CodeTrek: Flexible Modeling of Code using an Extensible Relational Representation

Pardis Pashakhanloo[1], Aaditya Naik[1], Yuepeng Wang[2], Hanjun Dai[3], Petros Maniatis[3], Mayur Naik[1]

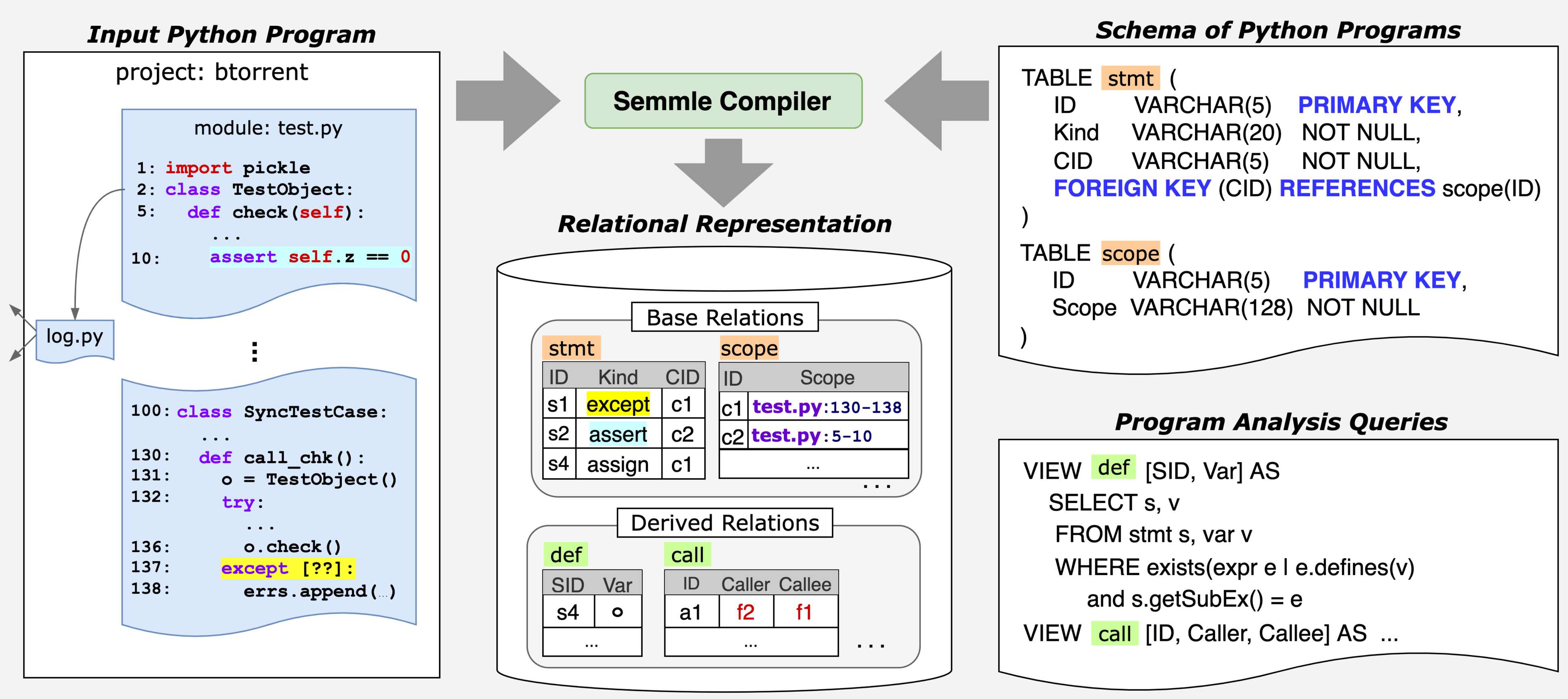
[1]University of Pennsylvania, [2]Simon Fraser University, [3]Google Brain

