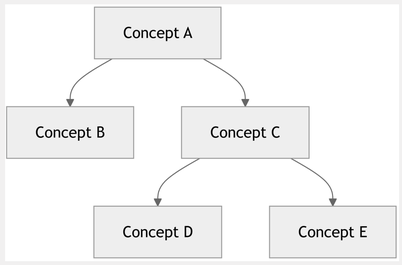
Consistent Graph Model Generation with Large Language Models

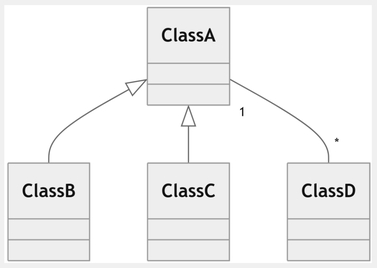
Boqi (Percy) Chen
boqi.chen@mail.mcgill.ca
McGill University, Montreal, Canada

1. Graph Model Generation

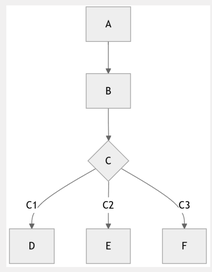
Software engineering processes, such as requirements engineering, model-based testing, and code generation, involve many different types of graph models



Taxonomies

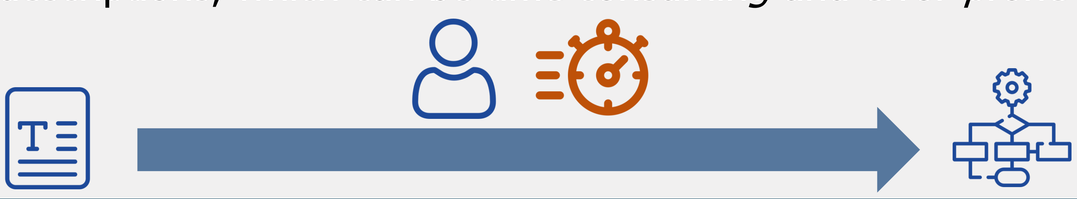


Domain models



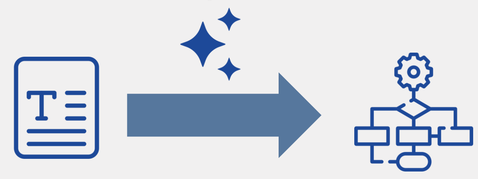
Flowcharts

Traditionally, these models are created manually by engineers from textual requirement descriptions, which can be *time consuming* and *error-prone*



2. LLMs for Graph Model Generation

LLMs enable fast automated graph model generation directly from textual descriptions

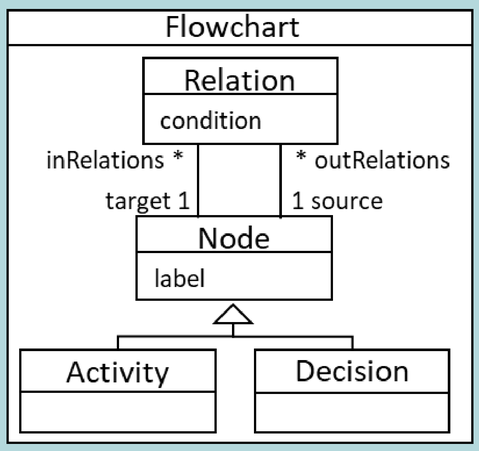


However, graph models generated from LLMs may contain several potential **issues**

- **Syntax error**
- **Inconsistency**
- **Inaccuracy**

3. Improving LLMs for Model Generation with Self-Consistency, Partial Models, and Constraints

Metamodel

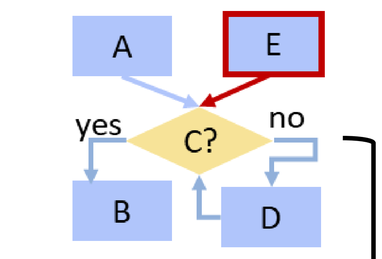


Constraints

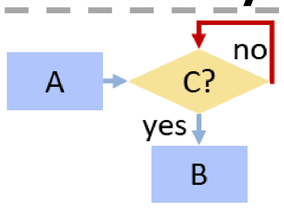
1. There is only one starting node
2. There is no self-loop
3. Decision nodes must have conditions

First perform A. After that, check condition C. While the condition is no, perform activity D, otherwise perform activity B and finish.

