

Exploiting Code Symmetries for Learning Program Semantics

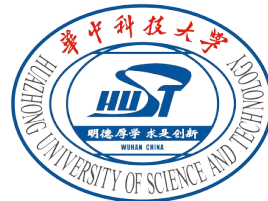
Kexin Pei, Weichen Li*, Qirui Jin*, Shuyang Liu, Scott Geng
Lorenzo Cavallaro, Junfeng Yang, Suman Jana



COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK



W UNIVERSITY of WASHINGTON



Machine Learning Shows Promise for Analyzing Programs



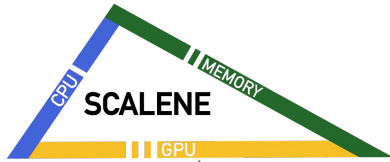
Detecting and Exploiting Vulnerabilities



Brandon Rossi
@OxConda

I told ChatGPT to pentest a scope of IPs and it found me \$69,420 worth of bugs.

Python Profiler



GitHub Copilot



Program Optimization

```
⚡ z1 = [i for i in range(0, 300000)][299999]
```

15

```
# Proposed optimization:  
# This code can be optimized by using the built-in function max()  
z1 = max(range(0, 300000)) # ~10x faster
```

Explain Code

The screenshot shows the GitHub Copilot interface with the 'EXPLAIN' tab selected. It displays a code snippet for a FizzBuzz program. Below the code, there is a dropdown menu set to 'Explain code' and a button labeled 'Ask Copilot'. The 'RESULT' section shows a detailed explanation of the code's logic.

```
const donut = Number(process.argv[2])  
  
if ((donut % 3 === 0) && (donut % 5 === 0))  
{  
  console.log('FizzBuzz')  
}
```

Explain code

Advanced

Ask Copilot

RESULT

/* Here is the explanation for the code above:
1. We are taking the argument from the command line and converting it to a number.
2. We then use an if statement to check if the number is divisible by 3 and 5.
3. If the number is divisible by 3 and 5, we output FizzBuzz.
4. If the number is not divisible by 3 and 5, we output the number.*/

Translate Code

The screenshot shows the GitHub Copilot interface with the 'LANGUAGE TRANSLATION' tab selected. It displays the same code snippet as the 'Explain Code' section. Below the code, there is a dropdown menu set to 'python' and a button labeled 'Ask Copilot'. The 'Translate code into:' section shows the target language.

```
const donut = Number(process.argv[2])  
  
if ((donut % 3 === 0) && (donut % 5 === 0))  
{  
  console.log('FizzBuzz')  
}
```

Translate code into:

python

Ask Copilot

Limitations: Lack Understanding of Program Semantics

A code summarization based on GGNN (Li et al. 2017, Rabin et al. 2020, Bui et al. 2021)

Permute lines 7,8,9 (move line 9 before line 8).

```
1 private void FuncName(String modeStr, Matcher matcher) {
2     boolean commaSeperated = false;
3     for (int i = 0; i < 1 || matcher.end() < modeStr.length(); i++) {
4         if (i > 0 && (!commaSeperated || !matcher.find())) {
5             throw new IllegalArgumentException(modeStr);
6         }
7         String str = matcher.group(2);
8         char type = str.charAt(str.length() - 1);
9         boolean user, group, others, stickyBit;
10        user = group = others = stickyBit = false;
11 }
```

“Run”

```
1 private void FuncName(String modeStr, Matcher matcher) {
2     boolean commaSeperated = false;
3     for (int i = 0; i < 1 || matcher.end() < modeStr.length(); i++) {
4         if (i > 0 && (!commaSeperated || !matcher.find())) {
5             throw new IllegalArgumentException(modeStr);
6         }
7         boolean user, group, others, stickyBit;
8         String str = matcher.group(2);
9         char type = str.charAt(str.length() - 1);
10        user = group = others = stickyBit = false;
11 }
```

“Update”

Consequences: Lacking Robustness and Generalization

- I. **Overfit** to spurious textual and task-specific patterns
- II. **Distribution shift**: program syntax and task requirement changes

