

# Learning To Edit Code

Ph.D. Defense Presentation

Saikat Chakraborty

Department of Computer Science  
Columbia University.

July 26, 2022

Committee

Dr. Gail Kaiser (Chair), Columbia University

Dr. Baishakhi Ray (Adviser), Columbia University

Dr. Kathleen McKeown, Columbia University

Dr. Junfeng Yang, Columbia University

Dr. Kai-Wei Chang, University of California at Los Angeles

# Automatic Code Editing

## Adding Feature

```
public String removeComment(String leftOver) {  
-    if (hasBlockComment(leftOver)){  
+    while (hasBlockComment(leftOver)){  
        leftOver = removeBlockComment(leftOver);  
    }  
-    if (hasLineComment(leftOver)){  
+    while (hasLineComment(leftOver)){  
        leftOver = removeLineComment(leftOver);  
    }  
    return leftOver;  
}
```

## Adding Feature

```
public String removeComment(String leftOver) {  
-    if (hasBlockComment(leftOver)){  
+    while (hasBlockComment(leftOver)){  
        leftOver = removeBlockComment(leftOver);  
    }  
-    if (hasLineComment(leftOver)){  
+    while (hasLineComment(leftOver)){  
        leftOver = removeLineComment(leftOver);  
    }  
    return leftOver;  
}
```

## Bug-fixing

```
public abstract void removeSessionCookies()  
                      throws Exception + ;  
-    throw new MustOverrideException();  
- }
```

## Adding Feature

```
public String removeComment(String leftOver) {  
-    if (hasBlockComment(leftOver)){  
+    while (hasBlockComment(leftOver)){  
        leftOver = removeBlockComment(leftOver);  
    }  
-    if (hasLineComment(leftOver)) {  
+    while (hasLineComment(leftOver)) {  
        leftOver = removeLineComment(leftOver);  
    }  
    return leftOver;  
}
```

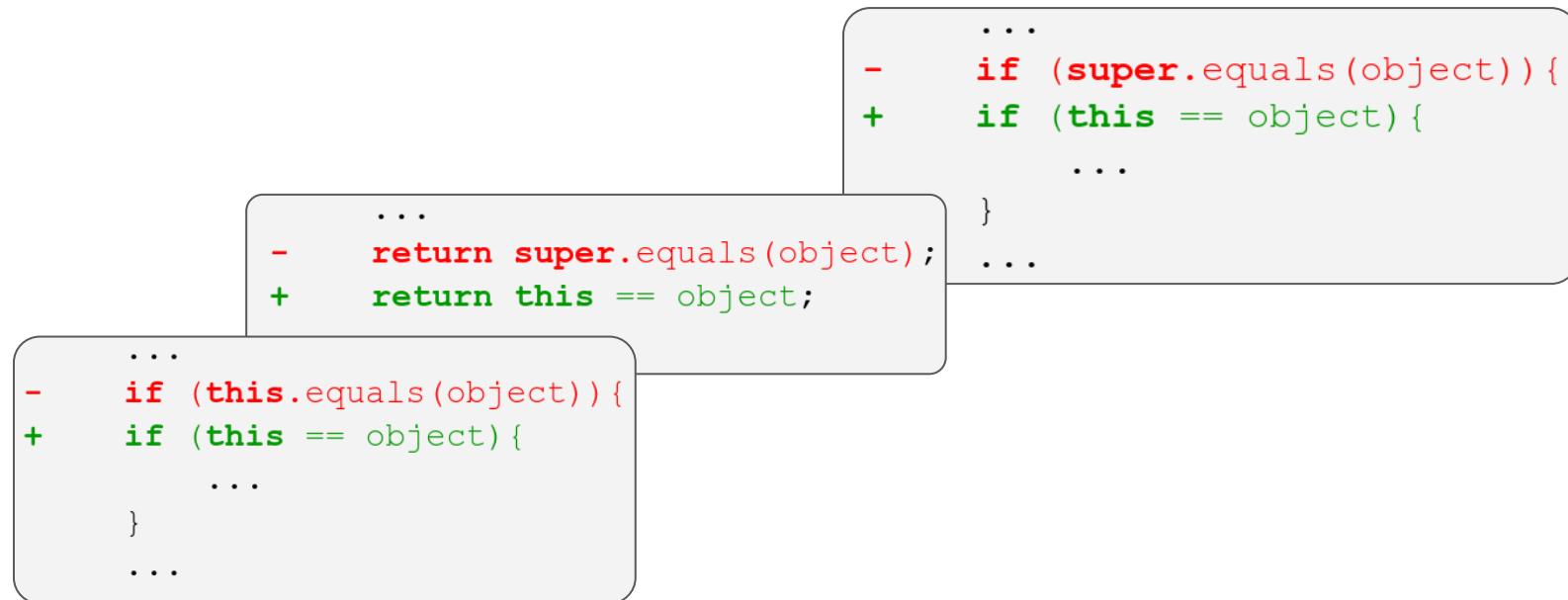
## Bug-fixing

```
public abstract void removeSessionCookies()  
    throws Exception + ;  
-    throw new MustOverrideException();  
- }
```

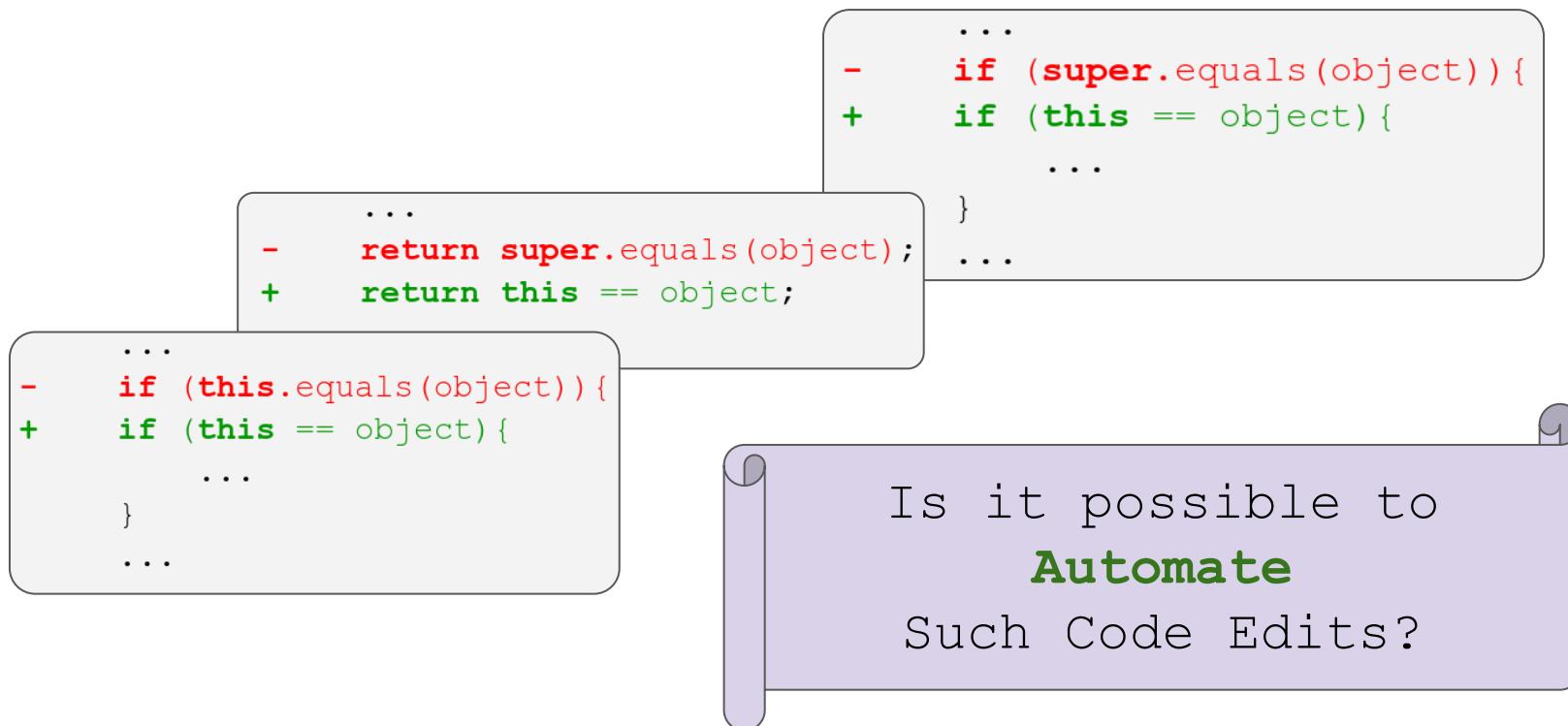
## Refactoring

```
void visit(JSession xsession, Timer t) throws Exception {  
    if (xsession != null && t.getTime() > xsession.getStartTime()) {  
        visit((JNode) xsession, t);  
    }  
    else {  
        visit(new JNode(), new Timer());  
    }  
}
```

# Code Edits Are Repetitive (Meng *et. al.* 2011[1], 2013[2], Ray *et. al.* 2015[3])



# Code Edits Are Repetitive (Meng *et. al.* 2011[1], 2013[2], Ray *et. al.* 2015[3])



# Automating Code Edits - Template/Search based

## Edit Template

```
...
- return super.equals(object);
+ return this == object;
...
```

# Automating Code Edits - Template/Search based

## Edit Template

```
...
- return super.equals(object);
+ return this == object;
...
```

```
public Model copy(Model instance) {
    ...
    instance.notify();
    if (super.equals(instance) && !instance.isEmpty()) {
        return instance.clone();
    }
    ...
}
```

# Automating Code Edits - Template/Search based

## Edit Template

```
...
- return super.equals(object);
+ return this == object;
...
```

## Match Found

```
public Model copy(Model instance) {
    ...
    instance.notify();
    if (super.equals(instance) && !instance.isEmpty()) {
        return instance.clone();
    }
    ...
}
```

# Automating Code Edits - Template/Search based

## Edit Template

```
...
- return super.equals(object);
+ return this == object;
...
```

## Patch Applied

```
public Model copy(Model instance) {
    ...
    instance.notify();
    if (this == instance && !instance.isEmpty()) {
        return instance.clone();
    }
    ...
}
```

# Automating Code Edits - Template/Search based

## Edit Template

```
...
- return super.equals(object);
+ return this == object;
...
```

## Patch Applied

```
public Model copy(Model instance) {
    ...
    instance.notify();
    if (this == instance && !instance.isEmpty()) {
        return instance.clone();
    }
    ...
}
```

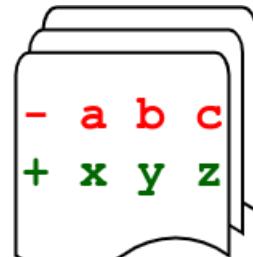
## Problems

- Too many templates to write.
- Bottleneck : Template Matching / Code Search

# Code Editing Task - Learning Based Solution

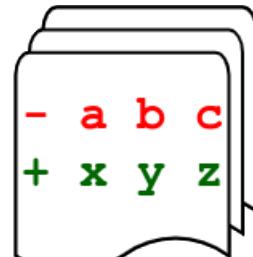
# Code Editing Task - Learning Based Solution

Example Code Edits



# Code Editing Task - Learning Based Solution

Example Code Edits



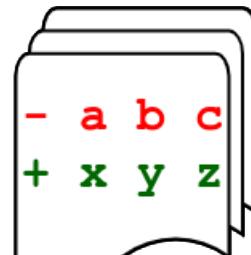
`return super.equals(object)` →

Code Before Edit



# Code Editing Task - Learning Based Solution

Example Code Edits



`return super.equals(object)` →

Code Before Edit

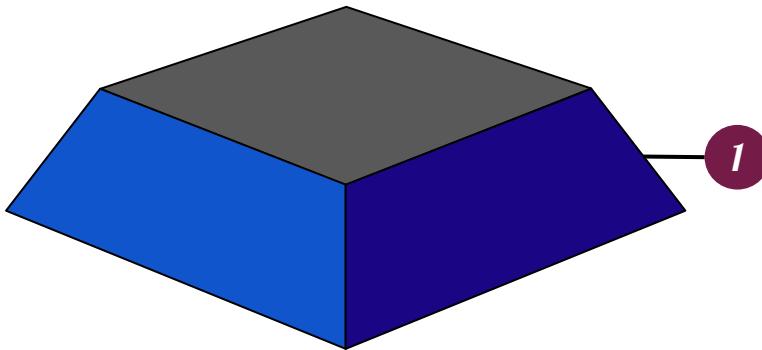


→ `return this == object`

Code After Edit

# What are my Contributions?

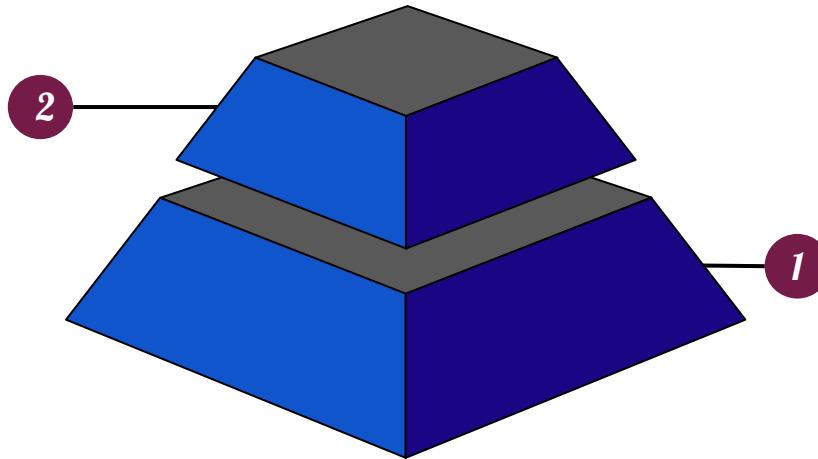
# What are my Contributions?



**Identification of  
Technical  
Challenges in  
Learning based Code  
Editing.**

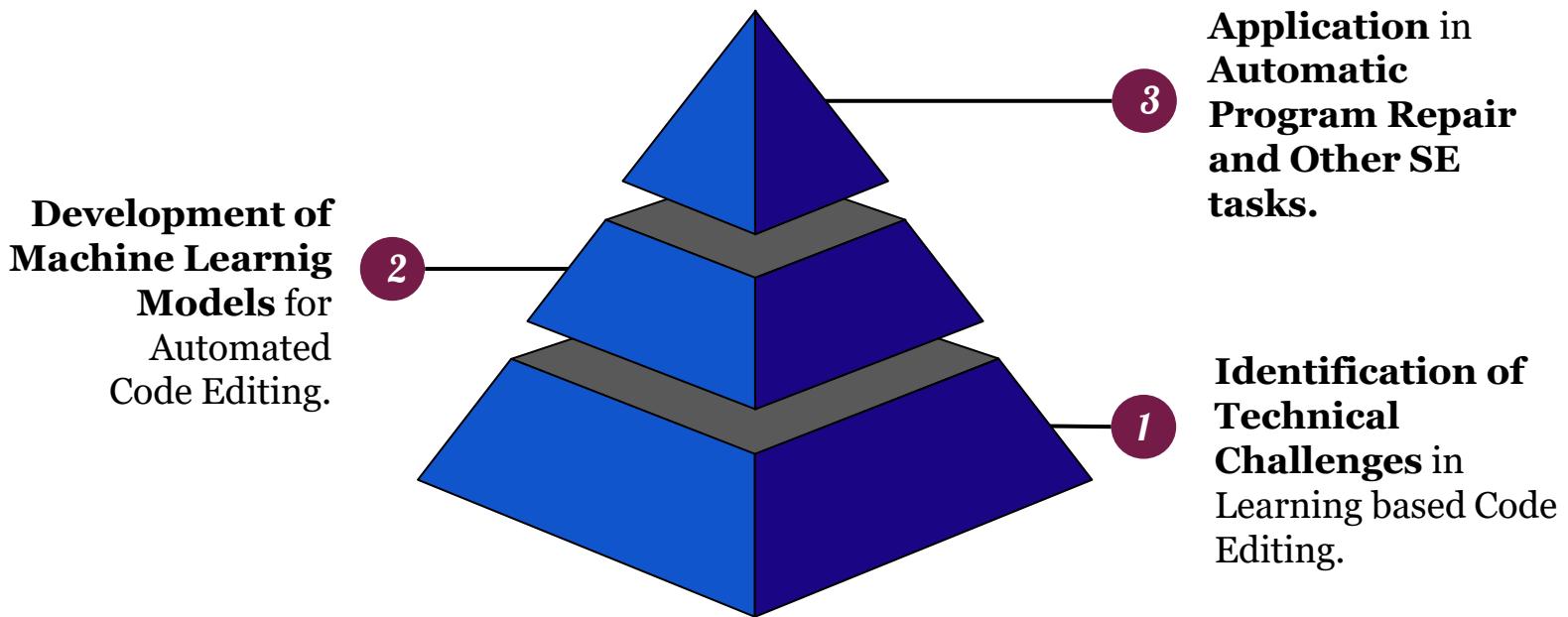
# What are my Contributions?

**Development of  
Machine Learning  
Models for  
Automated  
Code Editing.**



**Identification of  
Technical  
Challenges in  
Learning based Code  
Editing.**

# What are my Contributions?



# Automated Code Editing - Existing works.

# Automated Code Editing - Existing works.

## Template-based

1. Modern IDE (Eclipse, IntelliJ IDEA) - Refactoring, Boilerplate Code.
2. Meng et.al. - PLDI'11 - Infer edit template with graph matching [1].

# Automated Code Editing - Existing works.

## Template-based

1. Modern IDE (Eclipse, IntelliJ IDEA) - Refactoring, Boilerplate Code.
2. Meng et.al. - PLDI'11 - Infer edit template with graph matching [1].

## Mutation Learning

1. Rolim et. al. - ICSE'17 - Designed a DSL for representing Edits [4].
2. Dinella et.al. - ICLR'20 - Neural Turing Machine [5].

# Automated Code Editing - Existing works.

## Template-based

1. Modern IDE (Eclipse, IntelliJ IDEA) - Refactoring, Boilerplate Code.
2. Meng et.al. - PLDI'11 - Infer edit template with graph matching [1].