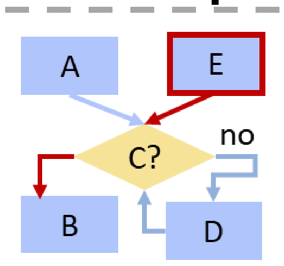
1.Generate *n* candidates



Extra activity E



Self loop

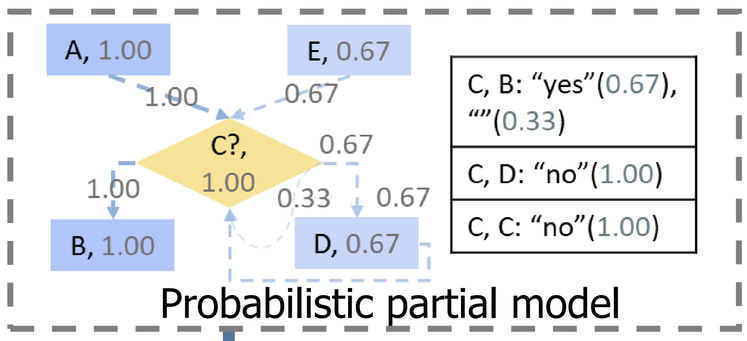


No condition

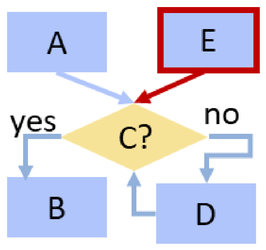
Filtered out

Syntax error

2.Abstraction with node similarity and graph matching



Probabilistic partial model



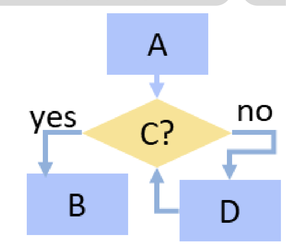
Maximum likelihood (Naïve) concretization ✗

3. Constraint translation

```
1. ∀Node u,v;!inRelations(u,*)∧
   !inRelations(v,*)⇒u=v
2. ∀Node u,v; Next(u,v)⇒u≠v
3. ∀Relation r,Decision d;
outRelations(d,r)⇒r.condition≠""
```