Summary

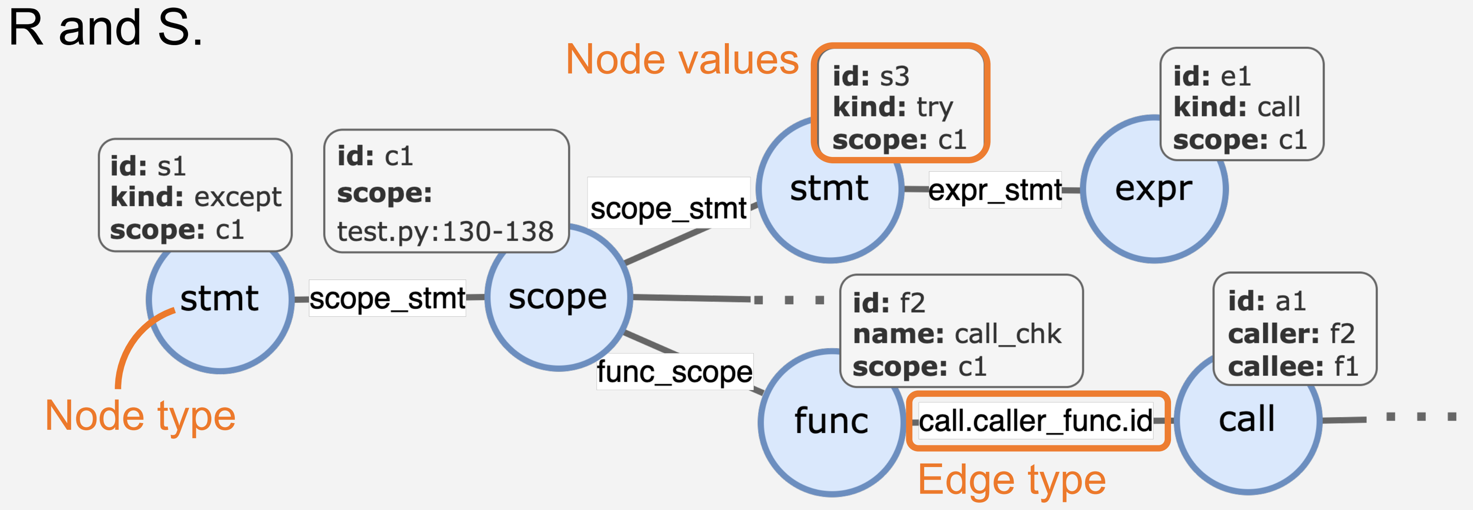
- **CodeTrek** is a deep learning approach that represents **codebases as databases** with rich relational schemas. CodeTrek embeds code using a set of walks that can traverse different relations. The relational representation allows CodeTrek to uniformly represent diverse kinds of program information and leverage program-analysis queries to derive new semantic relations.
- CodeTrek outperforms state-of-the-art neural models by **2-19%** points.

Codebase → Relational Database → Relational Graph

- CodeTrek uses a declarative program analysis framework (**Semmle**) to produce a rich, easily extensible representation of context as a relational database.
- Semmle converts codebases in C, Java, Python, etc., into relational databases that capture the underlying structure and semantics of code, as well as a query language, **CodeQL**, for specifying program analyses to compute new semantic information.
- CodeQL enables CodeTrek to query code as if it were data.

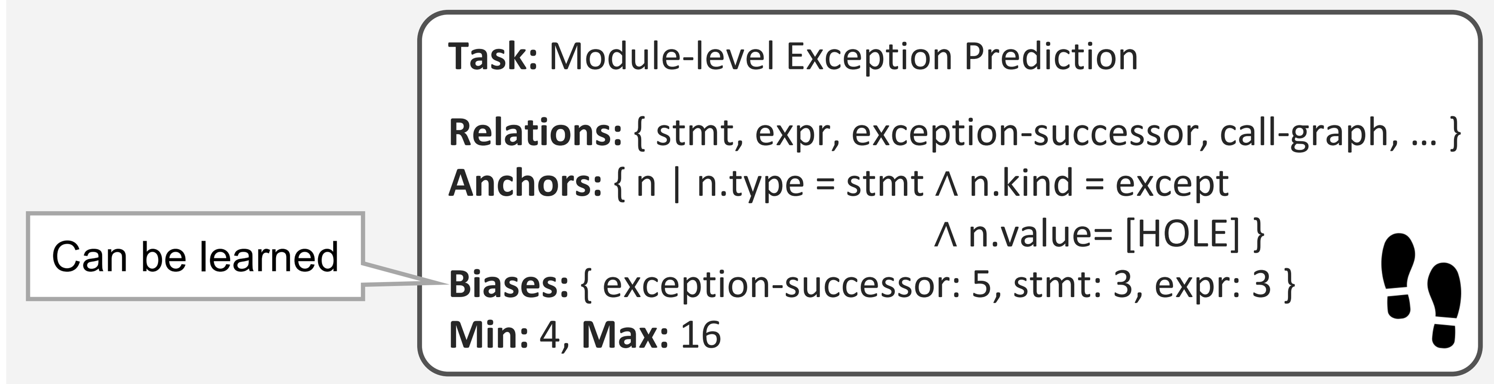


- CodeTrek interprets the produced relations as a graph.
- Each named tuple is represented by a **node** with the values of the tuple as its features.
- **Edges** are added between these nodes such that the edge type R.A_S.B is defined for each referential integrity constraint R.A → S.B between nodes representing tuples of relations R and S.