## Mutation Learning

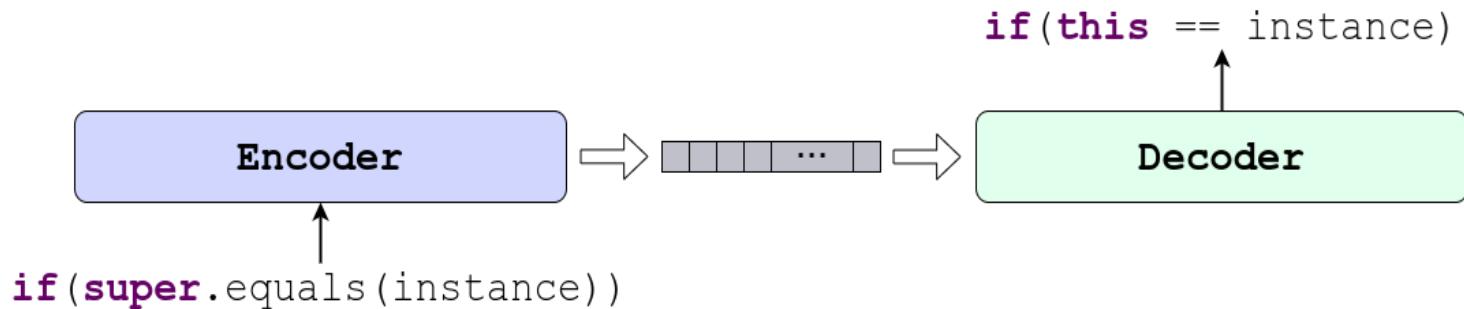
1. Rolim et. al. - ICSE'17 - Designed a DSL for representing Edits [4].
2. Dinella et.al. - ICLR'20 - Neural Turing Machine [5].

## Encoder-Decoder

1. Tufano et. al. - ASE'18 [6], ICSE'19 [7], TOSEM'19 [8].  
- Abstract tokenization.
2. Chen et. al. - TSE'19 - Copy Attention-based models [9].
3. Tufano et. al. - ICSE'21. - Multi-Encoder models[10].

# Encoder-Decoder Based Code Editing

# Code Editing with Encoder-Decoder



**Encoder** and **Decoder** learns **Edit Pattern** and  
To **Apply the Pattern** in **Similar Context**.

# Encoder-Decoder Based Code Editing

Where does my dissertation stand?

| Method   | Concrete Code | Syntactic Correctness | Contextual Correctness | Code Naturalness | Multi Modality |
|--|---------------|-----------------------|------------------------|------------------|----------------|
| M. Tufano <i>et. al.</i> ASE 2018 [6], ICSE 2019 [7] | ✗             | ⚠                     | ⚠                      | ✗                | ✗              |
| SequenceR - Chen <i>et. al.</i> TSE 2019 [9]         | ✓             | ⚠                     | ⚠                      | ⚠                | ✗              |
| <b>CODIT - TSE 2020</b>                              | ✓             | ✓                     | ⚠                      | ⚠                | ✗              |
| CodeBERT* - Feng <i>et. al.</i> EMNLP 2020 [11]      | ✓             | ⚠                     | ⚠                      | ⚠                | ✗              |
| <b>PLBART - NAACL 2021</b>                           | ✓             | ⚠                     | ⚠                      | ⚠                | ✗              |
| CoCoNut - Lutellier <i>et. al.</i> ISSTA 2020 [12]   | ✓             | ⚠                     | ⚠                      | ⚠                | ✓              |
| R. Tufano <i>et. al.</i> - ICSE 2021 [10]            | ✗             | ⚠                     | ⚠                      | ✗                | ✓              |
| <b>MODIT – ASE 2021</b>                              | ✓             | ⚠                     | ⚠                      | ⚠                | ✓              |
| <b>NatGen – FSE 2022</b>                             | ✓             | ⚠                     | ⚠                      | ⚠                | ✓              |

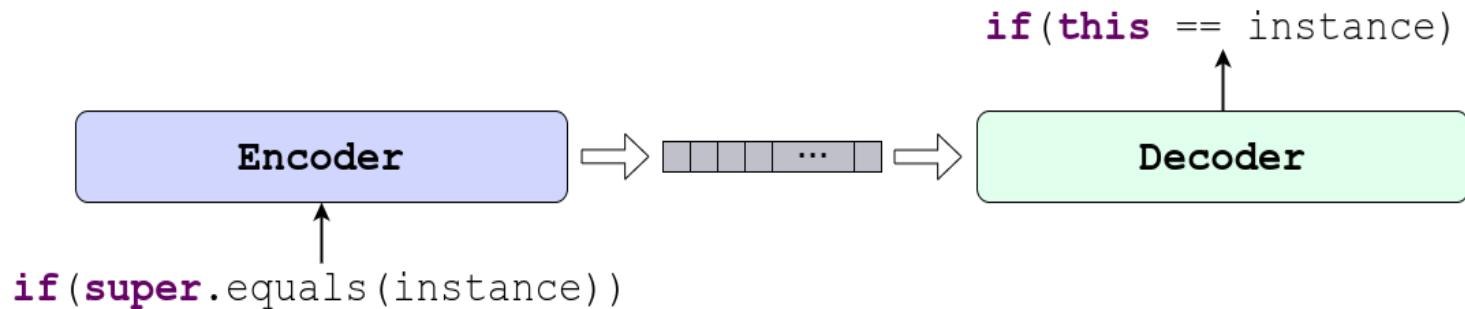
✓ Used/Guaranteed

✗ Not Used

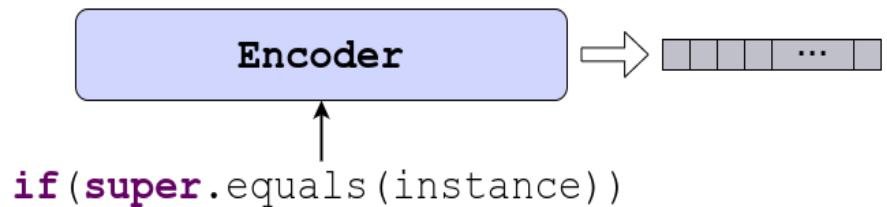
⚠ No guarantee

⚠ Empirical guarantee

# Code Editing with Encoder-Decoder

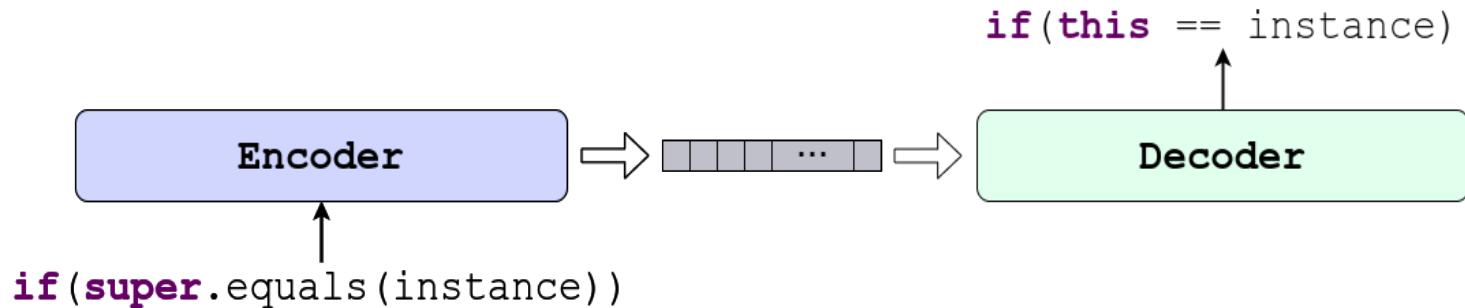


# Code Editing with Encoder-Decoder



**Encoder** encodes the input code to a vector or matrix.

# Code Editing with Encoder-Decoder

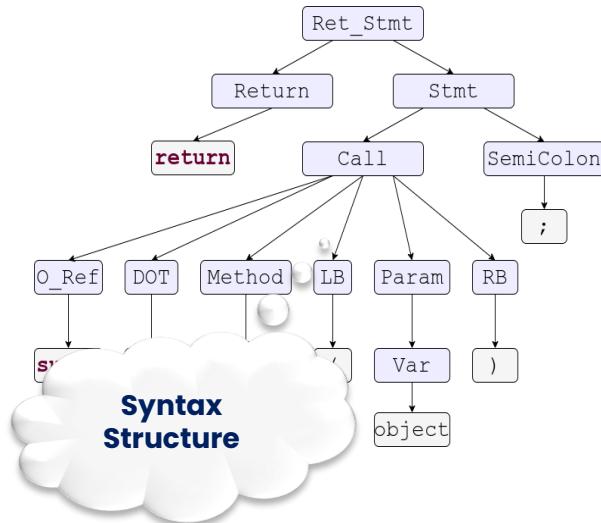


**Decoder** generates the edited code.

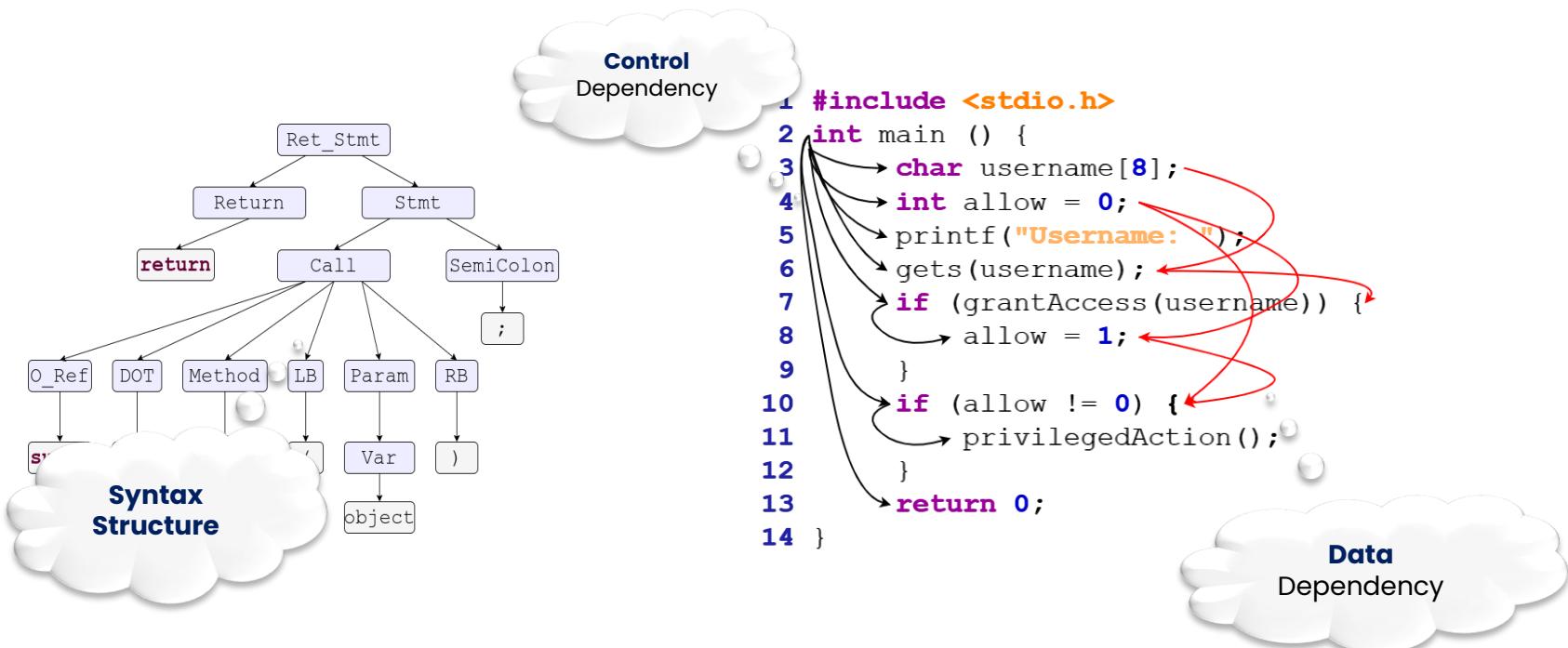
# Desired Properties of Encoder and Decoder

# Properties of Source Code.

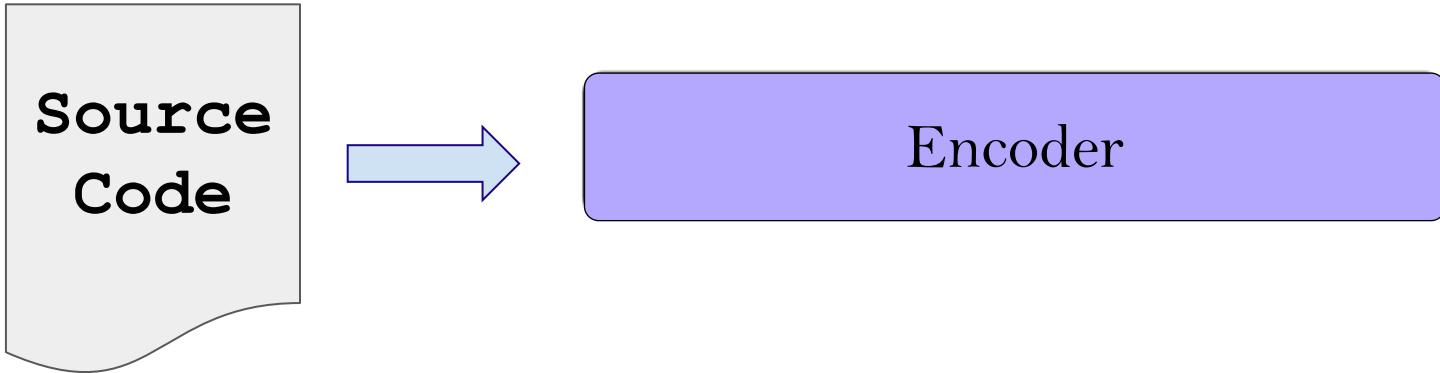
# Properties of Source Code.



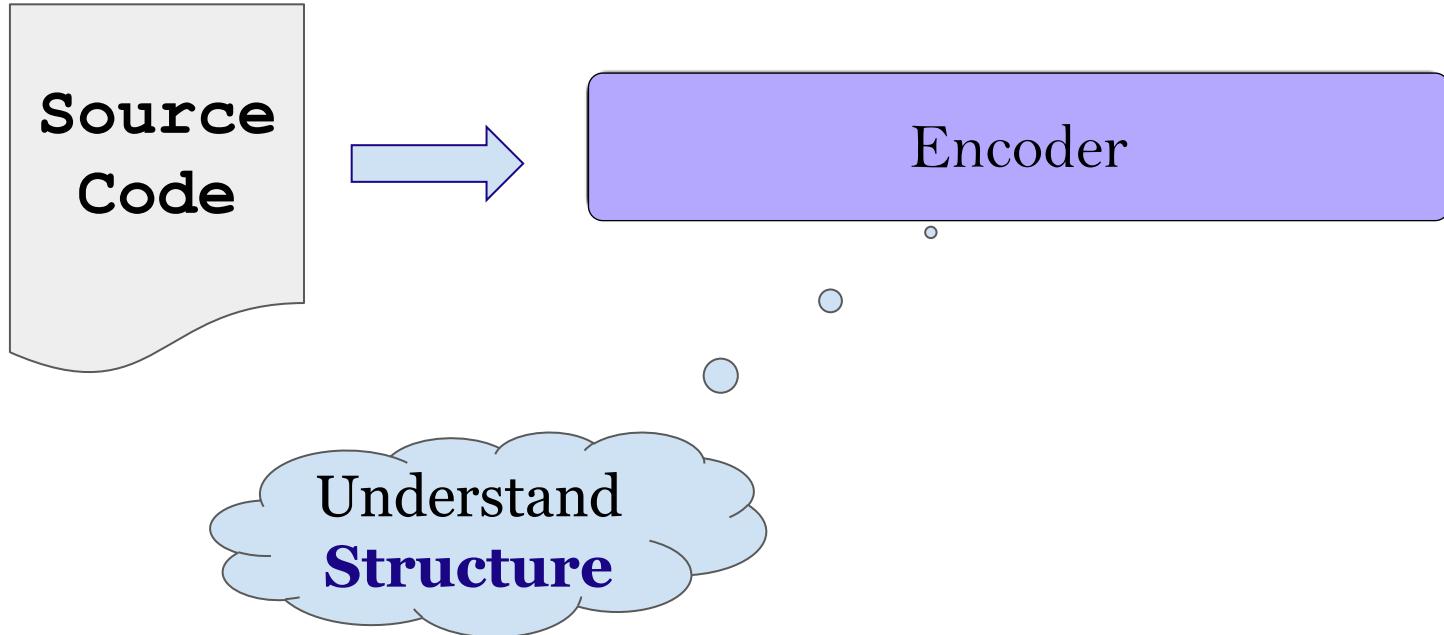
# Properties of Source Code.



