Generating code from natural language using Abstract Syntax Graphs

Alexandru-Gabriel Sîrbu

Babeș-Bolyai University

WeADL 2024 Workshop

The workshop is organized by the Machine Learning research group (www.cs.ubbcluj.ro/ml) and the Romanian Meteorological Administration (https://www.meteoromania.ro/)

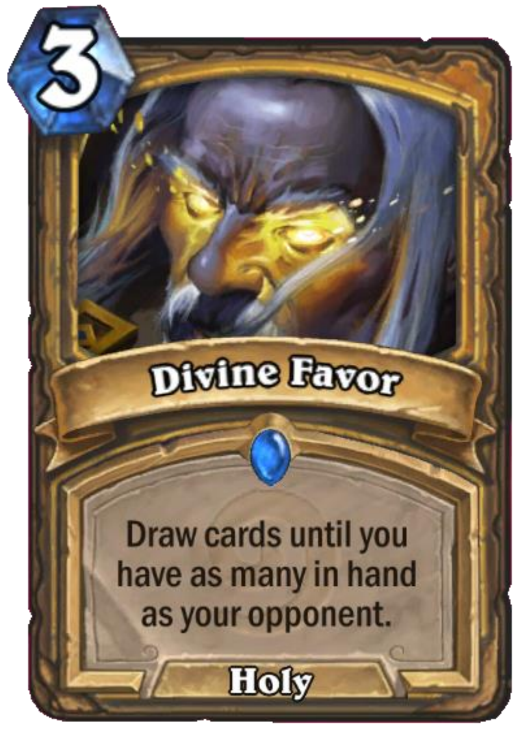
Machine Learning Research Group

MLyRE

METEO ROMANIA

Table of contents

Task presentation



Draw cards until you have as many in hand as your opponent.

Natural language text

```
class DivineFavor(SpellCard):
    def __init__(self):
        super().__init__("Divine Favor", 3, 
CHARACTER_CLASS.PALADIN, CARD_RARITY.RARE)

    def use(self, player, game):
        super().use(player, game)
        difference = len(game.other_player.hand) - 
len(player.hand)
        for i in range(0, difference):
            player.draw()
```