Existing Practice of Teaching ML Code Semantics

Common practice 1

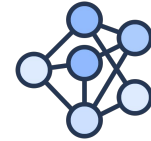


Program

Data Augmentation



Semantics-preserving code transformations



Code Models

Common practice 2



Program

Grounding

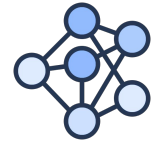


Structures

Execution Traces

Natural Language

Other code modalities



Code Models

Limitation: Expensive and No Guarantee

- I. **Expensive:** Enumerate samples with different transformations and modalities
- II. **No guarantee:** the make can still make mistakes on seen transformed samples

How to Specify Code Semantics and Build ML Models Respecting it?

Operational Semantics

$$\text{PRE} \frac{}{\alpha.t \xrightarrow{\text{true}.\alpha} t}$$

$$\text{COND} \frac{t \xrightarrow{b'.\alpha} t'}{\text{if } b \text{ then } t \xrightarrow{b' \wedge b.\alpha} t'} \quad \text{fv}(b) \cap \text{bv}(\alpha) = \emptyset$$

$$\text{SUM1} \frac{t \xrightarrow{b.\alpha} t'}{t + u \xrightarrow{b.\alpha} t'}$$

$$\text{SUM2} \frac{u \xrightarrow{b.\alpha} u'}{t + u \xrightarrow{b.\alpha} u'}$$

$$\text{PAR1} \frac{t \xrightarrow{b.\alpha} t'}{t \mid u \xrightarrow{b.\alpha} t' \mid u} \quad \text{bv}(\alpha) \cap \text{fv}(u) = \emptyset$$

$$\text{PAR2} \frac{u \xrightarrow{b.\alpha} u'}{t \mid u \xrightarrow{b.\alpha} t \mid u'} \quad \text{bv}(\alpha) \cap \text{fv}(t) = \emptyset$$

$$\text{COM} \frac{t \xrightarrow{b.c?x} t' \quad u \xrightarrow{b'.c!e} u'}{t \mid u \xrightarrow{b \wedge b'.c} t'[e/x] \mid u'}$$

$$\text{RES} \frac{t \xrightarrow{b.\alpha} t'}{t \setminus L \xrightarrow{b.\alpha} t' \setminus L} \quad \text{Chan}(\alpha) \notin L$$

$$\text{REN} \frac{t \xrightarrow{b.\alpha} t'}{t[f] \xrightarrow{b.f(\alpha)} t'[f]}$$

$$\text{REC} \frac{t[\bar{e}/\bar{x}] \xrightarrow{b.\alpha} t'}{P(\bar{e}) \xrightarrow{b.\alpha} t'} \quad P(\bar{x}) \Leftarrow t$$

- **Interpreter**
- **Static Analysis**
- ...

Bound by

- execution
- interpretation
- fix-point
- syntax

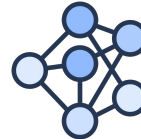
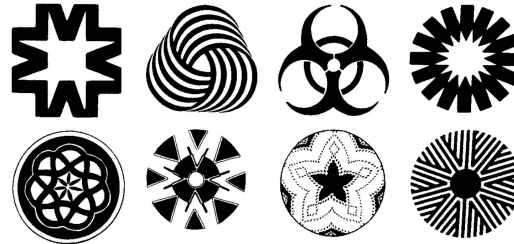