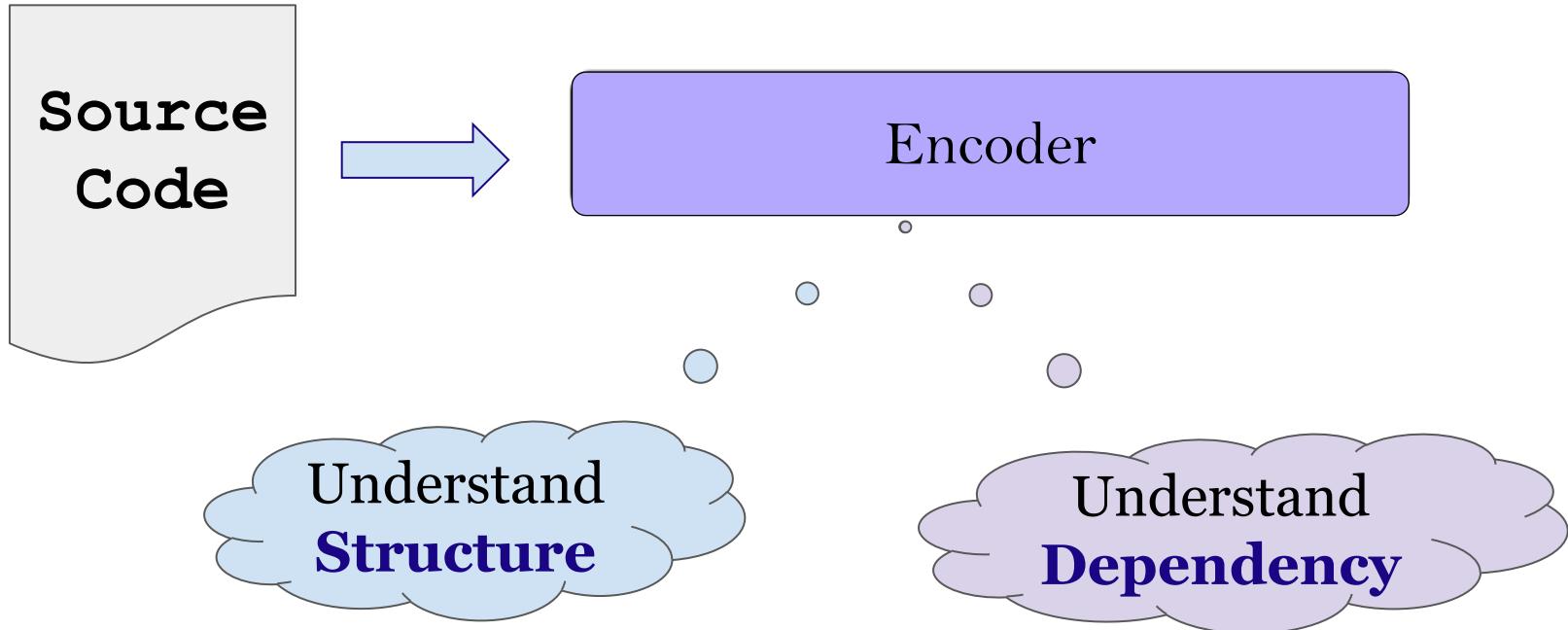
# Desired Properties of the Encoder



# Desired Properties of the Encoder



# Desired Properties of the Encoder



# Desired Properties of the Decoder

```
return this == object
```



Decoder

1. Syntactic correctness.
2. Contextual correctness.

# Desired Properties of the Decoder

```
return this == object
```

Decoder

1. Syntactic correctness.
2. Contextual correctness.

## Syntactically Incorrect

```
boolean f (Object target) {  
    for (Object elem : if.elements) {  
        if (elem.equals(target)) {  
            return true;  
        }  
    }  
    return false;  
}
```

# Desired Properties of the Decoder

```
return this == object
```

Decoder

1. Syntactic correctness.
2. Contextual correctness.

## Syntactically Incorrect

```
boolean f (Object target) {  
    for (Object elem : if.elements) {  
        if (elem.equals(target)) {  
            return true;  
        }  
    }  
    return false;  
}
```

# Desired Properties of the Decoder

```
return this == object
```

Decoder

1. Syntactic correctness.
2. Contextual correctness.

## Syntactically Incorrect

```
boolean f (Object target) {  
    for (Object elem : if.elements) {  
        if (elem.equals(target)) {  
            return true;  
        }  
    }  
    return false;  
}
```

## Contextually Incorrect

```
boolean f (Object target) {  
    for (Object elem : this.elements) {  
        if (elem.equals(f)) {  
            return null;  
        }  
    }  
    return false;  
}
```

# Desired Properties of the Decoder

```
return this == object
```

Decoder

## Syntactically Incorrect

```
boolean f (Object target) {  
    for (Object elem : if.elements) {  
        if (elem.equals(target)) {  
            return true;  
        }  
    }  
    return false;  
}
```

1. Syntactic correctness.
2. Contextual correctness.

## Contextually Incorrect

```
boolean f (Object target) {  
    for (Object elem : this.elements) {  
        if (elem.equals(f)) {  
            return null;  
        }  
    }  
    return false;  
}
```

# Desired Properties of the Decoder

```
return this == object
```

Decoder

1. Syntactic correctness.
2. Contextual correctness.

## Syntactically Incorrect

```
boolean f (Object target) {  
    for (Object elem : if.elements) {  
        if (elem.equals(target)) {  
            return true;  
        }  
    }  
    return false;  
}
```

## Contextually Incorrect

```
boolean f (Object target) {  
    for (Object elem : this.elements) {  
        if (elem.equals(f)) {  
            return null;  
        }  
    }  
    return false;  
}
```

# Code Editing as Understanding and Generation

# Code Editing as Understanding and Generation

## Understanding

### Source Code

Understanding Structure  
And Functionality of Source Code



## Generating

### Source Code

Ensuring the Syntactic and Semantic  
Correctness for Generating Source Code



# Code Editing as Understanding and Generation

## Understanding

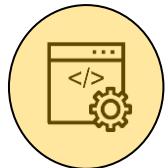
### Source Code

Understanding Structure  
And Functionality of Source Code

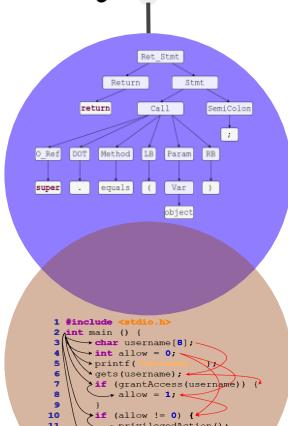


### Generating Source Code

Ensuring the Syntactic and Semantic  
Correctness for Generating Source Code



### Syntax



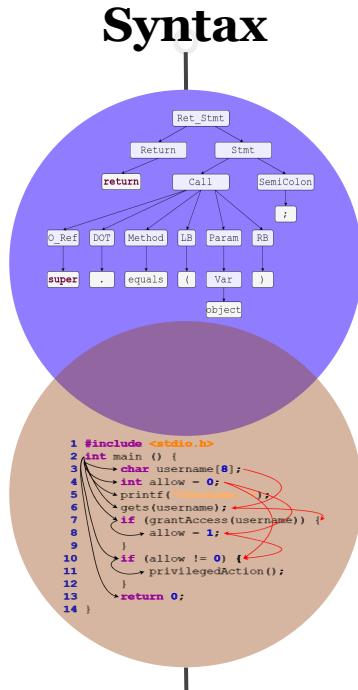
Encoding

### Semantics

```
1 #include <stdio.h>
2 int main () {
3     char username[8];
4     int allow = 0;
5     printf("Enter Username: ");
6     gets(username);
7     if (grantAccess(username)) {
8         allow = 1;
9     }
10    if (allow != 0) {
11        privilegedAction();
12    }
13 }
14 }
```

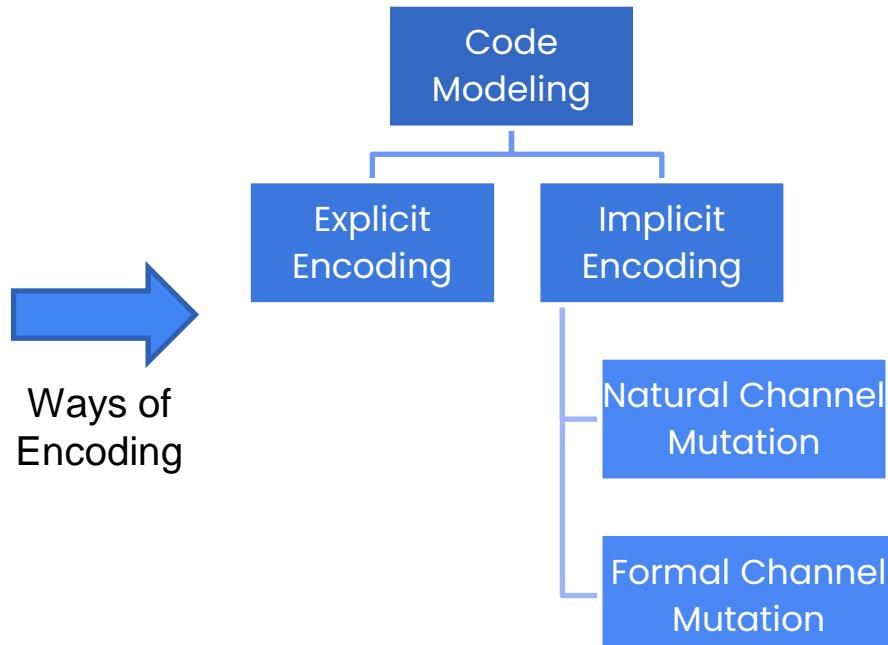


# Encoding PL Properties

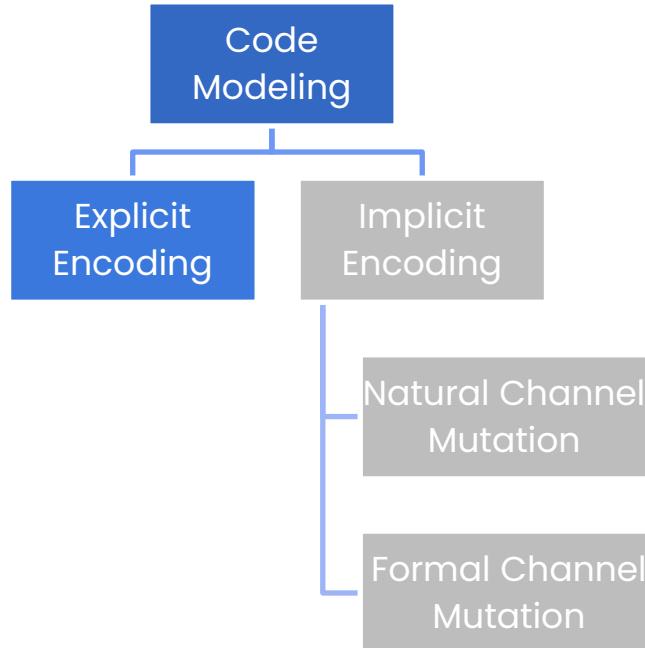


Semantics

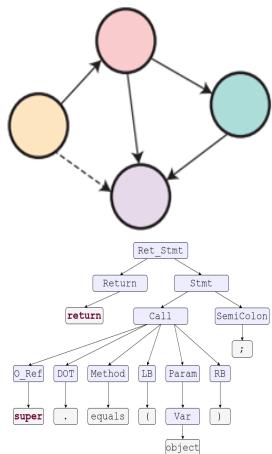
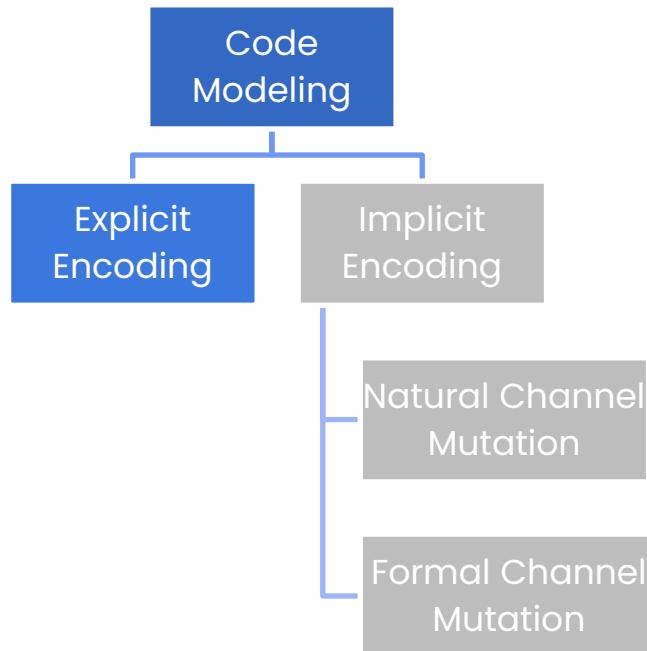
Ways of  
Encoding



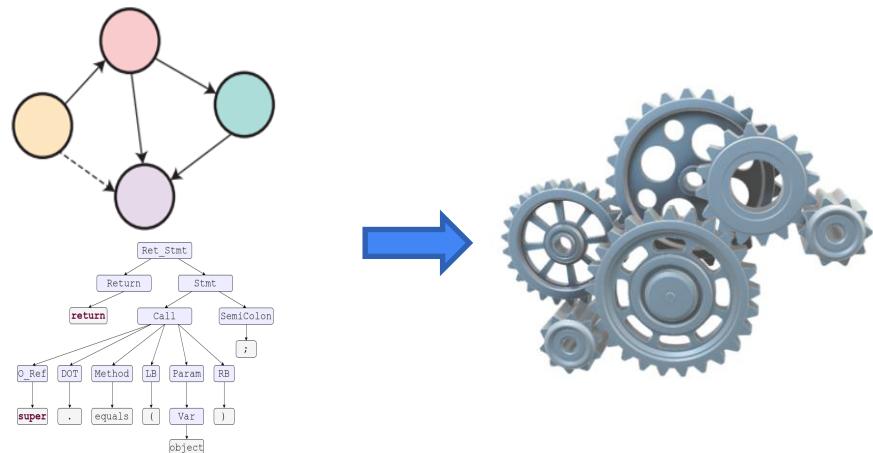
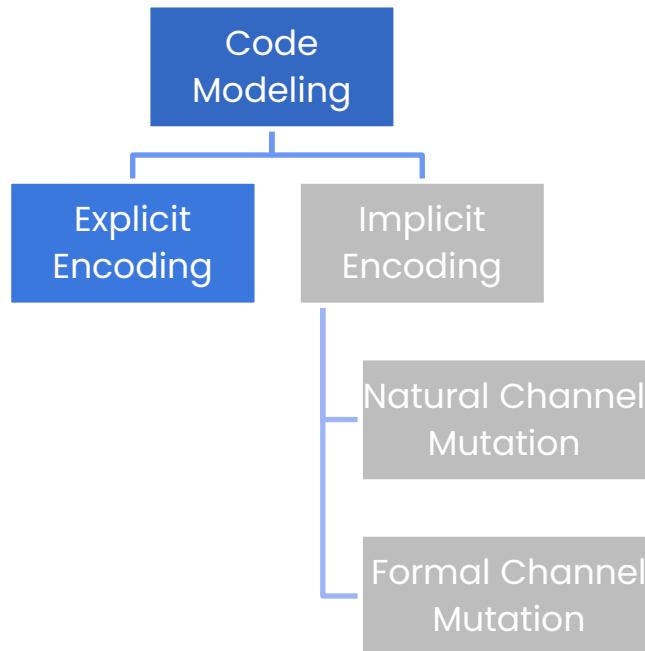
# Explicit Encoding



# Explicit Encoding



# Explicit Encoding



- [13] Learning to Represent Program as Graphs  
Allamanis et. al. 2017
- [14] Learning to Represent Edits  
Yin et. al. 2019
- [5] HOPPITY - Dinella et. al. 2020

# CODIT: Code Editing With Tree Based Neural Models

TSE - 2020

## Findings

Generation of Syntax Tree  
instead of code **Guarantees**  
**Syntactic Correctness.**

## Contribution

Tree/Grammar Based Model  
for Automatic Code Editing.

# CODIT: Code Editing With Tree Based Neural Models

# CODIT: Code Editing With Tree Based Neural Models

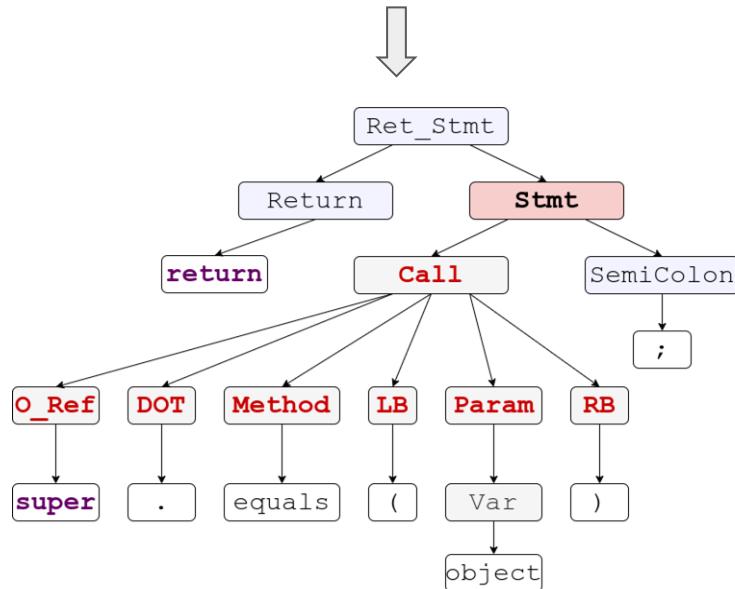
## Code Before Edit

```
return super.equals(object);
```