Code

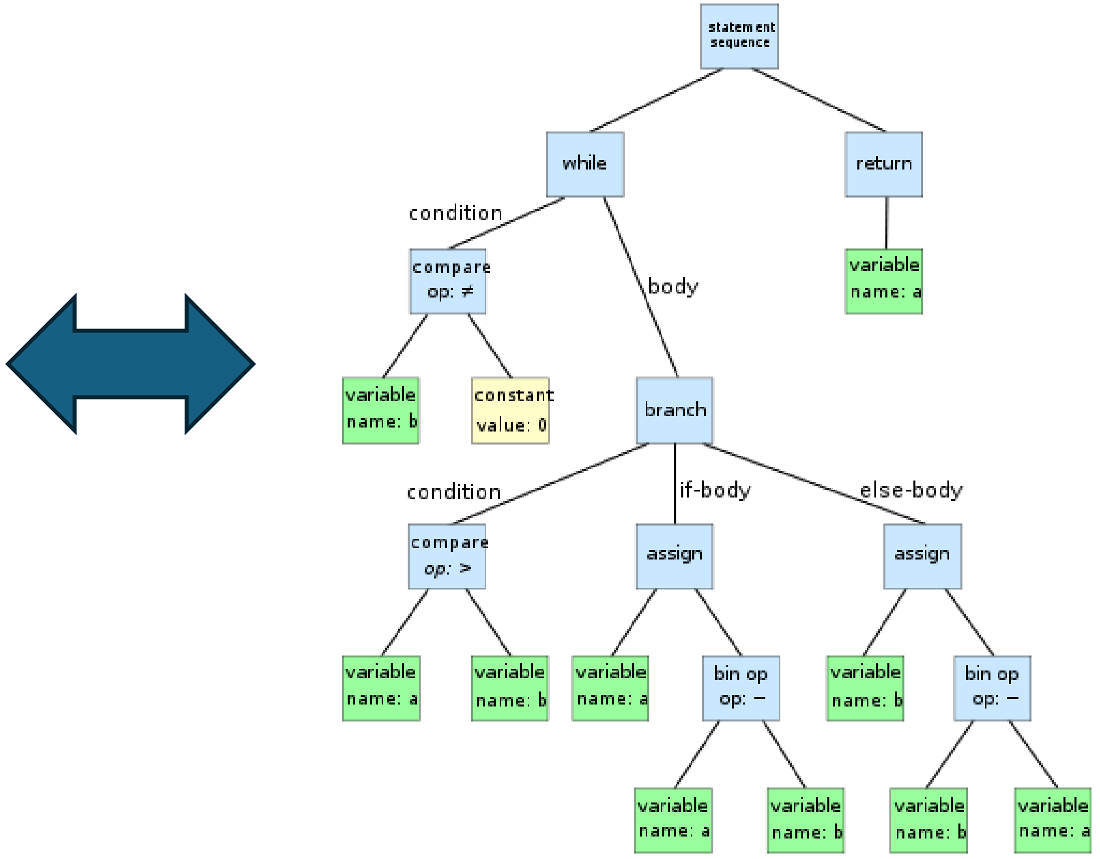
Task presentation

- Similar to Machine Translation
- Based on Sequence-to-Sequence models
- Evaluation: Token Accuracy and BLEU

Abstract Syntax Tree-based approach

```
while b ≠ 0:
    if a > b:
        a := a - b
    else:
        b := b - a
return a
```