ML Models
e.g., LLMs

A New Group-Theoretic Framework to Formalize Learning Code Semantics

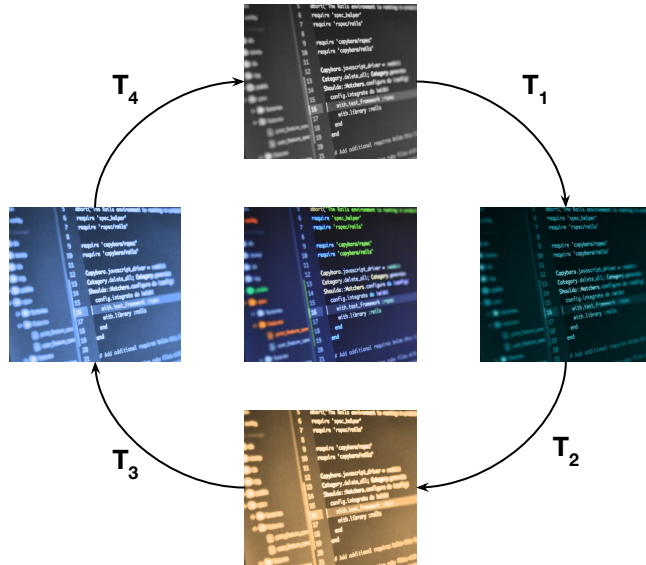


Symmetry Groups



Code Models

Code Symmetry Group



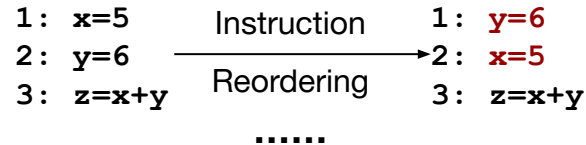
$$T_i \in \mathbf{T}$$

Semantics-preserving code transformations

\mathbf{T} is a **Group** iff

- **Associative:** $T_i \cdot (T_j \cdot T_k) = (T_i \cdot T_j) \cdot T_k$
- **Inverse:** $\forall T_i \in \mathbf{T}, \exists T_j \in \mathbf{T}, T_i \cdot T_j = \mathbf{1}$
- **Identity:** $\forall \mathbf{1} \in \mathbf{T}, \mathbf{1} \cdot T_i = T_i \cdot \mathbf{1}$
- **Closure:** $\forall T_i, T_j \in \mathbf{T}, (T_i \cdot T_j) \in \mathbf{T}$