# CODIT: Code Editing With Tree Based Neural Models

## Code Before Edit

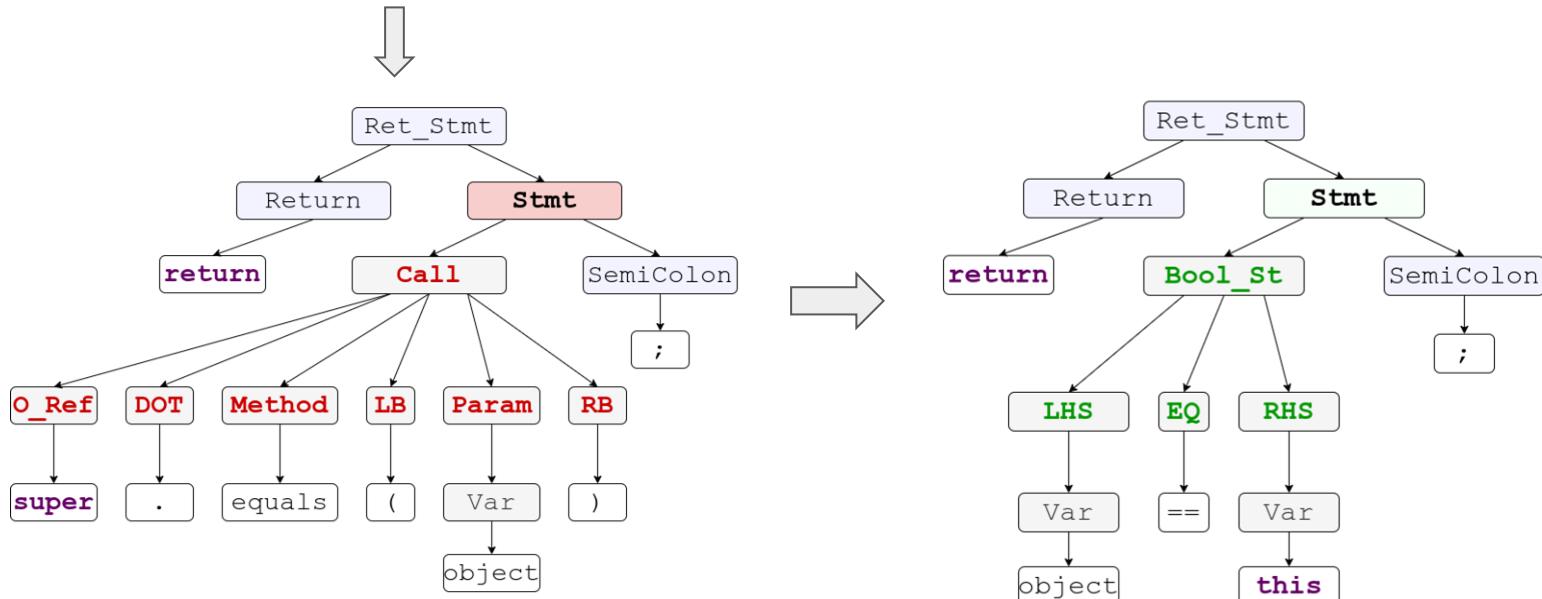
```
return super.equals(object);
```



# CODIT: Code Editing With Tree Based Neural Models

## Code Before Edit

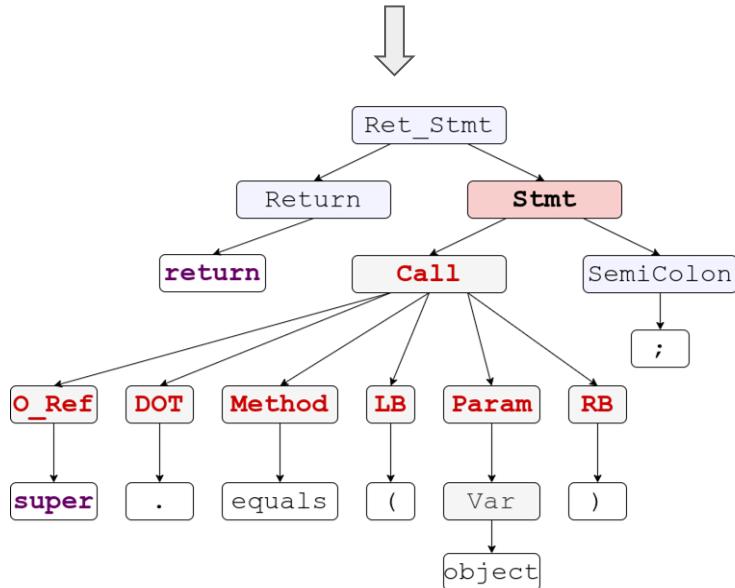
```
return super.equals(object);
```



# CODIT: Code Editing With Tree Based Neural Models

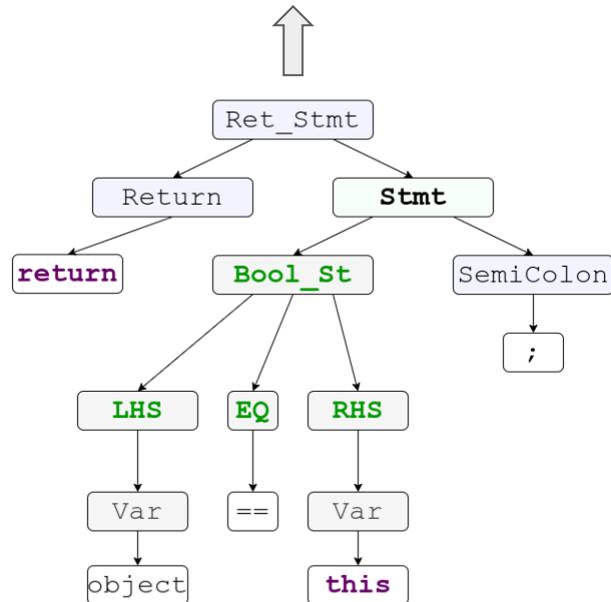
## Code Before Edit

```
return super.equals(object);
```



## Code After Edit

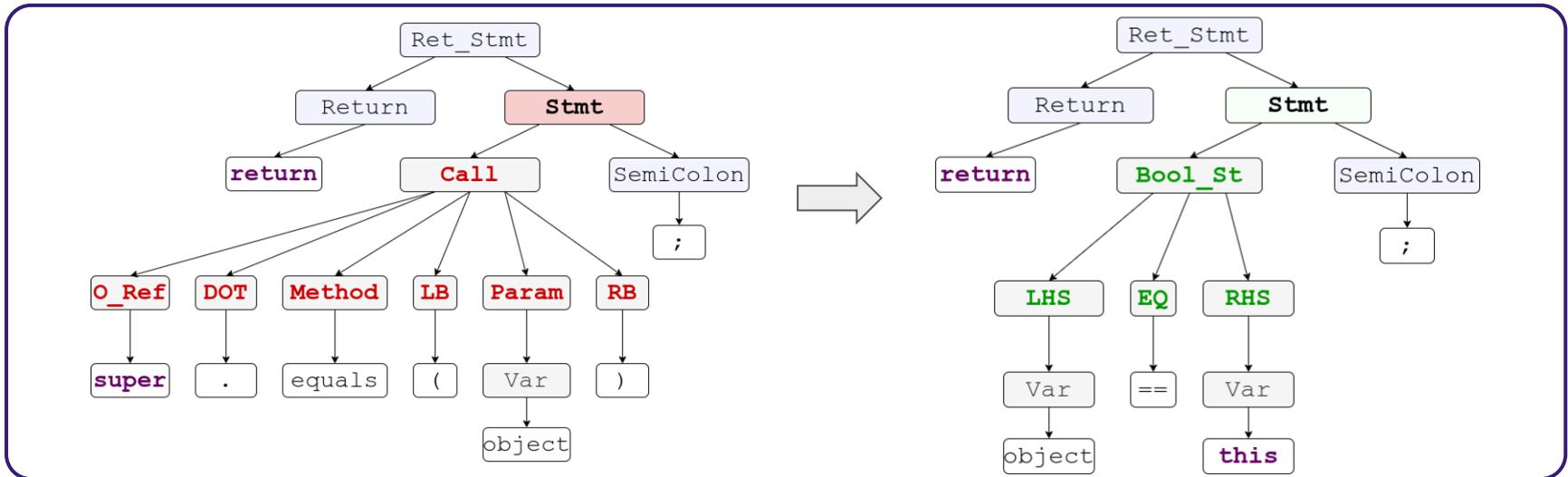
```
return this == object;
```



# CODIT: Code Editing With Tree Based Neural Models

## Code Before Edit

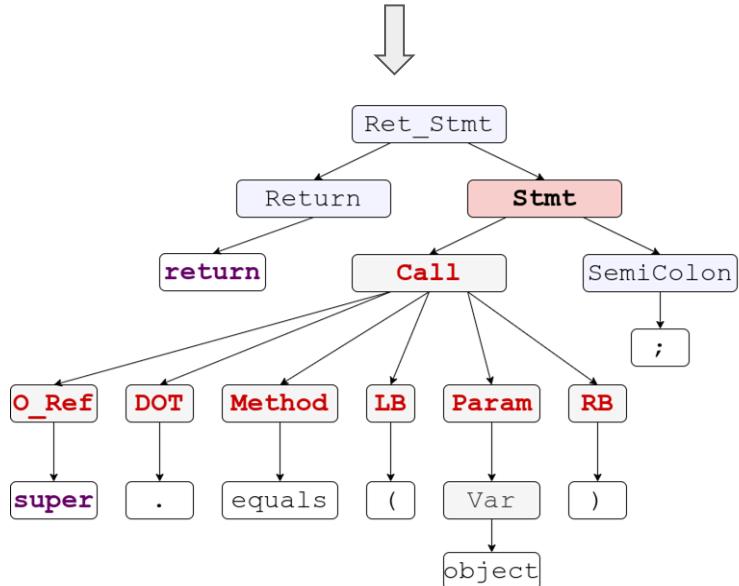
```
return super.equals(object);
```



# CODIT: Code Editing With Tree Based Neural Models

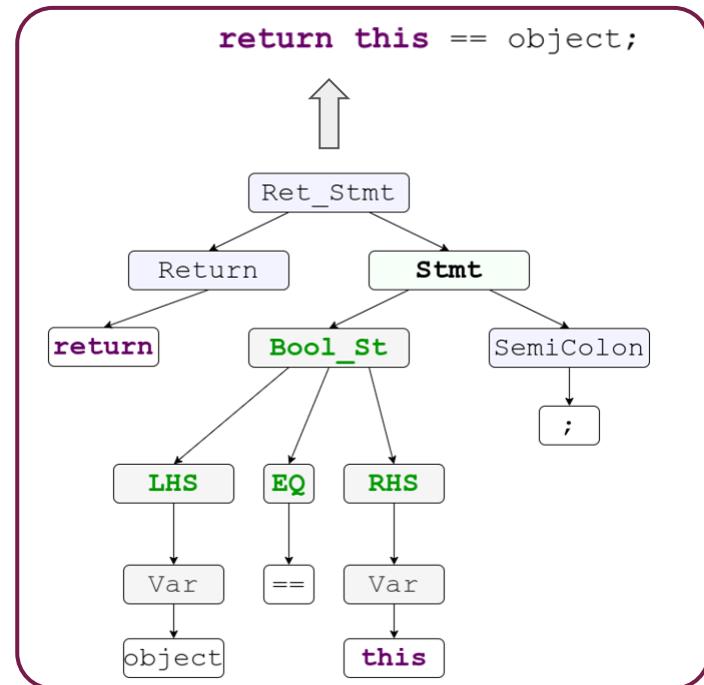
## Code Before Edit

```
return super.equals(object);
```

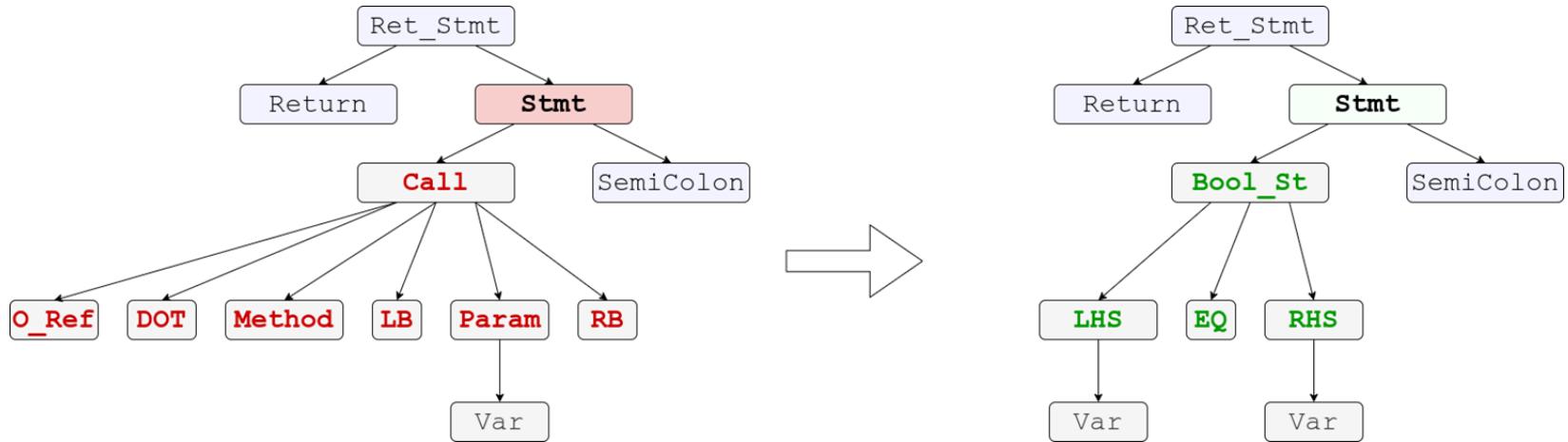


## Code After Edit

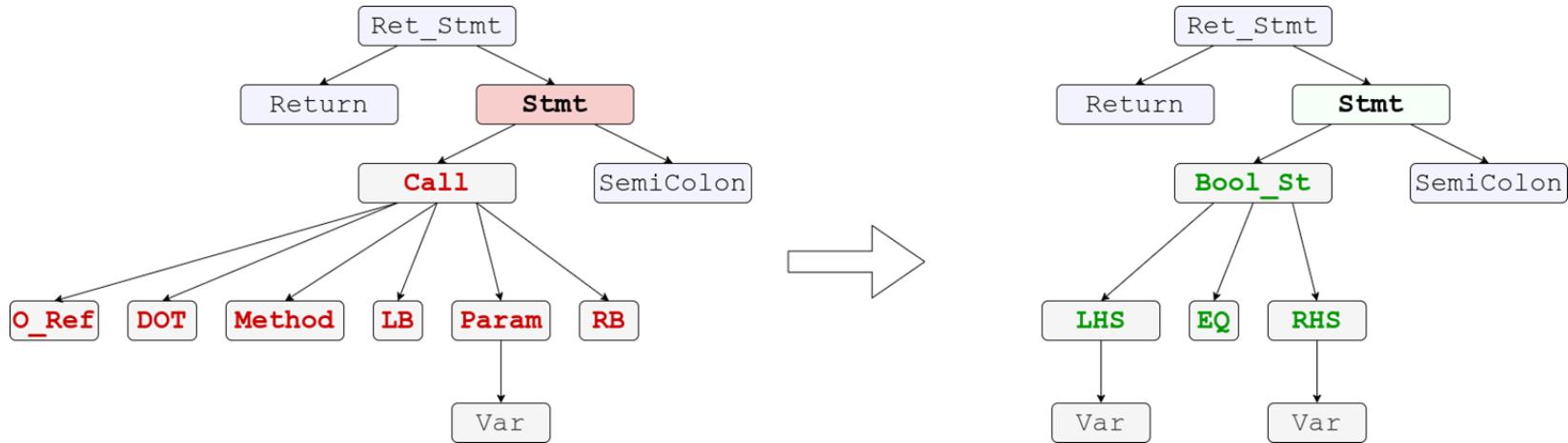
```
return this == object;
```



# CODIT Step 1 : Tree Translation



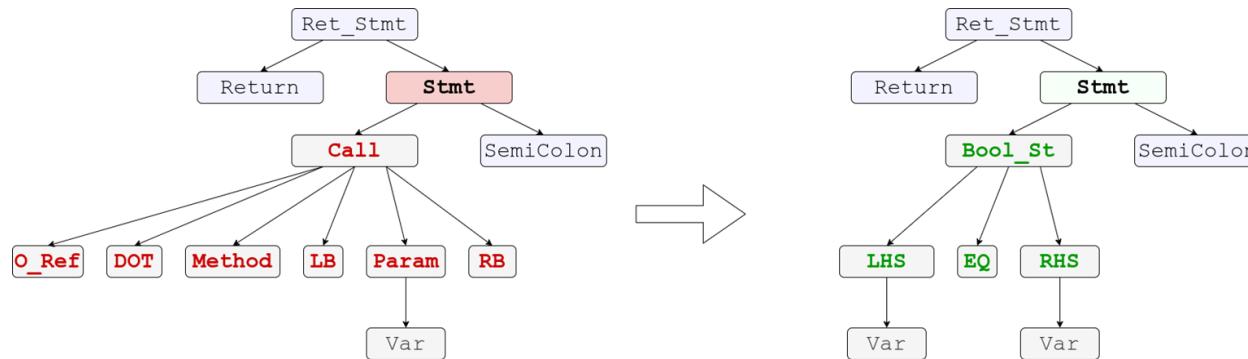
# CODIT Step 1 : Tree Translation



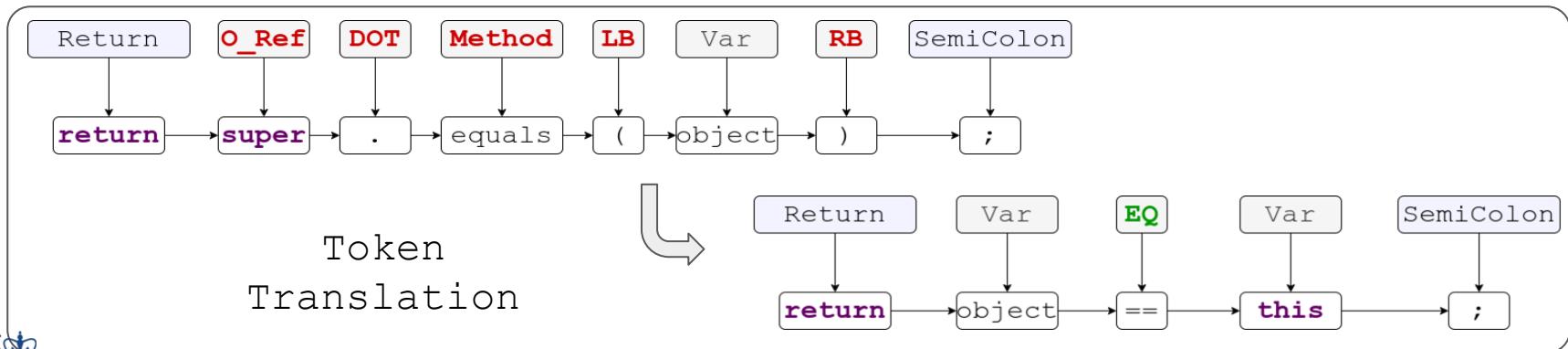
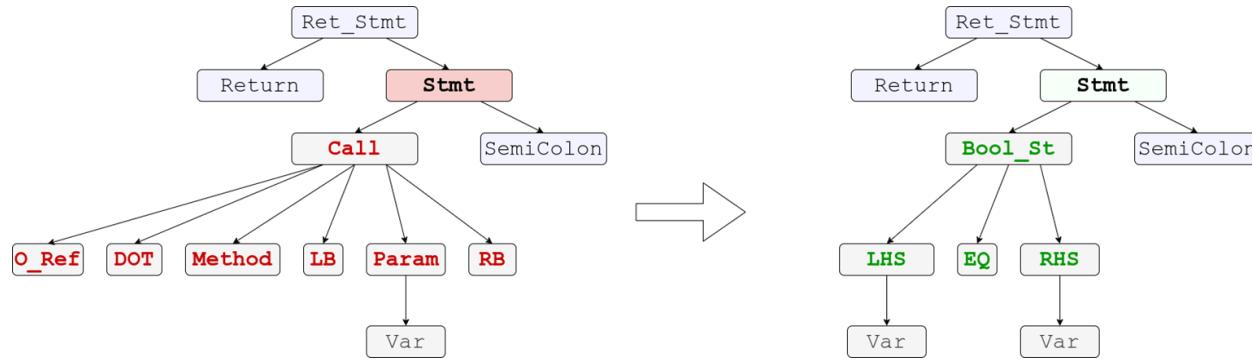
Ret\_Stmt → Return Stmt  
Stmt → Call Semicolon  
Call → O\_Ref DOT Method  
Param → LB Param RB  
Param → Var