Biased Random Walks over Relational Graphs

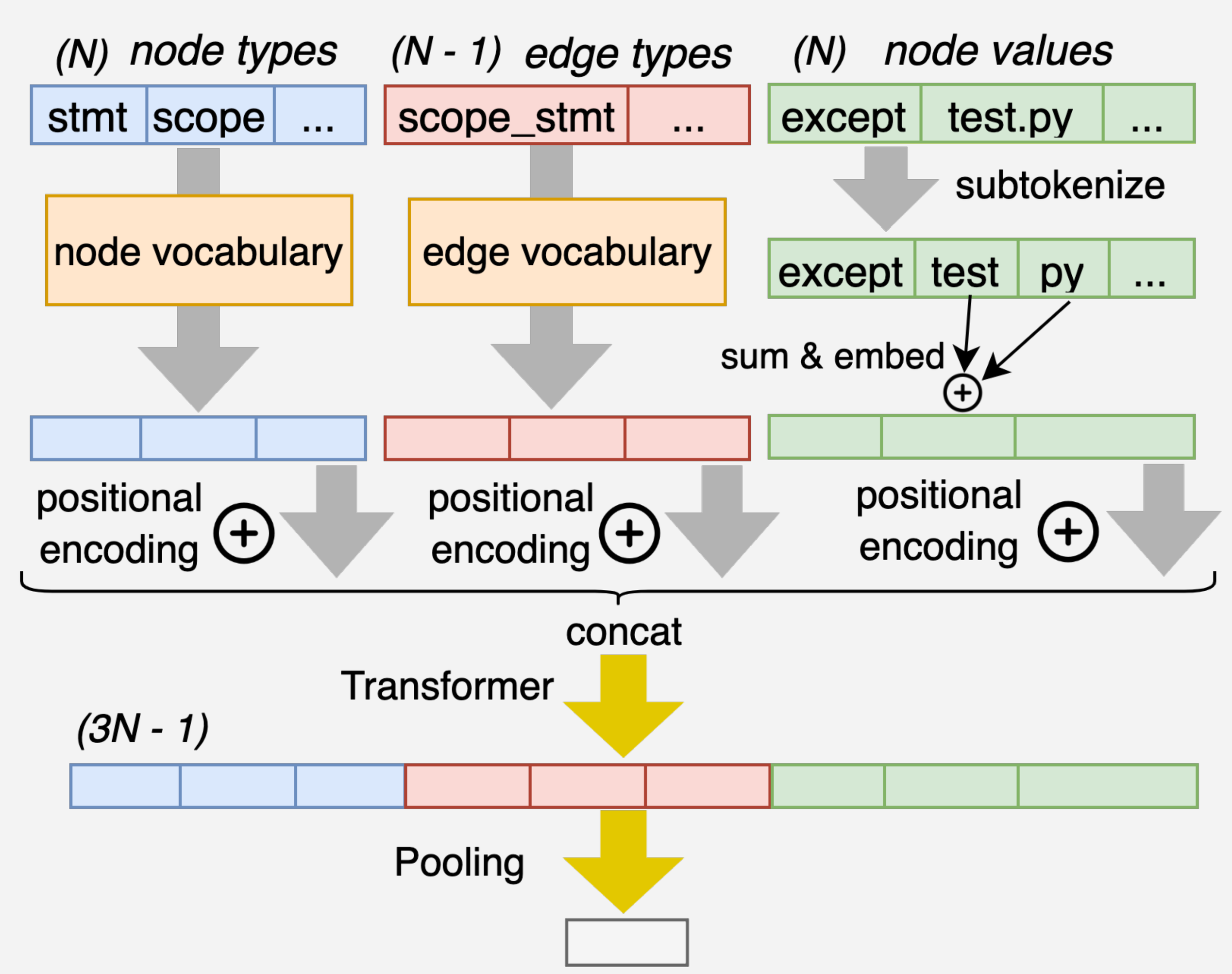
- Context extraction from the resulting graph is done via **biased random walks** over the graph, in a fashion defined by a **walk specification**.



- The walk generator traverses the graph by biasing traversal of edges according to neighbor's node type.
- If no bias is specified, walks are simply fair random walks.
- Different probability mixes for different node types encourage the model to sample walks that are more relevant to a task.