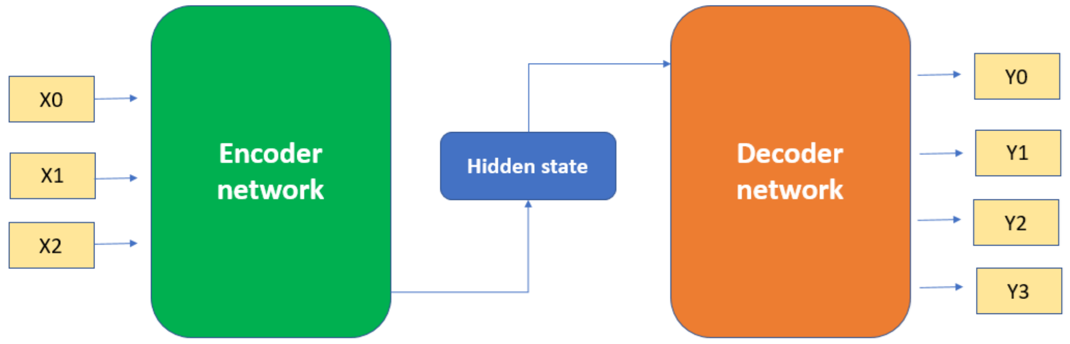
Code

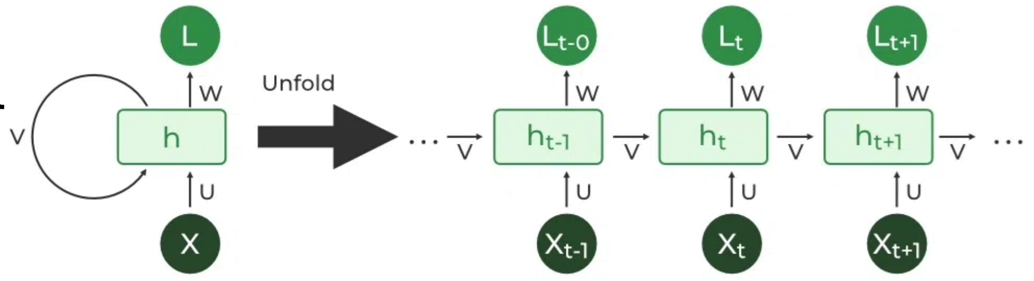


Abstract Syntax Tree

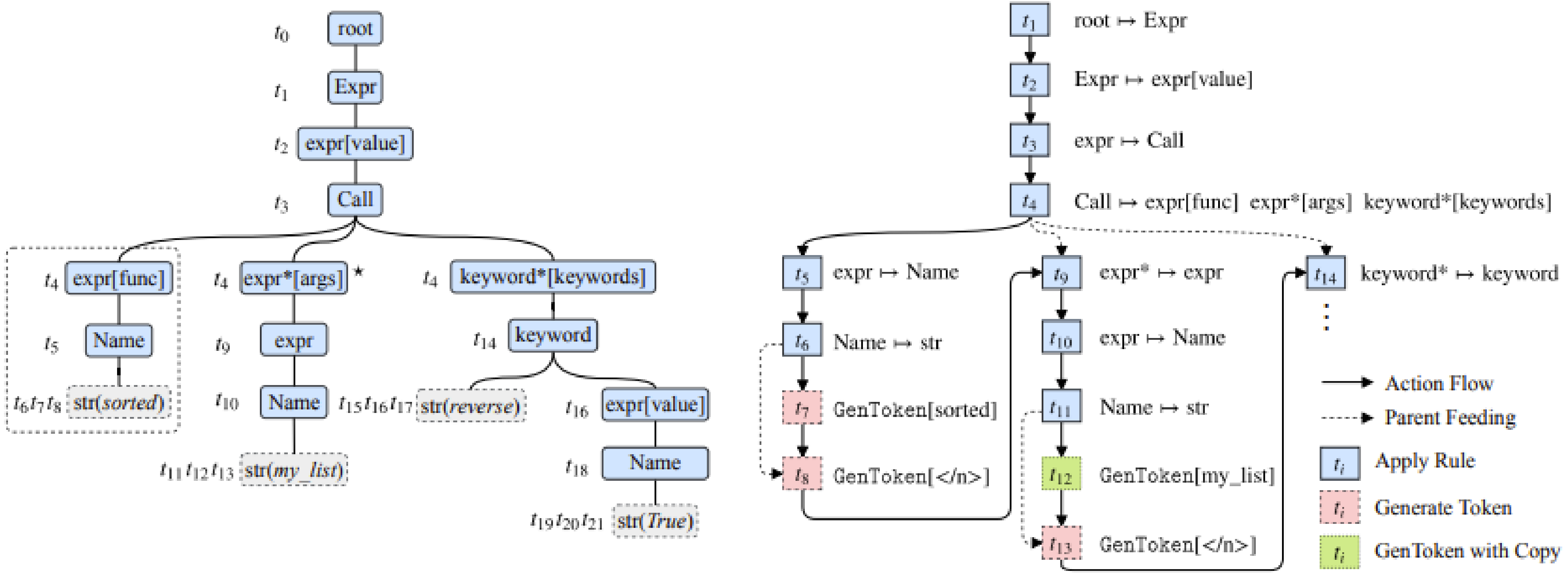
Abstract Syntax Tree-based approach

- AST = Code
- Generate the AST
- No syntax errors in generated code
- Current generated node depends on:
  - Description – initial problem
  - State – base model
  - Previously generated node – base model
  - Parent node – to generate a tree





Abstract Syntax Tree-based approach



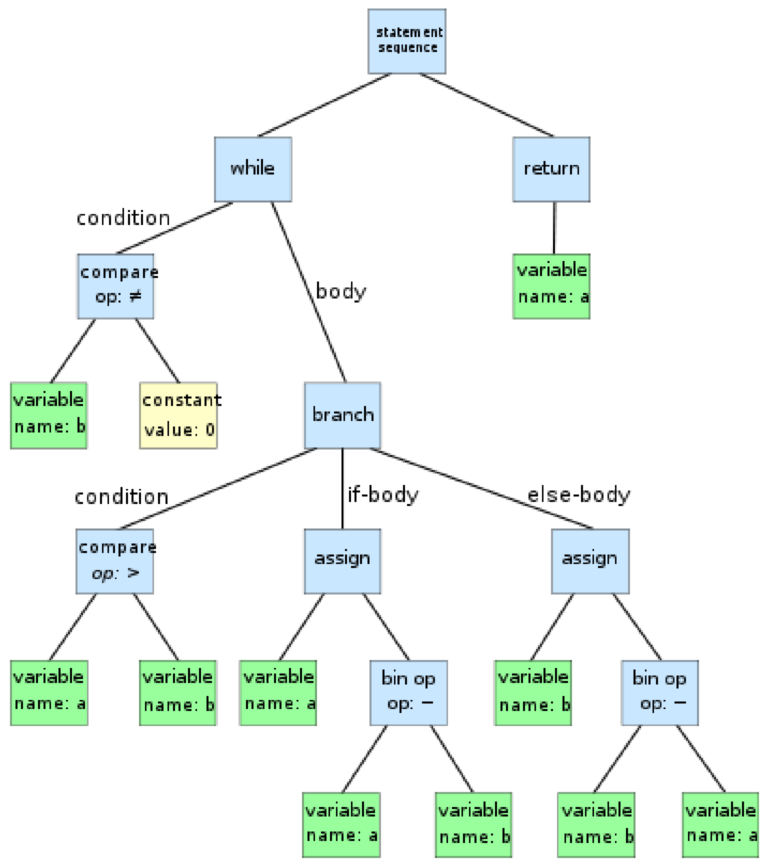
(a)