Ret\_Stmt → Return Stmt  
Stmt → Bool\_St Semicolon  
Bool\_St → LHS EQ RHS  
LHS → VAR  
RHS → VAR

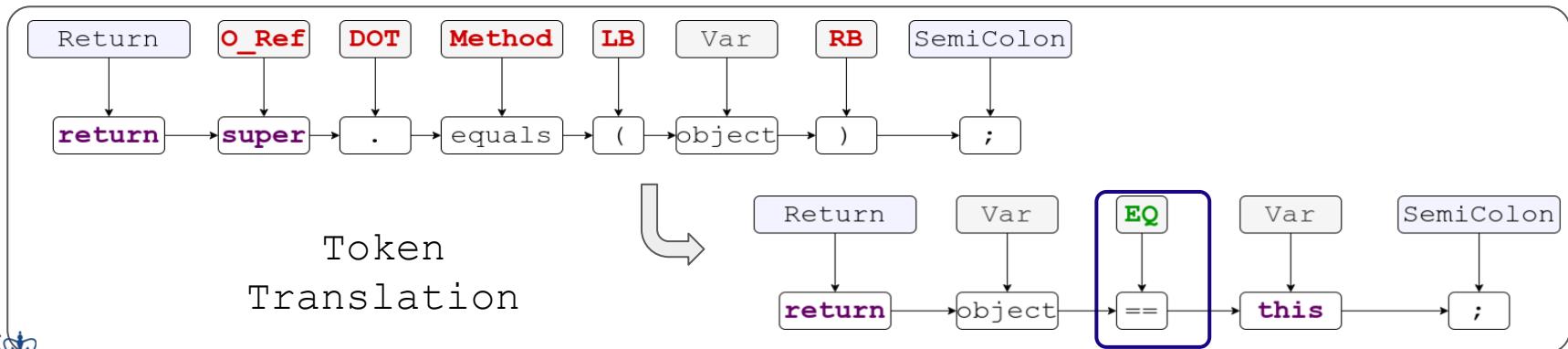
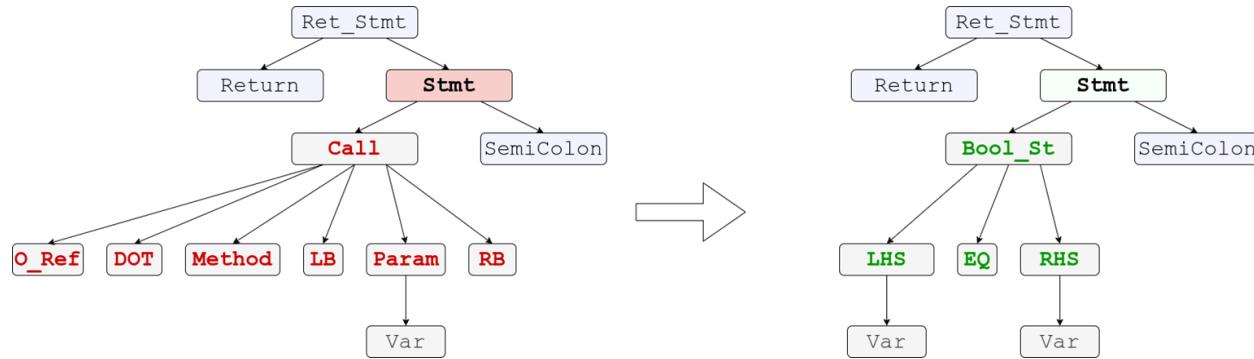
# CODIT Step 2 : Token Generation



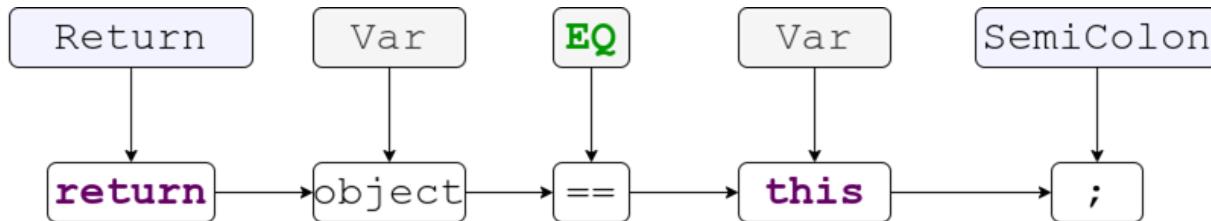
# CODIT Step 2 : Token Generation



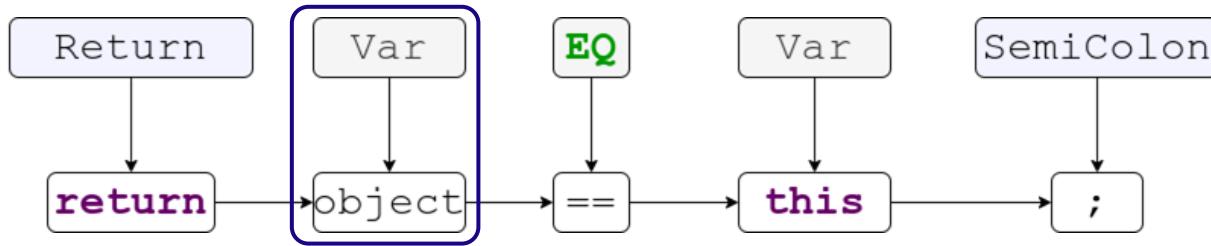
# CODIT Step 2 : Token Generation



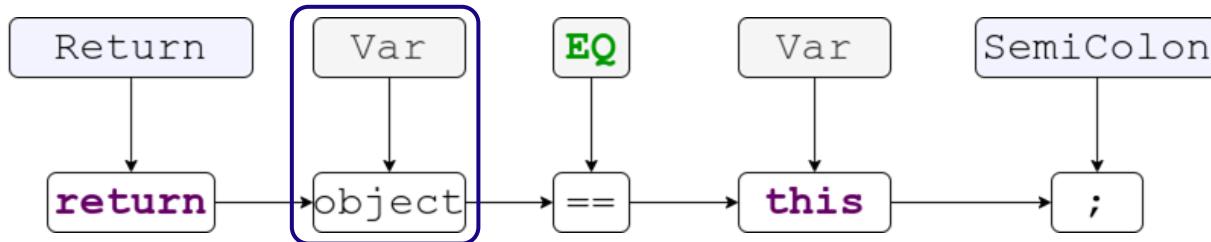
## CODIT Step 2 : Token Generation



## CODIT Step 2 : Token Generation



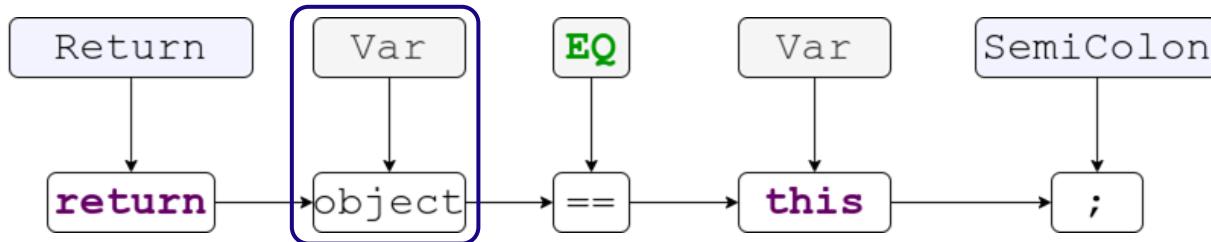
# CODIT Step 2 : Token Generation



```
public boolean checkEqual(
    Object inst, MyClass object){
    MyClass tmp = new MyClass();
    if (inst == null) {
        MyClass tmp2 = new MyClass();
        ...
    }
    return super.equals(object);
}
```

Reachability Analysis in Edit Location

# CODIT Step 2 : Token Generation



```
public boolean checkEqual(
    Object inst, MyClass object){
    MyClass tmp = new MyClass();
    if (inst == null) {
        MyClass tmp2 = new MyClass();
        ...
    }
    return super.equals(object);
}
```

Reachable Variables in Edit Location:

{inst, object, tmp}

Reachability Analysis in Edit Location

# CODIT Evaluation

# CODIT: Study Subjects

| Data Set                              | Number of Projects | Number of Edit Examples | Code Fragment Size   |                      |
|---------------------------------------|--------------------|-------------------------|----------------------|----------------------|
|                                       |                    |                         | Number of Tokens     | Number of Nodes      |
| Generic Code Edit from Github         | 48                 | 32,473                  | Max - 38<br>Avg - 15 | Max - 47<br>Avg - 20 |
| Pull Request Edits –Tufano et al. [7] | 3                  | 5546                    | Max - 34<br>Avg - 17 | Max - 47<br>Avg - 23 |

# CODIT: Results (Accuracy in top 5)

| Method         |                    | Generic Code Edits | Pull Request Edit |
|----------------|--------------------|--------------------|-------------------|
| Sequence Based | LSTM-Seq2Seq       | 3.77%              | 11.26%            |
|                | Tufano et. al. [7] | 6.57%              | 23.65%            |
|                | SequenceR [9]      | 9.76%              | 26.43%            |
| Tree Based     | Tree2Seq           | 11.04%             | 23.49%            |
|                | <b>CODIT</b>       | <b>15.94%</b>      | <b>28.87%</b>     |

# Example Edits

```
public void copyFrom( java.lang.Object arr){  
+    try{  
        android.os.Trace.traceBegin (...);  
+    finally{  
        android.os.Trace.traceEnd(...);  
+    }  
}
```

Addition

```
public abstract void removeSessionCookies (...)  
{  
    throw new android...MustOverrideException();  
}
```

Deletion

```
void visit (JSession * session , ...) throws Exception  
{  
    visit (((JNode) (* session)) , ...);  
}
```

Update

# Application - Automatic Program repair

CODIT fixes **15 bugs completely** and **10 bugs partially**, out of 80 bugs in **Defects4j**.

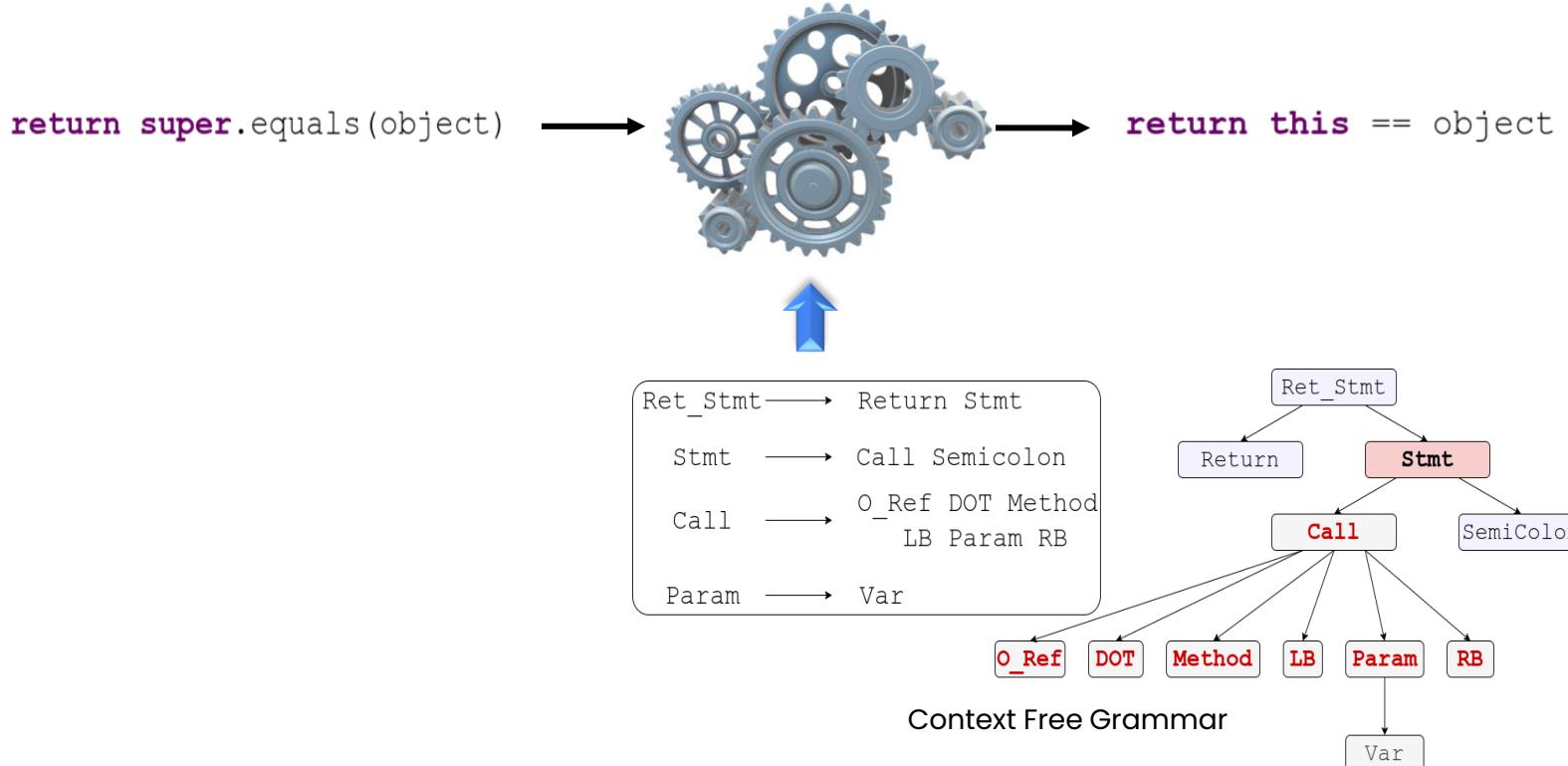
JFreeChart  
: Bug-8

```
7     public Week(Date time, TimeZone zone) {  
8         // defer argument checking...  
9         -     this(time, RegularTimePeriod.DEFAULT_TIME_ZONE, Locale.getDefault());  
10        +     this(time, zone, Locale.getDefault());  
11    }
```

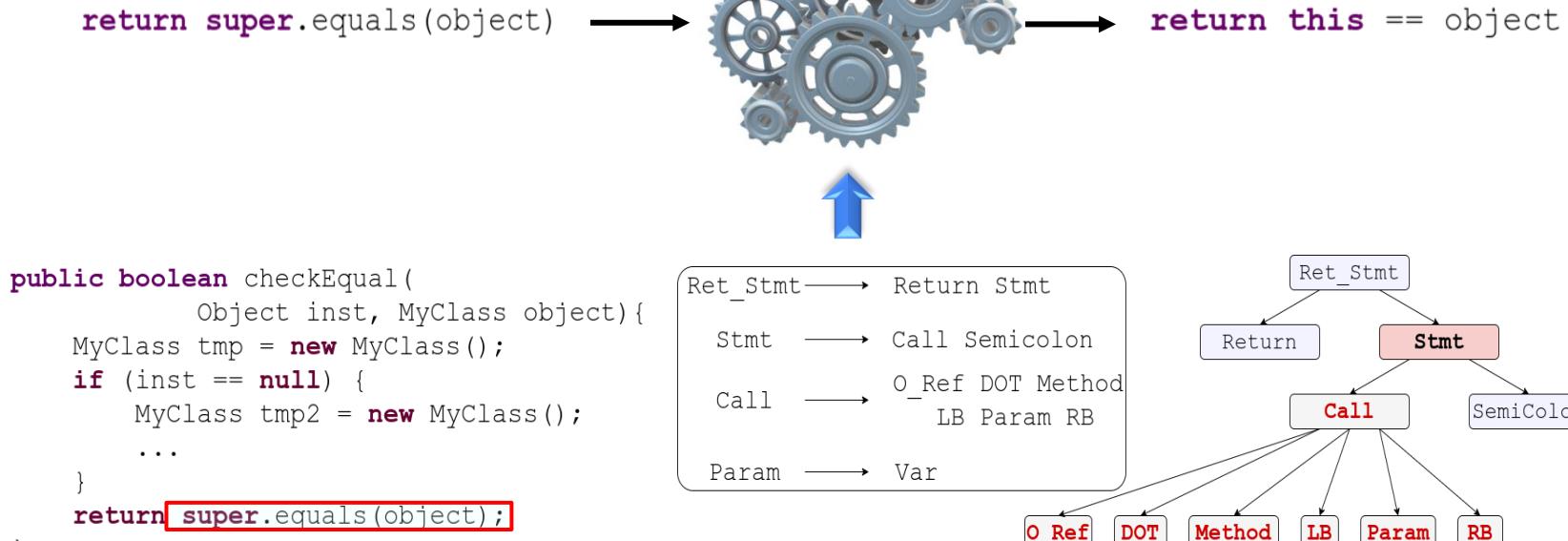
Closure  
Compiler:  
Bug-3

```
6     reachingUses = new MaybeReachingVariableUse(cfg, t.getScope(), compiler);  
7     reachingUses.analyze();  
8     for (Candidate c : candidates) {  
9         -     if (c.canInline()) {  
10            +     if (c.canInline(t.getScope())) {  
11                c.inlineVariable();
```