Embedding Random Walks

- To convert random walks to a distributed representation, CodeTrek embeds each walk with *N* nodes using a **Transformer** encoder, and then produces an order-invariant representation of the set of walks using the **Deep Set** architecture. The resulting hidden representation can then be used to make predictions for the code-reasoning task.

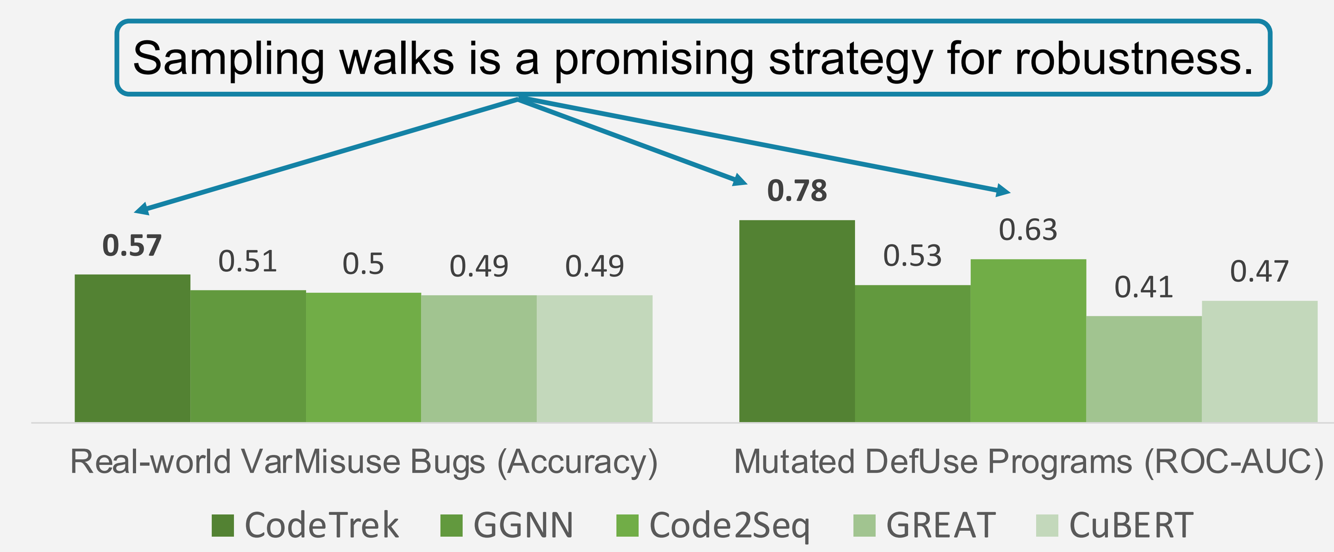


Results

1. *CodeTrek models **perform** better; especially on long-range ( ) and complex logic tasks ( ).*

Task	CodeTrek	GGNN	Code2Seq	GREAT	CuBERT
VarMisuse	<u>91%</u>	69%	–	82%	89%
VarMisuseFun	70%	54%	52%	<u>89%</u>	84%
Exception	<u>63%</u>	28%	30%	44%	42%
ExceptionFun	65%	51%	51%	68%	<u>69%</u>
DefUse	<u>98%</u>	76%	–	84%	76%
DefUseFun	<u>91%</u>	77%	66%	82%	71%
VarShadow	<u>94%</u>	71%	70%	93%	91%

2. *CodeTrek produces **robust** models.*

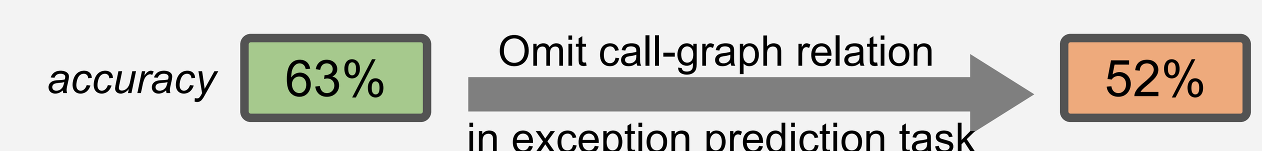


3. *CodeTrek is effective on **long-range** tasks.*

Long-range code-understanding tasks are tasks that require reasoning beyond a single function, e.g.,

- Inter-procedural tasks (which exception to catch?)
- Global tasks (is any global variable shadowed?)

"call-graph" relations enable CodeTrek to make informed predictions in long-range tasks.



4. ***Richness** of semantic information available to the model has a significant accuracy impact.*