4. Problem formulation

5. Optimization



Constraint-aware concretization ✓

4. Evaluation

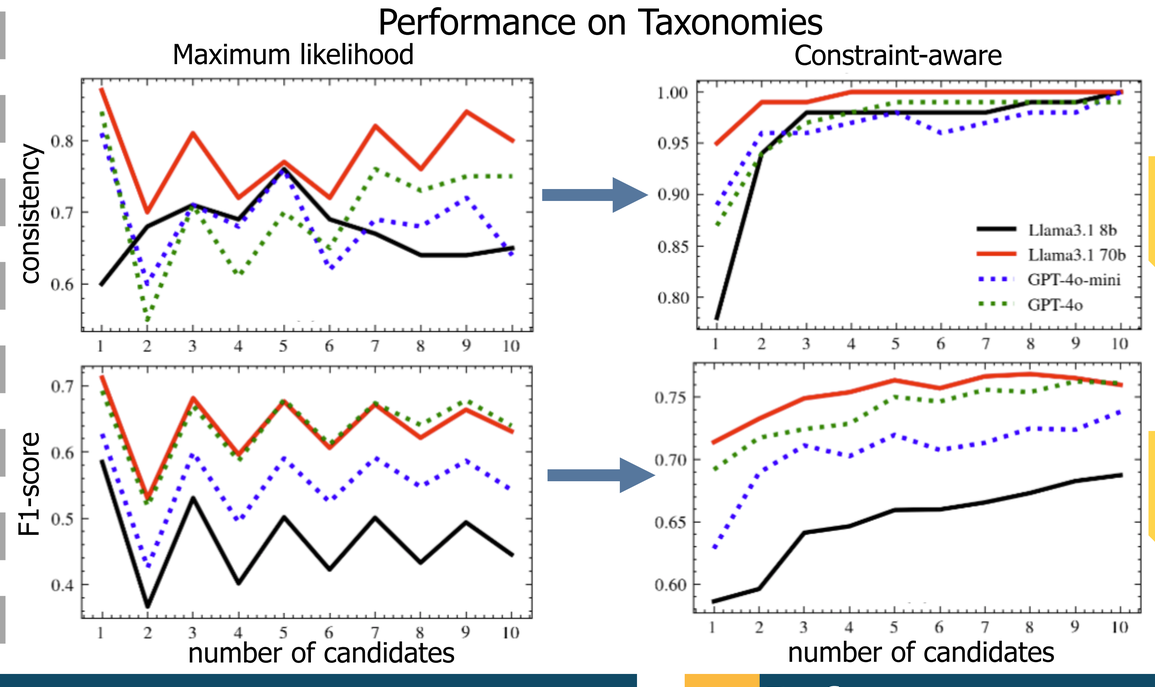
Evaluated use cases

Taxonomy
- WordNet dataset
- 100 taxonomies
- Parent-child relations

Flowchart
- PAGED dataset
- 300 procedure flowcharts

Evaluation metrics:
- **Consistency** ratio
- F1-score (**model accuracy**)

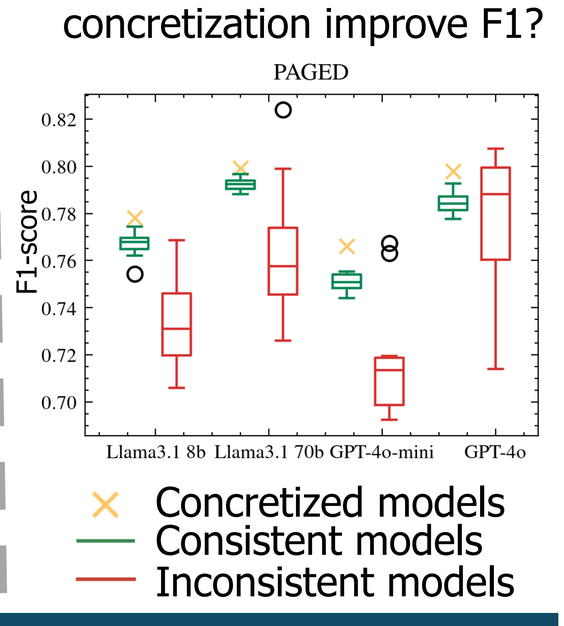
Performance on Taxonomies



Near perfect consistency

F1 scores improve significantly

Why does constraint-aware concretization improve F1?



5. Conclusion

Constraint aware self-consistency serves as a *test-time compute* method to significantly improve the model quality

Constraint aware concretization ➡ Improved consistency

Consistency in the output model ➡ Improved model accuracy

6. References

[1] Famelis, Michalis, et al. "Partial Models: Towards Modeling and Reasoning with Uncertainty." ICSE 2012.
[2] Semeráth, Oszkár, et al. "A Graph Solver for the Automated Generation of Consistent Domain-Specific Models." ICSE 2018.
[3] Chen, Boqi, et al. "Consistent Scene Graph Generation by Constraint Optimization." ASE 2022.
[4] Wang, Xuezhi, et al. "Self-Consistency Improves Chain of Thought Reasoning in Language Models." ICLR 2023.