# Explicit Encoding

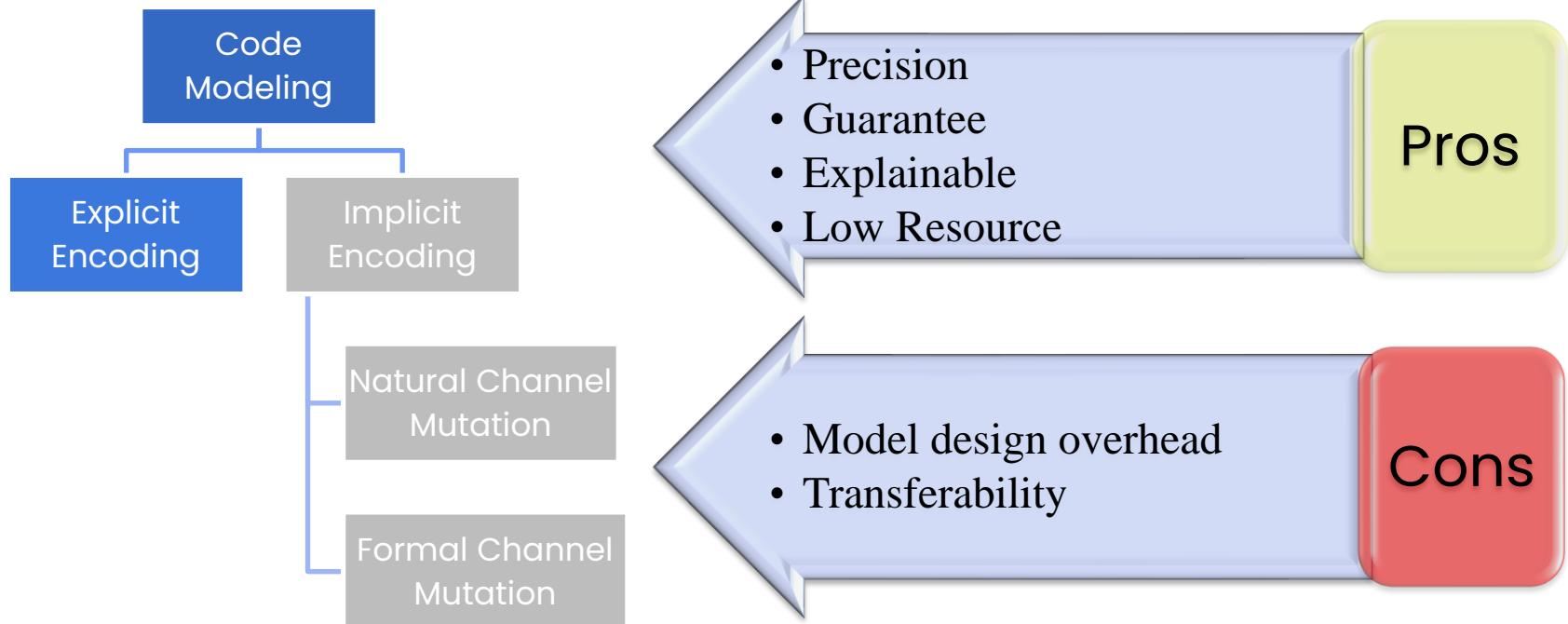


# Explicit Encoding

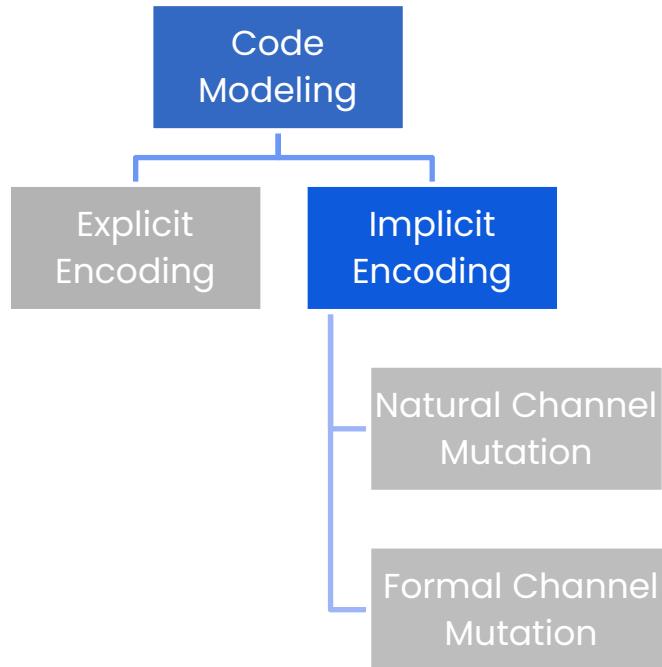


Reachability Analysis

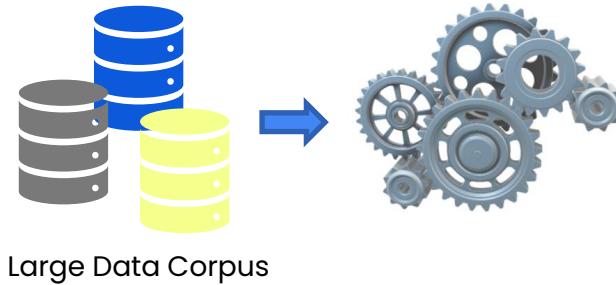
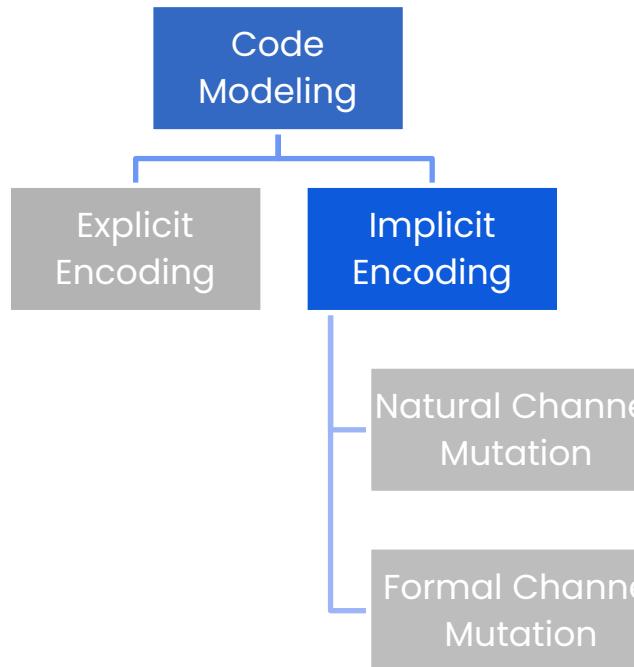
# Explicit Encoding – Take Away



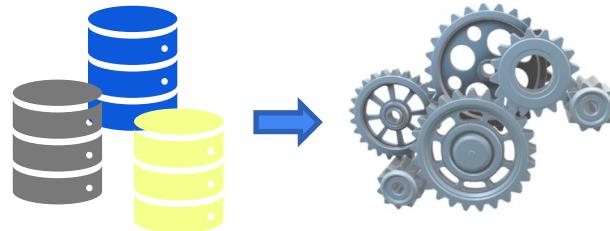
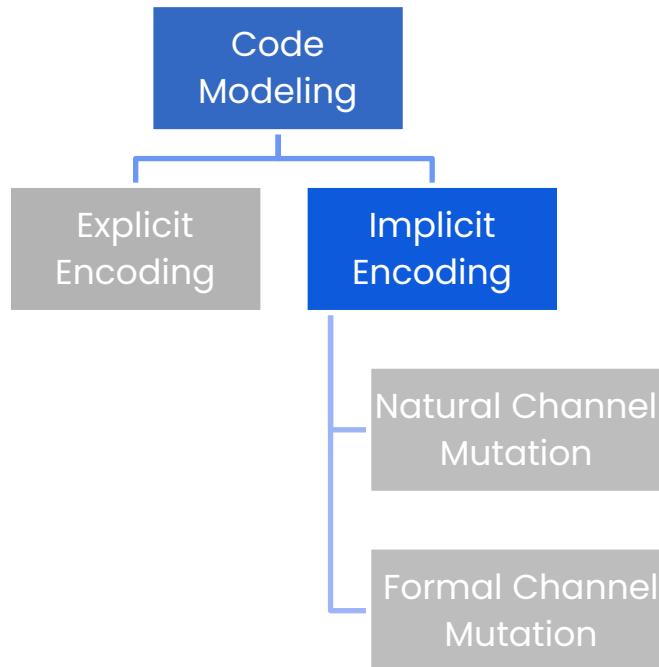
# Implicit Encoding



# Implicit Encoding



# Implicit Encoding



Large Data Corpus

```
printf [MASK] "Username: " ;  
gets ( username ) ;  
if ( [MASK]( username ) ) {  
    allow = 1;  
}  
if ( allow != 0 ) [MASK]  
    privilegedAction ( [MASK] ) ;  
}
```



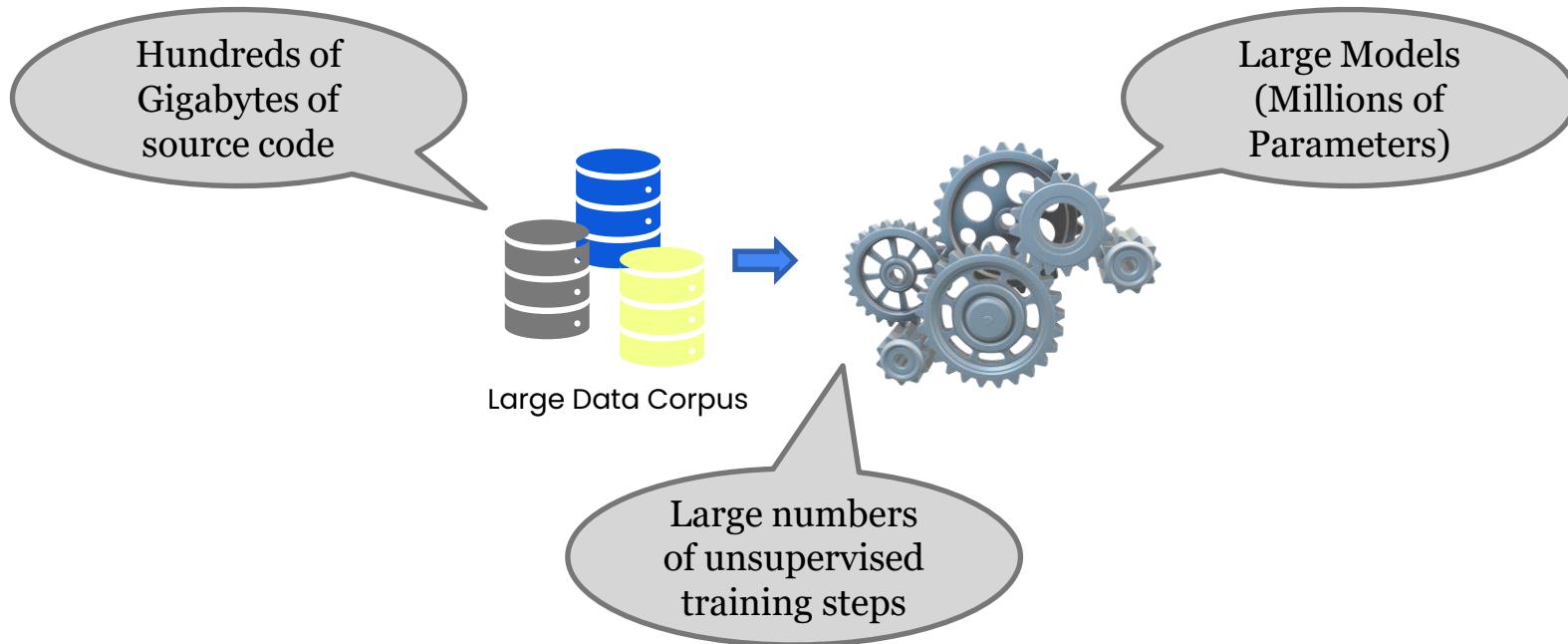
Mutated Input

```
printf ( "Username: " ) ;  
gets ( username ) ;  
if ( gets( username ) ) {  
    allow = 1;  
}  
if ( allow != 0 ) {  
    privilegedAction ( ) ;  
}
```

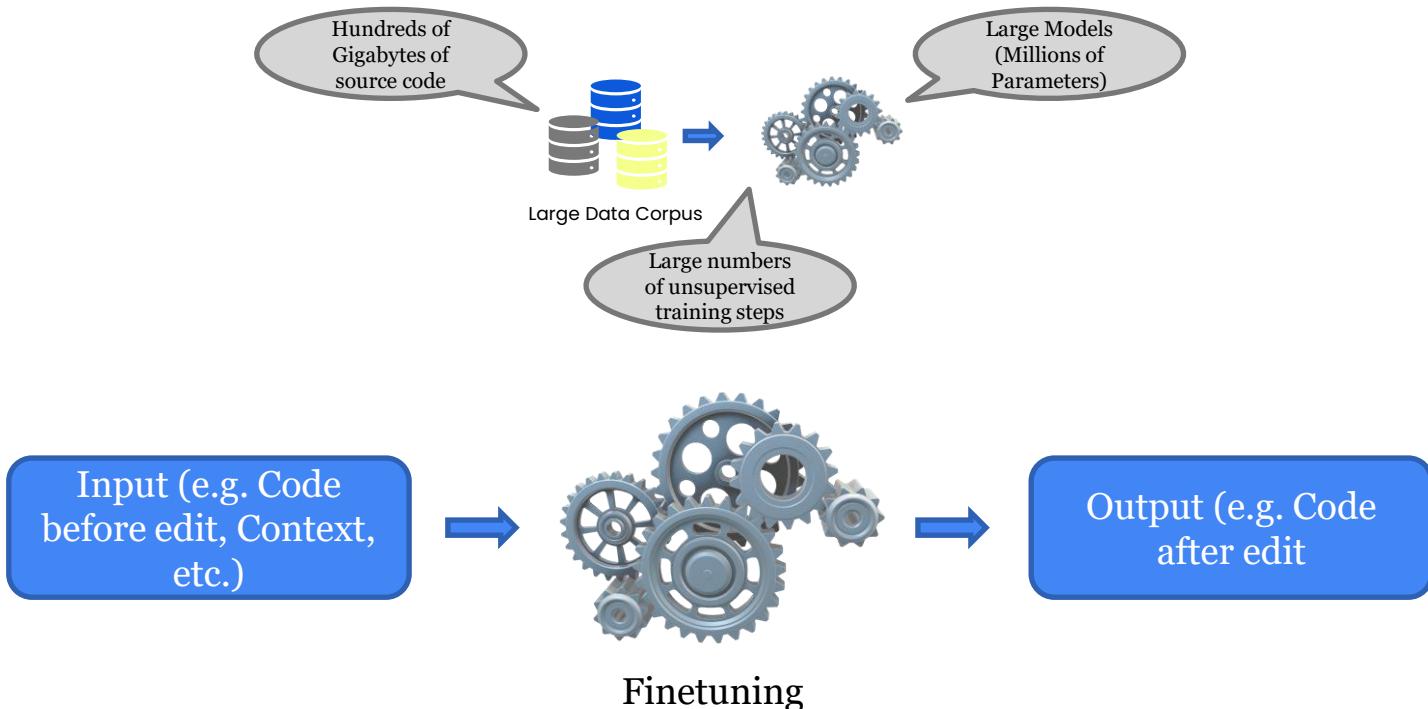
De-Mutation

- [11] CodeBERT - Feng et. al. 2019
- [16] GraphCodeBERT - Guo et. al. 2020
- [17] CodeX - Chen et. al. 2021
- [18] CodeT5 - Wang et. al. 2021

# Implicit Encoding - Pretraining



# Implicit Encoding - Finetuning



# Dual Channel Hypothesis for Source Code [15]

- [19] Casalanoovo et. al. 2020
- [20] Karampatsis et. al. 2020

# Dual Channel Hypothesis for Source Code [15]

```
1 #include <stdio.h>
2 int main () {
3     char username[8];
4     int allow = 0;
5     printf("Username: ");
6     gets(username);
7     if (grantAccess(username)) {
8         allow = 1;
9     }
10    if (allow != 0) {
11        privilegedAction();
12    }
13    return 0;
14 }
```

Natural Channel [19, 20]

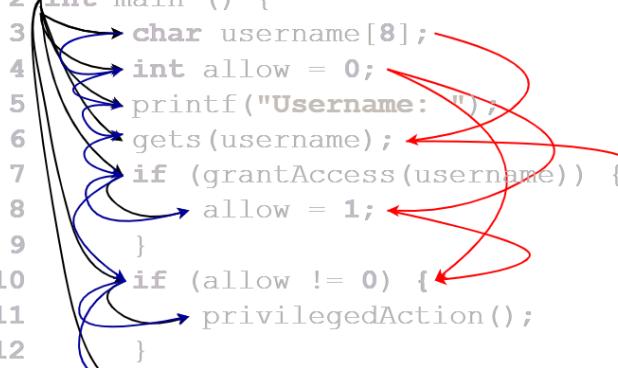
- [19] Casalanoovo et. al. 2020
- [20] Karampatsis et. al. 2020

# Dual Channel Hypothesis for Source Code [15]

```
1 #include <stdio.h>
2 int main () {
3     char username[8];
4     int allow = 0;
5     printf("Username: ");
6     gets(username);
7     if (grantAccess(username)) {
8         allow = 1;
9     }
10    if (allow != 0) {
11        privilegedAction();
12    }
13    return 0;
14 }
```

Natural Channel [19, 20]