Input: `sort my_list in descending order`

(b)

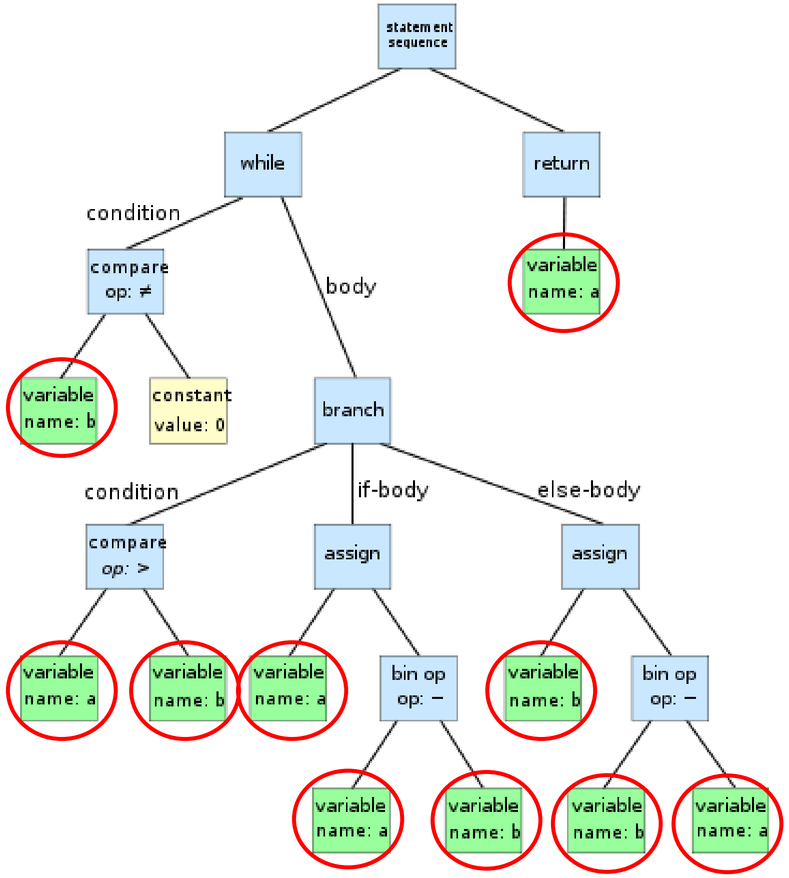
Code: `sorted(my_list, reverse=True)`

Abstract Syntax Tree-based approach



- Generate in DFS order
  - statement sequence
  - [LIST_START]
  - while
  - Compare op: ≠
  - variable name: b
  - constant value: 0
  - [LIST_START]
  - branch
  - …
  - return
  - variable name: a
  - [LIST_END]

Weaknesses



- More nodes generated => less accurate
- Redundancies in leaves generation
- One leaf miss-generated = different semantical meaning