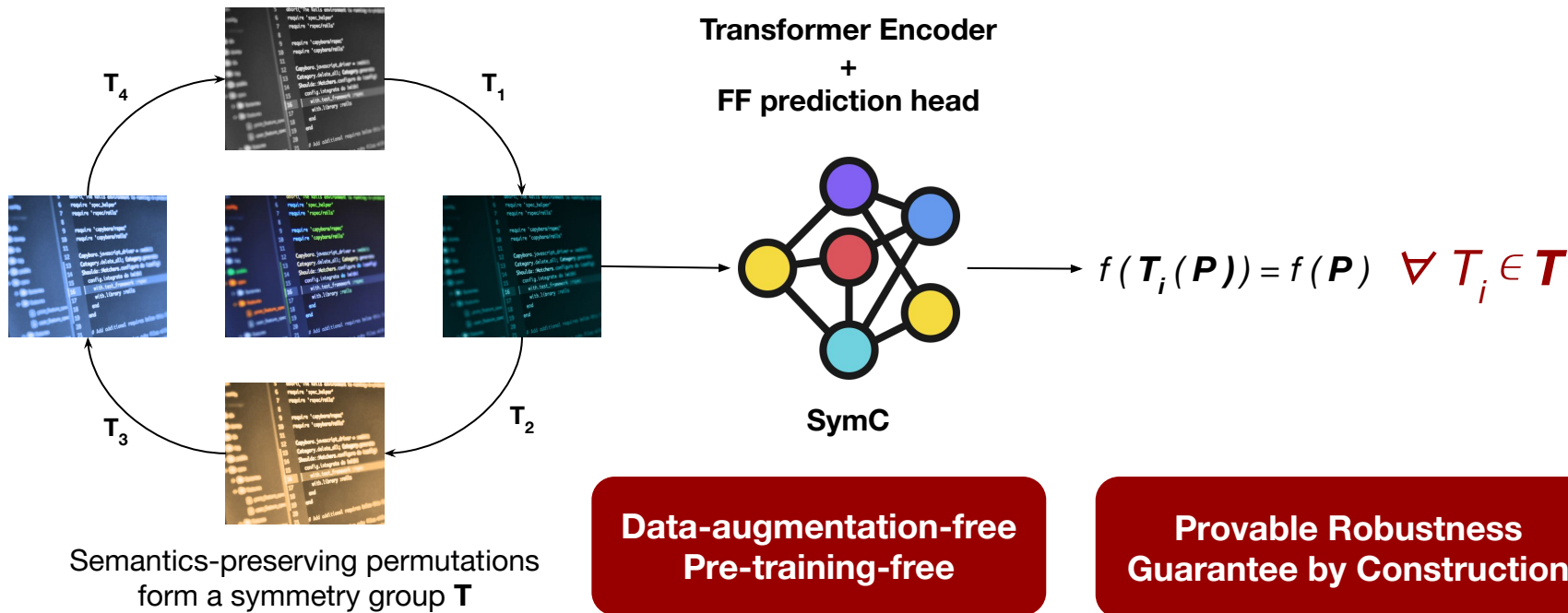
E.g., Permutation Group \mathbf{T}



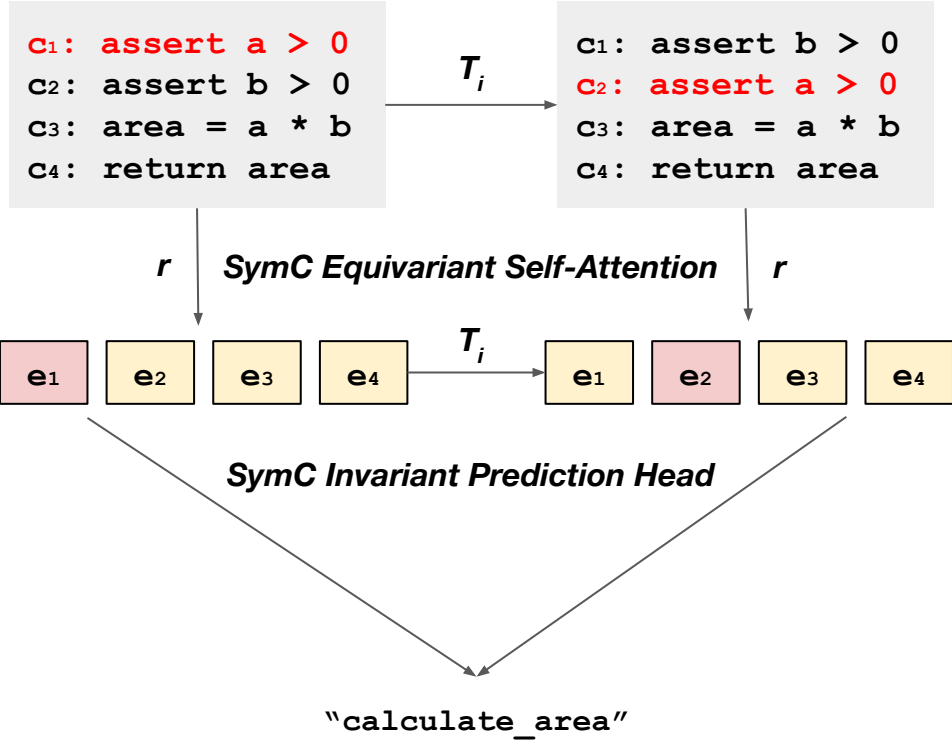
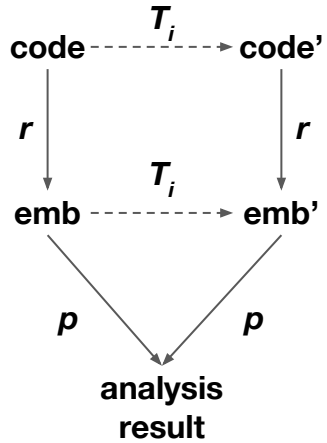
Key Benefit

- Compositionality
 - Prove by Induction
- Amenable to Neural Architecture Design
 - AI4Science
 - Geometric DL