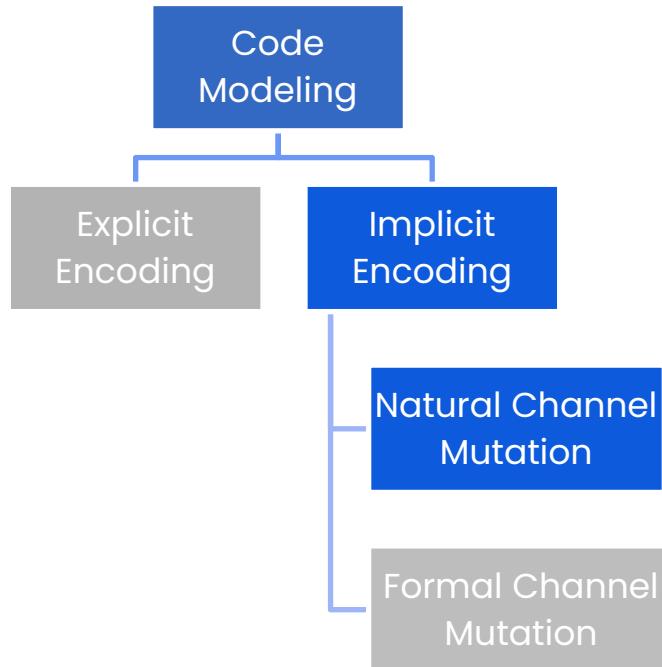
```
1 #include <stdio.h>
2 int main () {
3     char username[8];
4     int allow = 0;
5     printf("Username: ");
6     gets(username);
7     if (grantAccess(username)) {
8         allow = 1;
9     }
10    if (allow != 0) {
11        privilegedAction();
12    }
13    return 0;
14 }
```



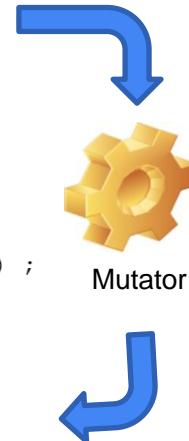
Formal Channel

- [19] Casalanoovo et. al. 2020  
[20] Karampatsis et. al. 2020

# Implicit Encoding



```
printf ("Username: " );
gets ( username ) ;
if ( gets( username ) ) {
    allow = 1 ;
}
if ( allow != 0 ) {
    privilegedAction ( ) ;
}
```



```
printf [MASK] "Username: " ;
gets ( username ) ;
if ( [MASK]( username ) ) {
    allow = 1;
}
if ( allow != 0 ) [MASK]
    privilegedAction ( [MASK] ;
}
```

# Unified Pre-training for Program Understanding and Generation (PLBART)

NAACL - 2021

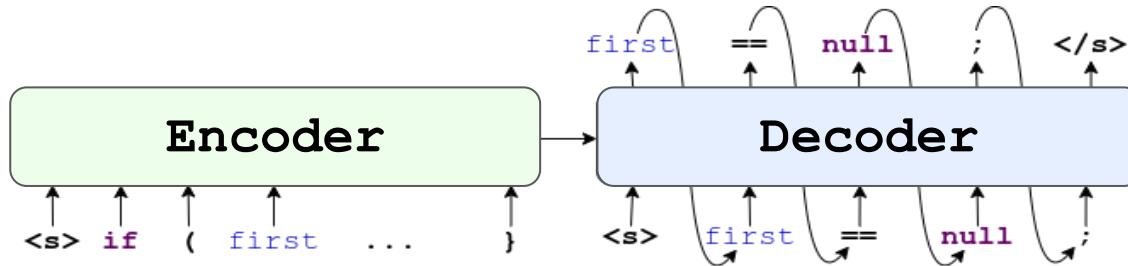
## Findings

Contextual Correctness can be learned from large scale source- code dataset.

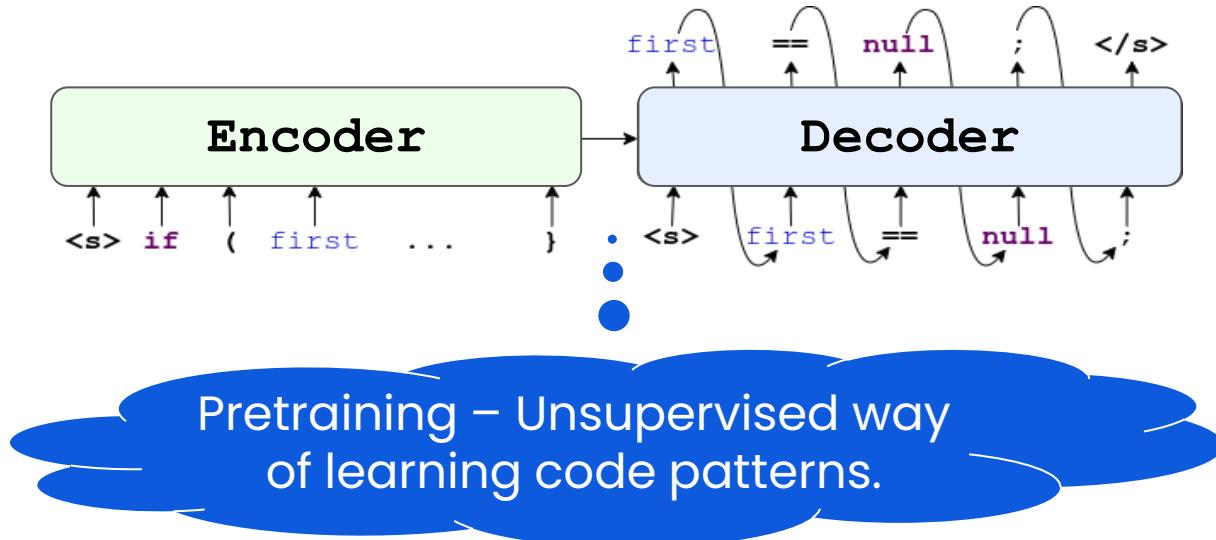
## Contribution

Developed large scale pre-trained models for different SE tasks.

# PLBART – What Is It?

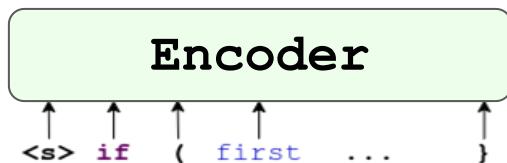


# PLBART – What Is It?



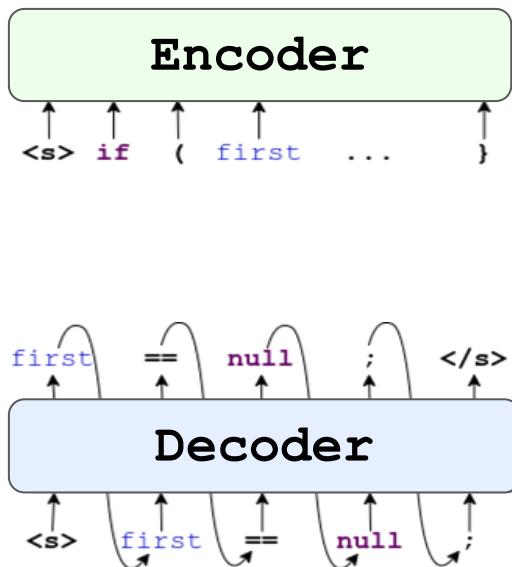
# PLBART – Components

# PLBART – Components



- 1. Read Code.**
- 2. Understands Code.**
- 3. Reason about any errors in code.**
- 4. Learns robust representation**

# PLBART – Components



- 1. Read Code.**
- 2. Understands Code.**
- 3. Reason about any errors in code.**
- 4. Learns robust representation**

1. Generate Code.
2. Learns **Coding Patterns.**

# PLBART – Pretraining

# PLBART – Pretraining

```
printf ( "Username: " ) ;
gets ( username ) ;
if ( gets( username ) ) {
    allow = 1 ;
}
if ( allow != 0 ) {
    privilegedAction ( ) ;
}
Correct Code
```



# PLBART – Pretraining

```
printf ( "Username: " ) ;  
gets ( username ) ;  
if ( gets( username ) ) {  
    allow = 1 ;  
}  
if ( allow != 0 ) {  
    privilegedAction ( ) ;  
}  
Correct Code
```



Noise Injector

```
printf ( "Username: " ) ;  
gets ( username ) ;  
if ( gets(  
    allow = 1 ;  
}  
if ( allow {  
    privilegedAction ( ) ;  
}  
Noisy Code
```

# PLBART – Pretraining

```
printf ( "Username: " ) ;  
gets ( username ) ;  
if ( gets( username ) ) {  
    allow = 1 ;  
}  
if ( allow != 0 ) {  
    privilegedAction ( ) ;  
}  
}
```

Correct Code



Noise Injector



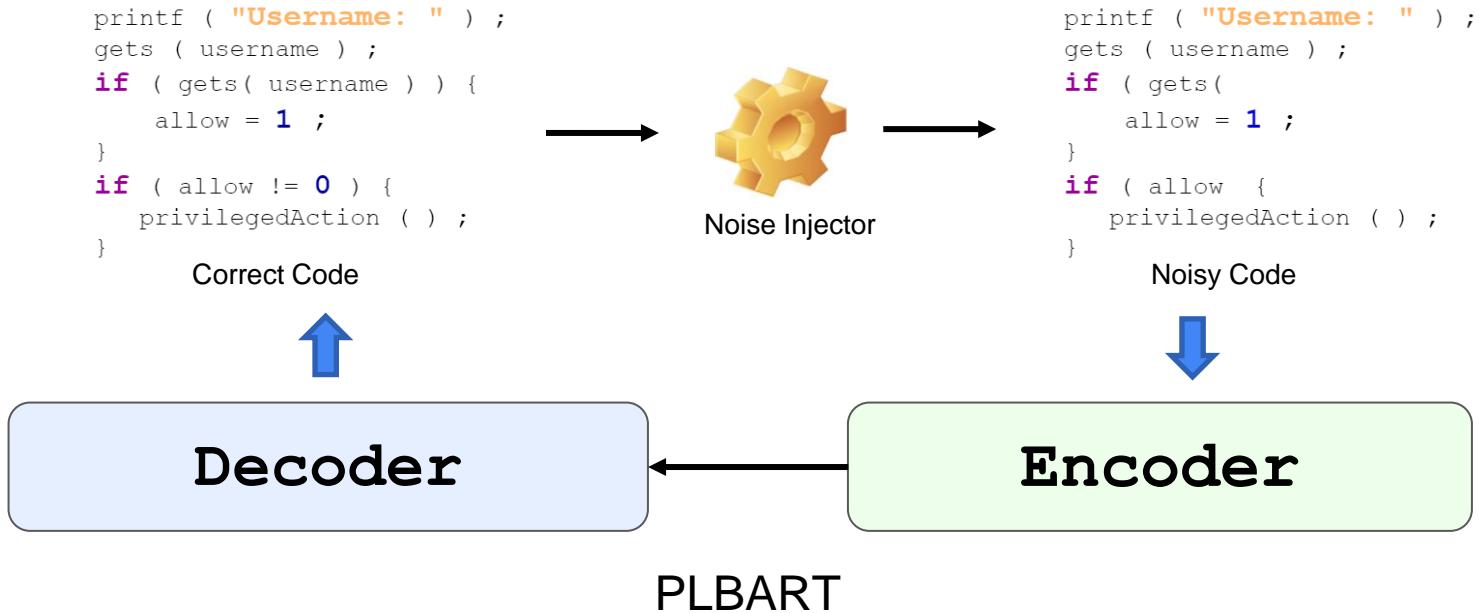
```
printf ( "Username: " ) ;  
gets ( username ) ;  
if ( gets(  
    allow = 1 ;  
}  
if ( allow {  
    privilegedAction ( ) ;  
}  
}
```

Noisy Code

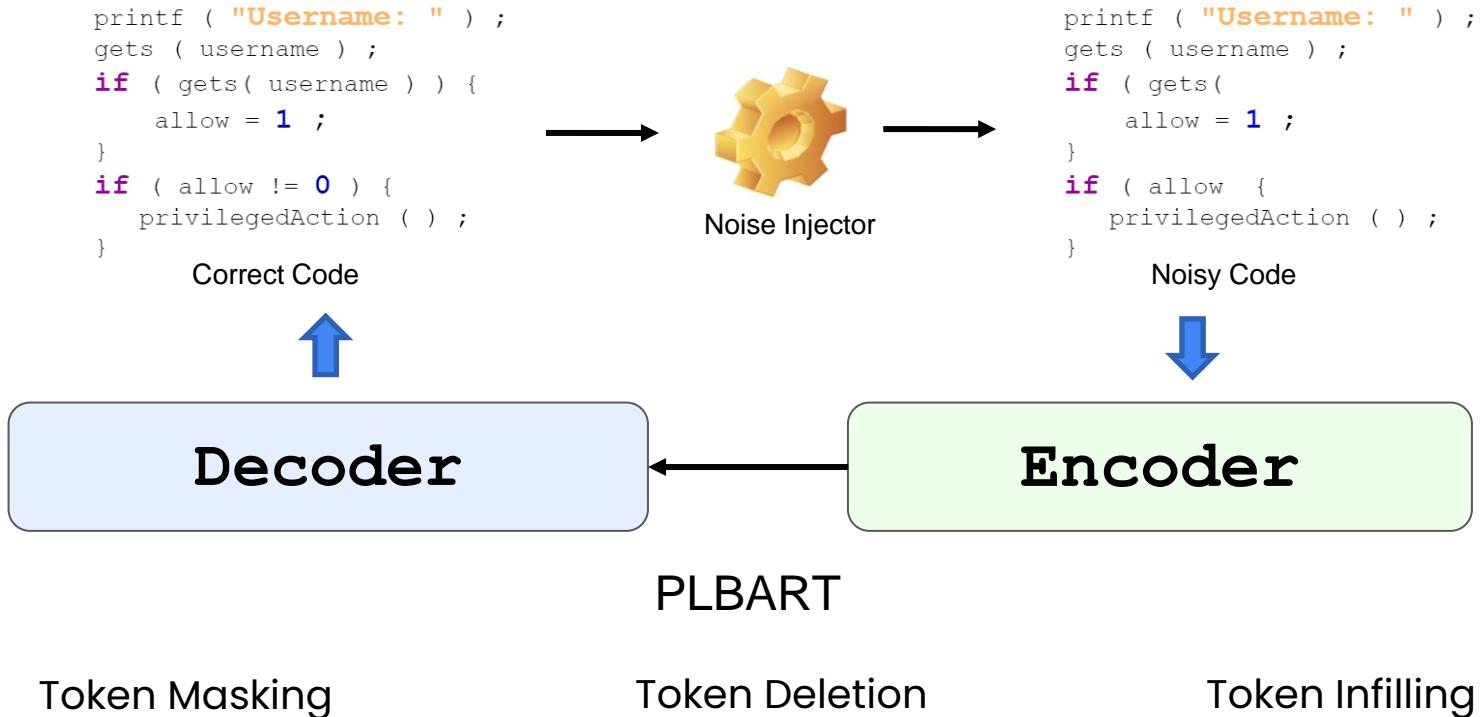


PLBART

# PLBART – Pretraining



# PLBART – Pretraining



# PLBART – Jointly Learning to Understand and Generate

## Token Masking

```
printf [MASK] "Username: " ) ;  
gets ( username ) ;  
if ( [MASK] ( username ) ) {  
    allow = 1;  
}  
if ( allow != 0 ) [MASK]  
    privilegedAction ( [MASK] ;  
}
```

# PLBART – Jointly Learning to Understand and Generate

## Token Masking

```
printf [MASK] "Username: " ) ;  
gets ( username ) ;  
if ( [MASK] ( username ) ) {  
    allow = 1;  
}  
if ( allow != 0 ) [MASK]  
    privilegedAction ( [MASK] ;  
}
```



# PLBART – Jointly Learning to Understand and Generate

## Token Masking

```
printf [MASK] "Username: " ) ;  
gets ( username ) ;  
if ( [MASK] ( username ) ) {  
    allow = 1;  
}  
if ( allow != 0 ) [MASK]  
    privilegedAction ( [MASK] ) ;  
}
```

```
printf ( "Username: " ) ;  
gets ( username ) ;  
if ( gets( username ) ) {  
    allow = 1;  
}  
if ( allow != 0 ) {  
    privilegedAction ( ) ;  
}
```



# PLBART – Jointly Learning to Understand and Generate

## Token Deletion

```
printf ( "Username: " ) ;
gets ( ) ;
if ( gets( username ) ) {
    allow = 1

if ( allow = 0 ) {
    privilegedAction ( ) ;
}
```

# PLBART – Jointly Learning to Understand and Generate

## Token Deletion

```
printf ( "Username: " ) ;  
gets ( ) ;  
if ( gets( username ) ) {  
    allow = 1  
  
if ( allow = 0 ) {  
    privilegedAction ( ) ;  
}
```



# PLBART – Jointly Learning to Understand and Generate

## Token Deletion

```
printf ( "Username: " ) ;  
gets ( ) ;  
if ( gets( username ) ) {  
    allow = 1  
  
if ( allow = 0 ) {  
    privilegedAction ( ) ;  
}
```

```
printf ( "Username: " ) ;  
gets ( username ) ;  
if ( gets( username ) ) {  
    allow = 1 ;  
}  
if ( allow != 0 ) {  
    privilegedAction ( ) ;  
}
```



# PLBART – Jointly Learning to Understand and Generate

## Token Infilling

```
printf ( "Username: " ) ;
gets ( username ) ;
if ( gets (
    allow = 1 ;
)
if ( allow {
    privilegedAction ( ) ;
}
```

# PLBART – Jointly Learning to Understand and Generate

## Token Infilling

```
printf ( "Username: " ) ;  
gets ( username ) ;  
if ( gets (  
    allow = 1 ;  
}  
if ( allow {  
    privilegedAction ( ) ;  
}
```



# PLBART – Jointly Learning to Understand and Generate

## Token Infilling

```
printf ( "Username: " ) ;  
gets ( username ) ;  
if ( gets (  
    allow = 1 ;  
}  
if ( allow {  
    privilegedAction ( ) ;  
}
```

```
printf ( "Username: " ) ;  
gets ( username ) ;  
if ( gets( username ) ) {  
    allow = 1 ;  
}  
if ( allow != 0 ) {  
    privilegedAction ( ) ;  
}
```



# PLBART – Pretraining

- Noise Properties
  - Mutate Natural Channel
  - Likely to Break Syntax

```
printf ( "Username: " ) ;
gets ( ) ;
if ( gets( username ) ) {
    allow = 1

if ( allow = 0 ) {
    privilegedAction ( ) ;
}
```

