *MODELS Companion*, 2023.

[3] L. Netz, J. Reimer, and B. Rumpe, "Using grammar masking to ensure syntactic validity in llm-based modeling tasks," in *MODELS Companion*, 2024.

[4] B. Chen *et al.*, "Consistent scene graph generation by constraint optimization," in *ASE*, 2022.

[5] X. Wang *et al.*, "Self-consistency improves chain of thought reasoning in language models," in *ICLR 2023*.

[6] M. Famelis, R. Salay, and M. Chechik, "Partial models: Towards modeling and reasoning with uncertainty," in *ICSE*, 2012.

[1]https://wordnet.princeton.edu/

[2]https://github.com/coin-or/Cbc