SymC: Permutation-Group-Invariant Transformer



Group Invariance and Equivariance in SymC



r: representation learning module, e.g., self-attention layers

p: predictive learning module, e.g., fully-connected prediction head

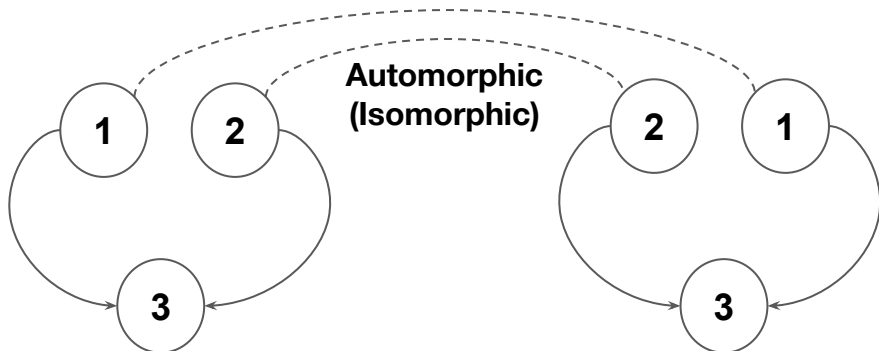
Aut(PDG): The Automorphism Group of Program Dependence Graph

```
1 int a = 1;  
2 int b = 2;  
3 int c = a + b;
```

```
1 int b = 2;  
2 int a = 1;  
3 int c = a + b;
```

Program Dependence Graph

Program Dependence Graph



- **Data Dependency**

- **RAW:** Read after Write
- **WAR:** Write after Read
- **WAW:** Write after Write

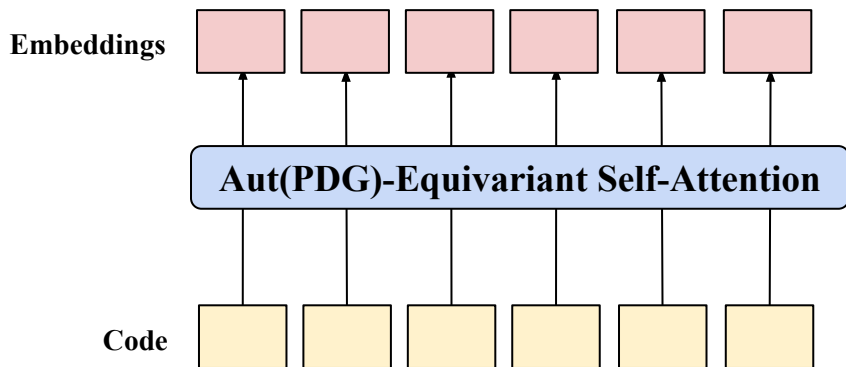
- **Control Dependency**

Distance Matrix d :

Relative distance to the lowest common ancestor (LCA)

0	inf	0
inf	0	0
1	1	0

Aut(PDG)-Equivariant Self-Attention



$$\text{Attn}(x) = V \cdot (K^T \cdot Q)$$

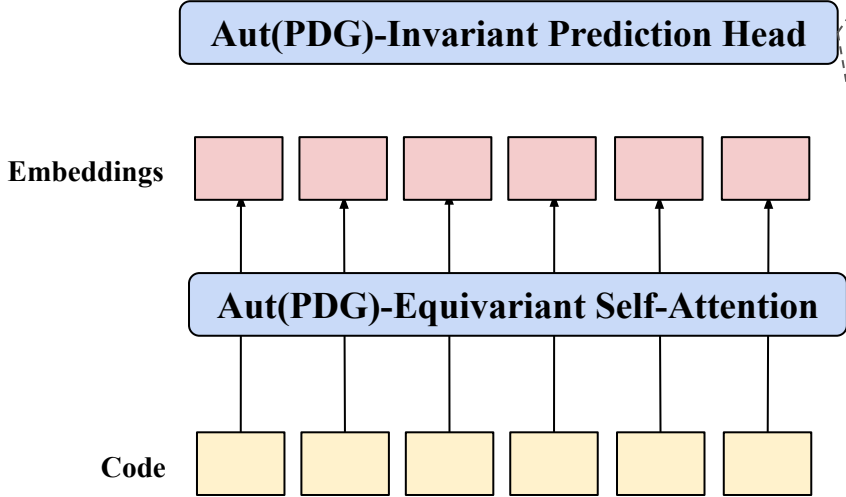
$$\begin{aligned} \text{Attn}(xp) &= Vp \cdot ((Kp)^T \cdot Qp) \\ &= Vpp^T(K^T \cdot Q)p = V \cdot (K^T \cdot Q)p = \text{Attn}(x)p \end{aligned}$$