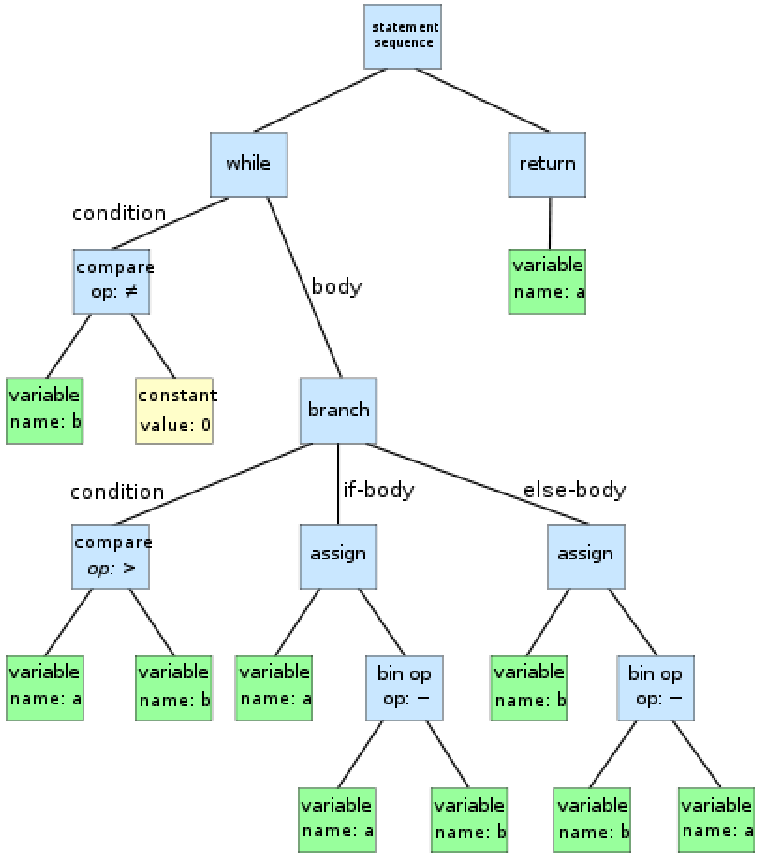
Original contribution

- Abstract Syntax Graph concept
- Graph Neural Networks for code generation task
- Generate the code as an Abstract Syntax ~~Tree~~ Graph

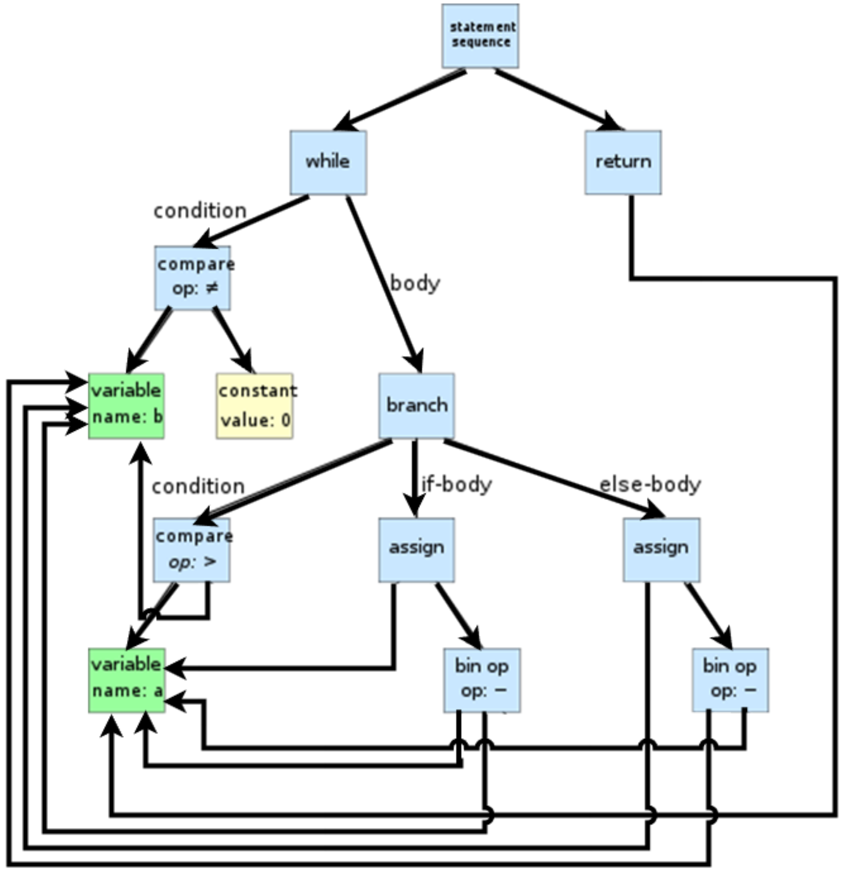
Abstract Syntax Graph

- Similar to Abstract Syntax Tree
- Incorporates “system memory” idea for variables and their values
- Loops in the initial tree by creating unique leaves
- Generate it using a Graph Recurrent Neural Network