How to make it equivariant **only to** Aut(PDG), the semantics-preserving permutations?

$$G\text{-Attn}(x) = V \cdot (K^T \cdot Q + d_{PDG})$$

$$\begin{aligned} G\text{-Attn}(xp) &= Vp \cdot ((Kp)^T \cdot Qp + d_{PDG}) \\ &= Vpp^T(K^T \cdot Q)p + Vpd_{PDG} \\ &= V \cdot (K^T \cdot Q + d_{PDG})p = G\text{-Attn}(x)p \end{aligned}$$

Aut(PDG)-Invariant Prediction Head



Pooling Based, e.g., Mean, Sum, etc.

Sum([] [] [] [] [] [])

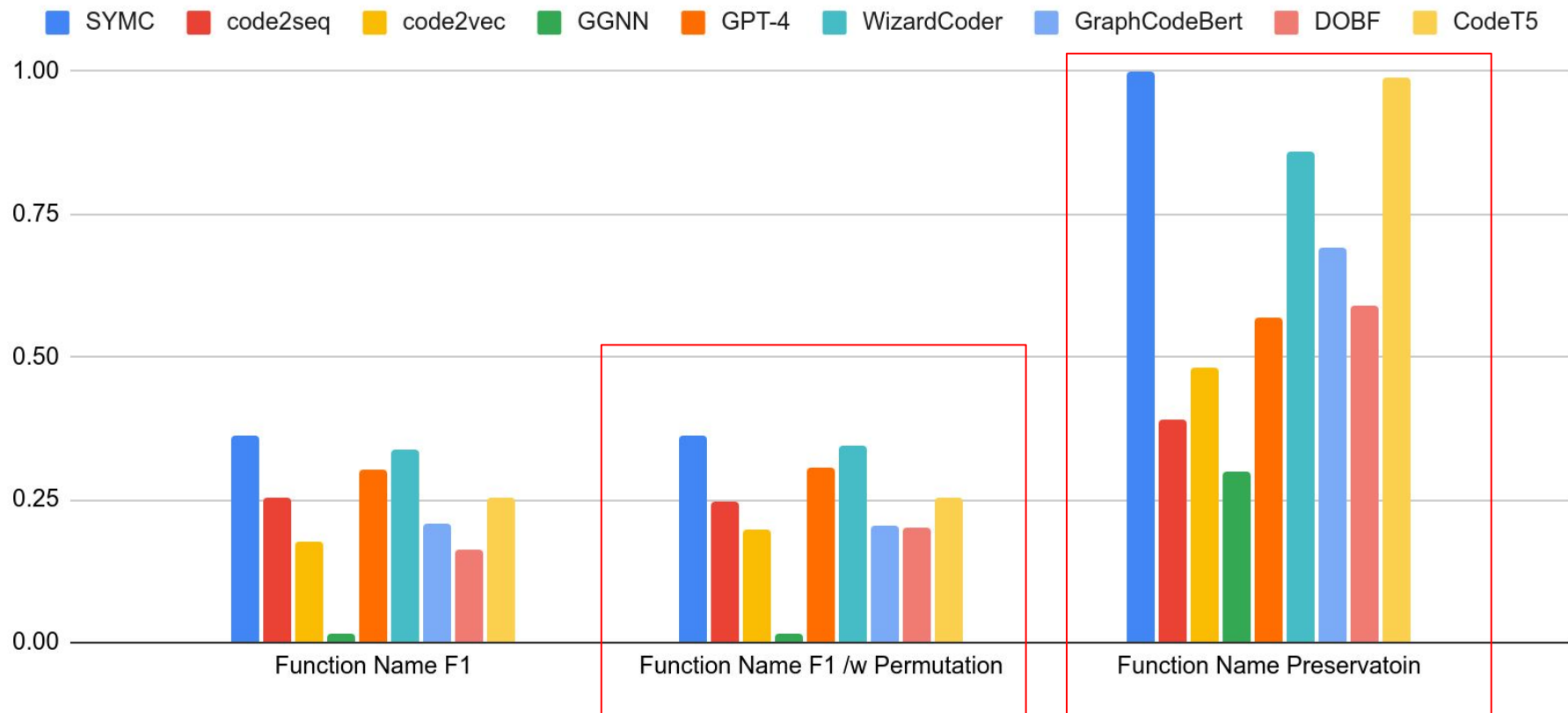
Avg([] [] [] [] [] [])

Token-level Based, e.g., [CLS]

$$GA = V \cdot (K^T \cdot q + d_{[:,q]})$$

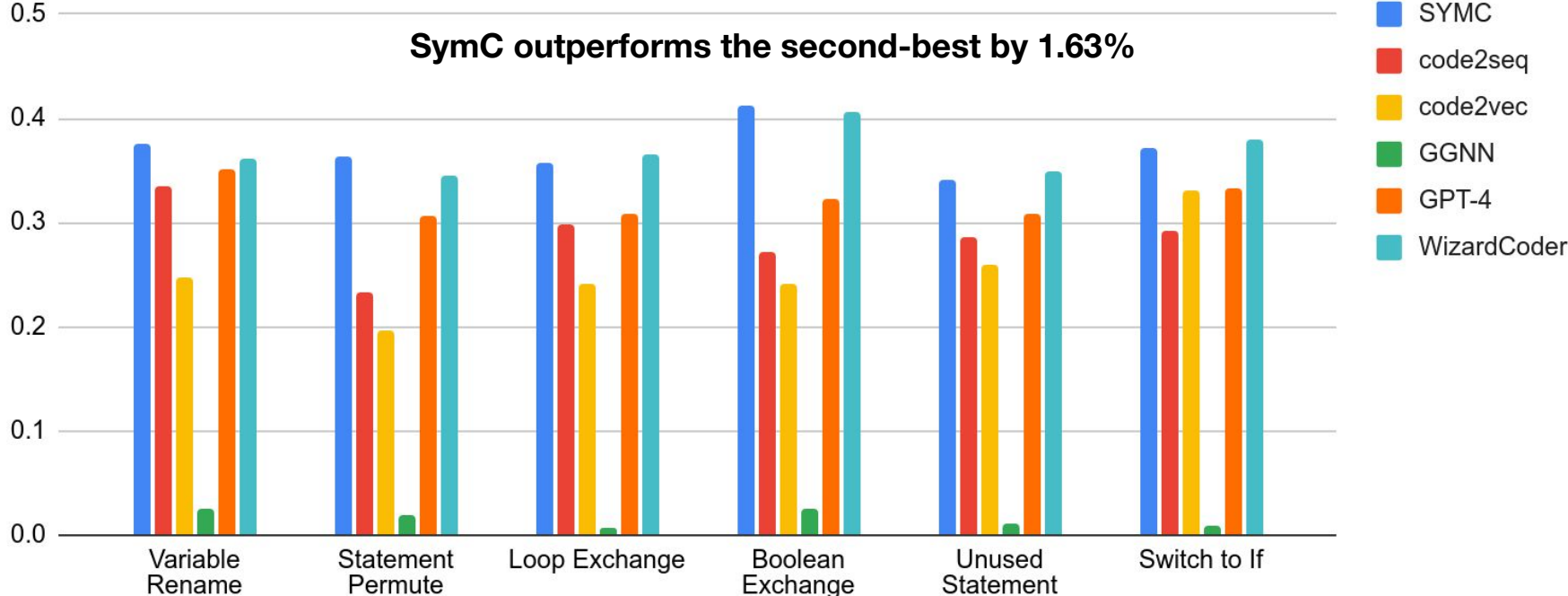
a single query token column of query token to all others

Results: Robustness Against Permutation



Results: Generalization against Unseen Transformations

The performance (F1) of SYMC and baselines against different unseen permuted code transformations.



Results: Invariance and Generalization

■ SymC ■ Palmtree ■ PalmTree-O ■ PalmTree-N