Abstract Syntax Graph



AST



ASG

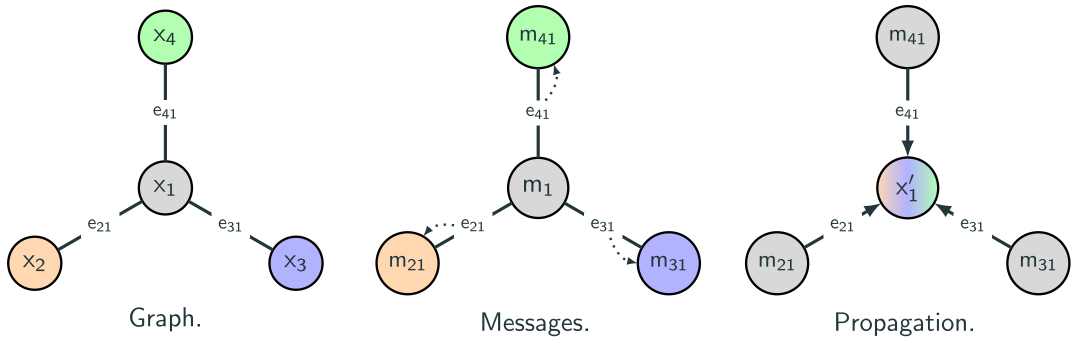
Abstract Syntax Graph

Abstract Syntax Tree to Abstract Syntax Graph

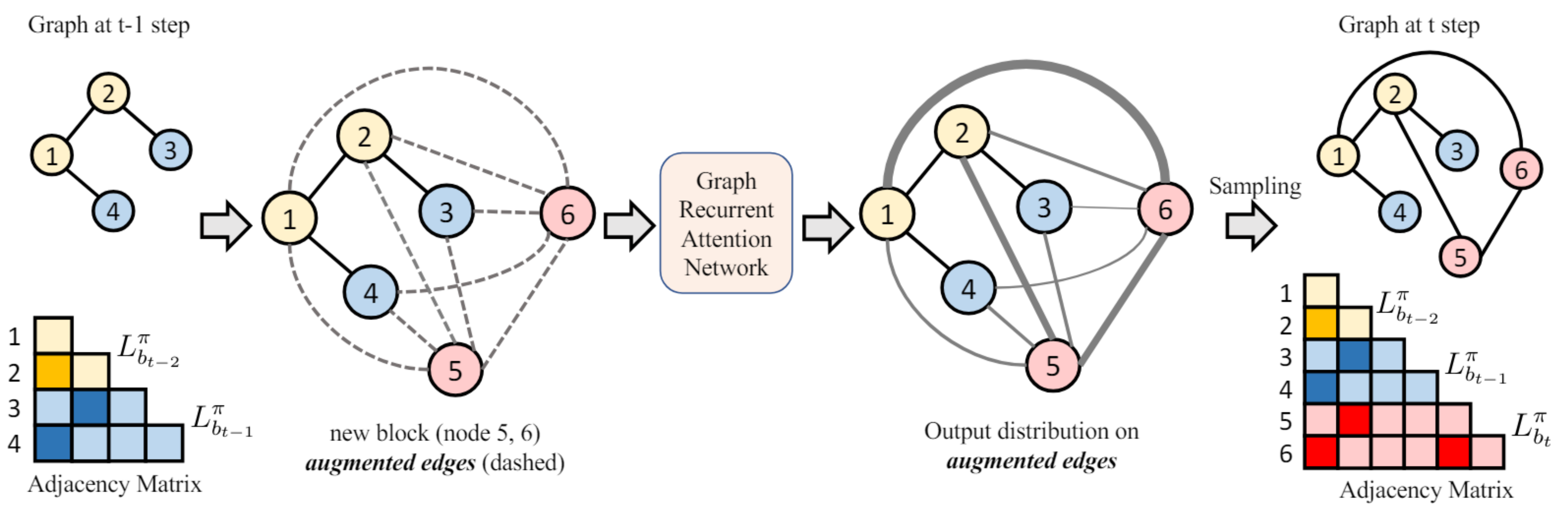
1. Each edge receives a direction from parent to child
2. Iterate the leaf nodes found in the DFS order
    1. If the same leaf node already exists, delete it and create an edge
    2. Otherwise, keep the node

Graph Neural Network

- Input: Graph with node features and edges
- Message Passing: Nodes aggregate information from neighbors
- Update: Nodes update their representations based on aggregated information
- Output: Task-specific outputs



Graph Recurrent Neural Network



Liao, Renjie, et al. "Efficient graph generation with graph recurrent attention networks." *Advances in neural information processing systems* 32 (2019).

Results

	Abstract Syntax Tree	Abstract Syntax Graph
Node type accuracy	89.62%	27.56%
Node type BLEU-score	79.76%	0.17%
Edge accuracy	-	99.48%
Edge precision	-	0

Discussion

- ASG – considers the order of the children
- GNN – does not consider the order of the children
- ASG – sparse edges
- Tends to generate isolated nodes

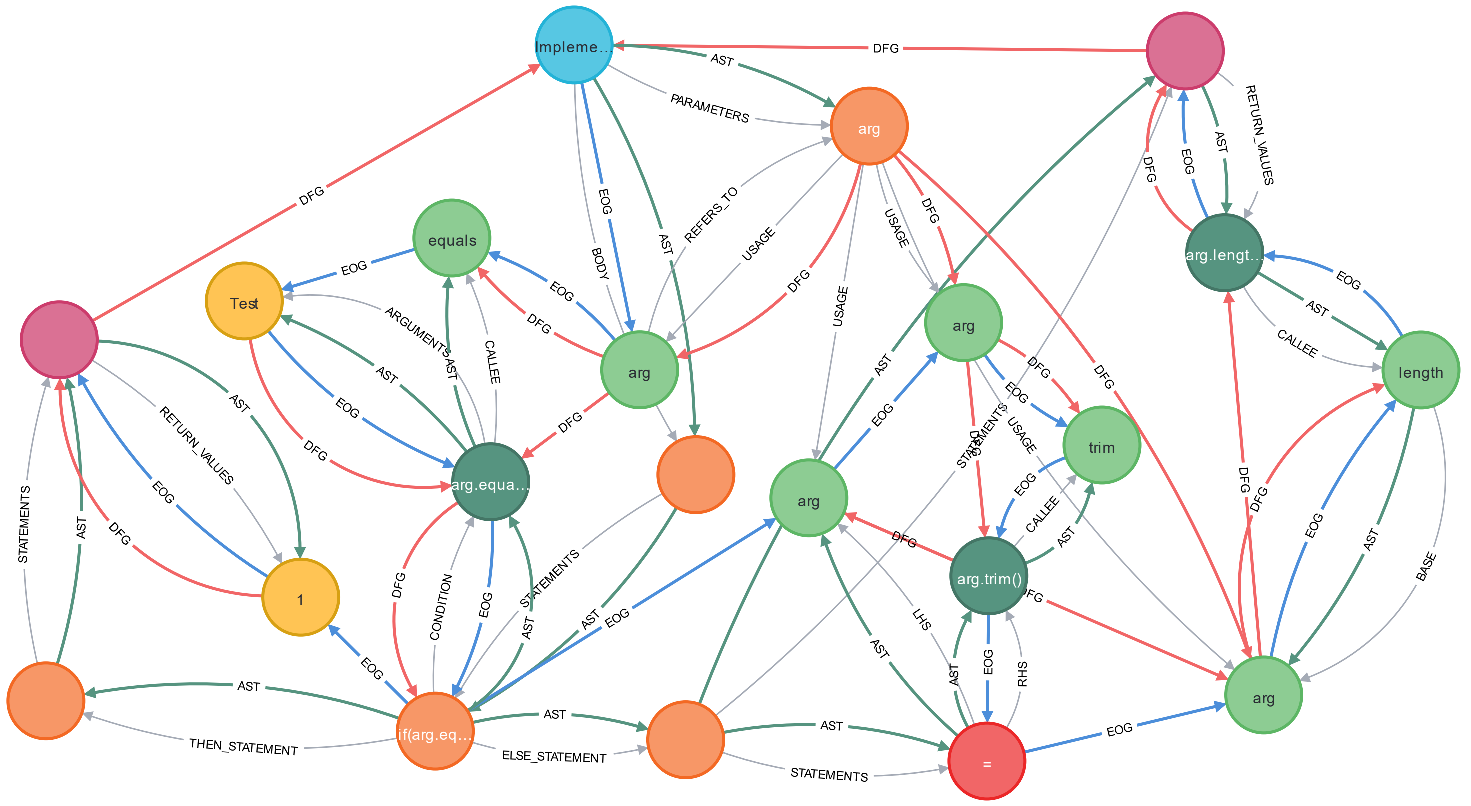
Conclusion

- Abstract Syntax Graph – not a graph (requires order)
- GNNs – not appropriate for code-related tasks
- Current approaches still have weaknesses

Future work

- Code Property Graph = AST + CFG + many others
- Generate a Code Property Graph
- Research ways to deserialize code from Code Property Graph
  - Possible issue: CPG is not injective

Future work



https://fraunhofer-aisec.github.io/cpg/

Thank you!

Q&A