# PLBART – Pretraining

- Noise Properties
  - Mutate Natural Channel
  - Likely to Break Syntax

```
printf ( "Username: " ) ;
gets ( ) ;
if ( gets( username ) ) {
    allow = 1

if ( allow = 0 ) {
    privilegedAction ( ) ;
}
```

- Training Objectives
  - Generate the whole code.
  - Learns syntax implicitly
  - Learn Coding Patterns.
- Multi-lingual Training
  - Java
  - Python
  - NL from Stack overflow

# PLBART – Pretraining

- Noise Properties
  - Mutate Natural Channel
  - Likely to Break Syntax

```
printf ( "Username: " ) ;
gets ( ) ;
if ( gets( username ) ) {
    allow = 1

if ( allow = 0 ) {
    privilegedAction ( ) ;
}
```

- Training Objectives
  - Generate the whole code.
  - Learns syntax implicitly
  - Learn Coding Patterns.
- Multi-lingual Training
  - Java
  - Python
  - NL from Stack overflow

# PLBART Evaluation

# Downstream Tasks

| Task           | Dataset  | Language               | Train   | Valid  | Test    |
|----------------|--|------------------------|---------|--------|---------|
| Summarization  | <a href="#">Husain et al. (2019)</a>                             | Ruby                   | 24,927  | 1,400  | 1,261   |
|                |  | Javascript             | 58,025  | 3,885  | 3,291   |
|                |  | Go                     | 167,288 | 7,325  | 8,122   |
|                |  | Python                 | 251,820 | 13,914 | 14,918  |
|                |  | Java                   | 164,923 | 5,183  | 10,955  |
|                |  | PHP                    | 241,241 | 12,982 | 14,014  |
| Generation     | <a href="#">Iyer et al. (2018)</a>                               | NL to Java             | 100,000 | 2,000  | 2,000   |
| Translation    | Code-Code ( <a href="#">Lu et al., 2021</a> )                    | Java to C#             | 10,300  | 500    | 1,000   |
|                |  | C# to Java             | 10,300  | 500    | 1,000   |
|                | Program Repair<br>( <a href="#">Tufano et al., 2019</a> )        | Java <sub>small</sub>  | 46,680  | 5,835  | 5,835   |
|                |  | Java <sub>medium</sub> | 52,364  | 6,545  | 6,545   |
| Classification | Vulnerability Detection<br>( <a href="#">Zhou et al., 2019</a> ) | C/C++                  | 21,854  | 2,732  | 2,732   |
|                | Clone Detection<br>( <a href="#">Wang et al., 2020</a> )         | Java                   | 100,000 | 10,000 | 415,416 |

# Downstream Tasks

| Task           | Dataset  | Language               | Train   | Valid  | Test    |
|----------------|--|------------------------|---------|--------|---------|
| Summarization  | <a href="#">Husain et al. (2019)</a>                             | Ruby                   | 24,927  | 1,400  | 1,261   |
|                |  | Javascript             | 58,025  | 3,885  | 3,291   |
|                |  | Go                     | 167,288 | 7,325  | 8,122   |
|                |  | Python                 | 251,820 | 13,914 | 14,918  |
|                |  | Java                   | 164,923 | 5,183  | 10,955  |
|                |  | PHP                    | 241,241 | 12,982 | 14,014  |
| Generation     | <a href="#">Iyer et al. (2018)</a>                               | NL to Java             | 100,000 | 2,000  | 2,000   |
| Translation    | Code-Code ( <a href="#">Lu et al., 2021</a> )                    | Java to C#             | 10,300  | 500    | 1,000   |
|                |  | C# to Java             | 10,300  | 500    | 1,000   |
|                | Program Repair<br>( <a href="#">Tufano et al., 2019</a> )        | Java <sub>small</sub>  | 46,680  | 5,835  | 5,835   |
|                |  | Java <sub>medium</sub> | 52,364  | 6,545  | 6,545   |
| Classification | Vulnerability Detection<br>( <a href="#">Zhou et al., 2019</a> ) | C/C++                  | 21,854  | 2,732  | 2,732   |
|                | Clone Detection<br>( <a href="#">Wang et al., 2020</a> )         | Java                   | 100,000 | 10,000 | 415,416 |

# Code Editing (Program Repair) Result

**Dataset:** Bugfix dataset, Tufano *et. al.* 2019 [7] – Abstract Edit  
**Metric:** EM (exact match) in Top-1 position

**Small - Up to 50 tokens**  
**Medium - 51-100 tokens**

| Methods       | Small        | Medium      |
|---------------|--------------|-------------|
|               | EM           | EM          |
| Naive Copy    | 0            | 0           |
| Seq2Seq       | 10.00        | 2.50        |
| Transformer   | 14.70        | 3.70        |
| CodeBERT      | 16.40        | 5.16        |
| GraphCodeBERT | 17.30        | <b>9.10</b> |
| PLBART        | <b>19.21</b> | 8.98        |

```
    if  (
-      newJson.charAt(1) != wrappingQuote
+      !jsonObject.isEmpty() &&
+      (newJson.charAt(1) != wrappingQuote)
    ) {
```

```
// Guidance: fix problem which occurred when  
// the resulting json is empty ...
```

```
if ( - newJson.charAt(1) != wrappingQuote  
+ !jsonObject.isEmpty() &&  
+ (newJson.charAt(1) != wrappingQuote)  
){
```



## Developers Guidance

```
// Guidance: fix problem which occurred when
// the resulting json is empty ...

private String generateResultingJsonString(
    char wrappingQuote, Map<String, Object>jsonMap) {
    JSONObject jsonObject = new JSONObject(jsonMap);
    String newJson = jsonObject.toString(LT_COMPRESS);
    if (
        -      newJson.charAt(1) != wrappingQuote
        +      !jsonObject.isEmpty() &&
        +      (newJson.charAt(1) != wrappingQuote)
    ) {
        return replaceUnescaped(
            newJson, newJson.charAt(1), wrappingQuote);
    }
    return newJson;
}
```



## Developers Guidance



## Context

```
// Guidance: fix problem which occurred when
// the resulting json is empty ...
private String generateResultingJsonString(
    char wrappingQuote, Map<String, Object> jsonMap) {
    JSONObject jsonObject = new JSONObject(jsonMap);
    String newJson = jsonObject.toString(LT_COMPRESS);
    if (
        -      newJson.charAt(1) != wrappingQuote
        +      !jsonObject.isEmpty() &&
        +      (newJson.charAt(1) != wrappingQuote)
    ) {
        return replaceUnescaped(
            newJson, newJson.charAt(1), wrappingQuote);
    }
    return newJson;
}
```



## Developers Guidance



## Context

# MODIT : Multi- Modal Learning of Editing Source Code

- ASE 2021

## Findings

Developer Guidance and Context of Edit are very important for Automated Code Editing.

## Contribution

Leveraging multiple information modalities to Automate Code editing, and identification of best way of processing such modalities.

# MODIT : Multi Modal Code Editing (pipeline)

## Input Modalities

Modality 1: Code to be edited

```
newJson.charAt(1) != wrappingQuote
```

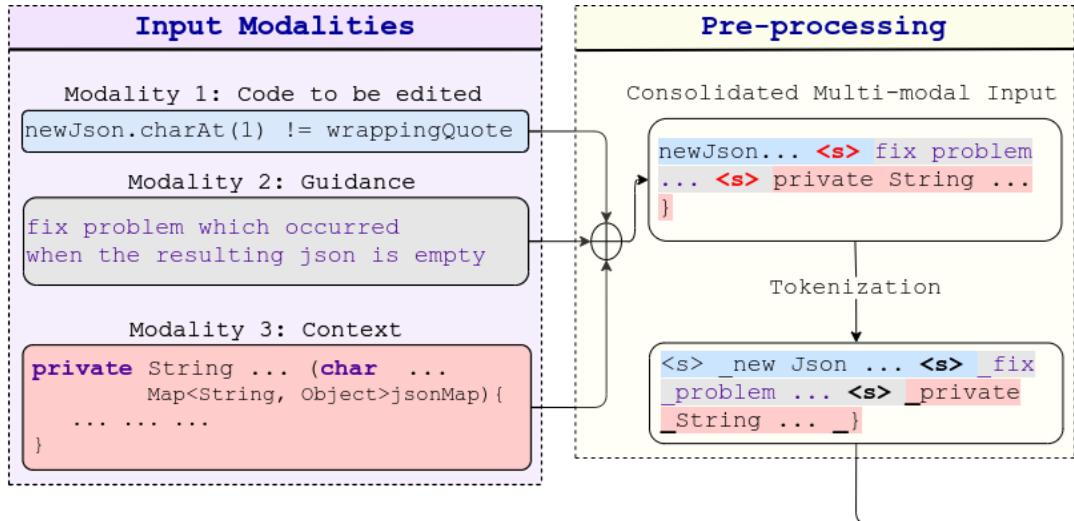
Modality 2: Guidance

```
fix problem which occurred  
when the resulting json is empty
```

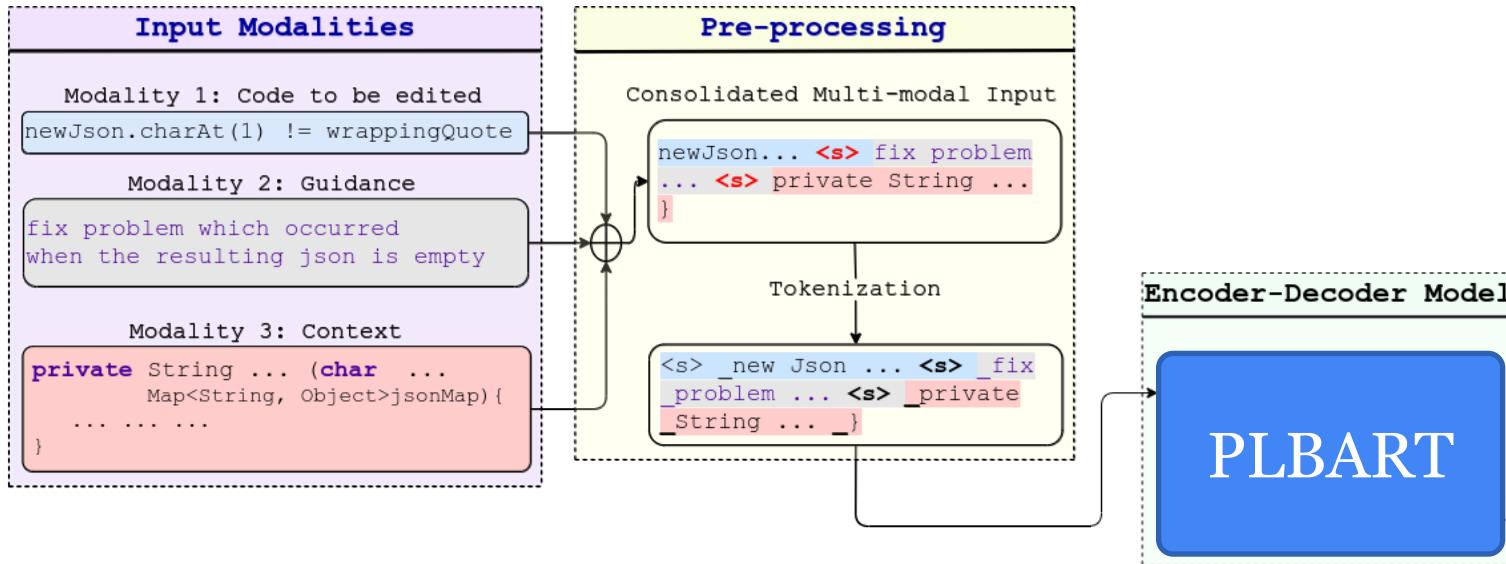
Modality 3: Context

```
private String ... (char ...  
    Map<String, Object>jsonMap)  
{ ... }
```

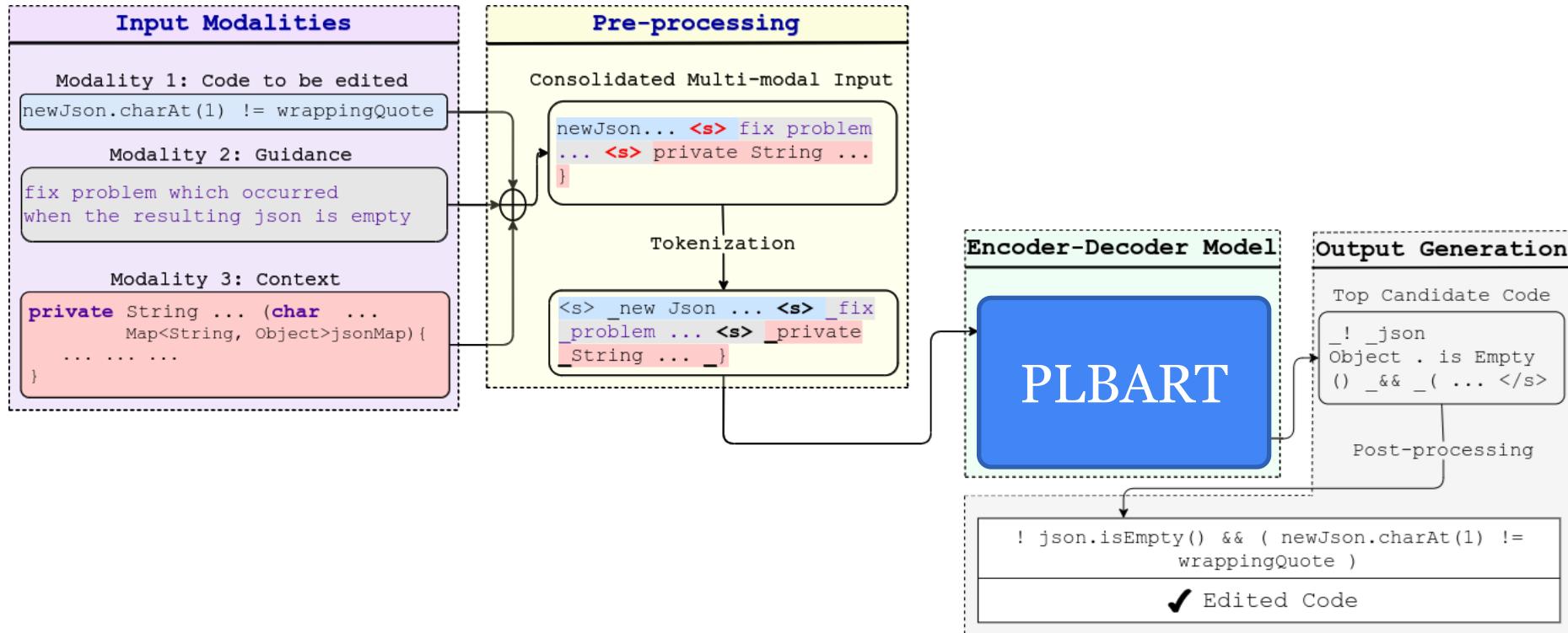
# MODIT : Multi Modal Code Editing (pipeline)



# MODIT : Multi Modal Code Editing (pipeline)



# MODIT : Multi Modal Code Editing (pipeline)



# MODIT Evaluation

# Experiments

Dataset (Tufano et. al. ICSE 2019 [7]) – Concrete Edits

| Dataset | Avg.<br>Tokens | Avg.<br>Change Size* | Avg. tokens<br>in Guidance | # examples |       |      |
|---------|----------------|----------------------|----------------------------|------------|-------|------|
|         |                |                      |                            | Train      | Valid | Test |
| $B2F_s$ | 32.27          | 7.39                 | 11.55                      | 46628      | 5828  | 5831 |
| $B2F_m$ | 74.65          | 8.83                 | 11.48                      | 53324      | 6542  | 6538 |

\* Change size measured as token edit distance.

# MODIT : Result

| Training Type | Model Name               | # of params (M) | Multi-Modal | Accuracy (%) |              |
|---------------|--------------------------|-----------------|-------------|--------------|--------------|
|               |                          |                 |             | $B2F_s$      | $B2F_m$      |
| From Scratch  | LSTM                     | 82.89           | ✓           | 6.14         | 1.04         |
|               | <i>Transformer-base</i>  | 139.22          | ✓           | 11.18        | 6.61         |
|               | <i>Transformer-large</i> | 406.03          | ✓           | 13.40        | 8.63         |
|               | CODIT                    | 105.43          | ✗           | 6.53         | 4.79         |
| Fine-tuned    | CodeBERT                 | 172.50          | ✗           | 24.28        | 16.76        |
|               |                          |                 | ✓           | 26.05        | 17.13        |
|               | GraphCodeBERT            | 172.50          | ✗           | 24.44        | 16.85        |
|               |                          |                 | ✓           | 25.67        | 18.31        |
|               | CodeGPT                  | 124.44          | ✗           | 28.13        | 16.35        |
|               |                          |                 | ✓           | 28.43        | 17.64        |
|               | MODIT (PLBART)           | 139.22          | ✗           | 26.67        | 19.79        |
|               |                          |                 | ✓           | <b>29.99</b> | <b>23.02</b> |

# Interesting Code Generated by PLBART

# Interesting Code Generated by PLBART

PLBART generated

```
public int getCells(){
    Iterator<Character> i =
        cells.keySet().iterator();
    int size = 0;
    for ( ; i.hasNext() ; ) {
        Cell e = at(c);
        ...
    }
    return size;
}
```

# Interesting Code Generated by PLBART

## PLBART generated

```
public int getCells(){
    Iterator<Character> i =
        cells.keySet().iterator();
    int size = 0;
    for ( ; i.hasNext() ; ) {
        Cell e = at(c);
        ...
    }
    return size;
}
```

## Better Code

```
public int getCells(){
    Iterator<Character> i =
        cells.keySet().iterator();
    int size = 0;
    while(i.hasNext()) {
        Cell e = at(c);
        ...
    }
    return size;
}
```

# Interesting Code Generated by PLBART

## PLBART generated

```
public int getCells(){
    Iterator<Character> i =
        cells.keySet().iterator();
    int size = 0;
    for ( ; i.hasNext() ; ) {
        Cell e = at(c);
        ...
    }
    return size;
}
```

## Better Code

```
public int getCells(){
    Iterator<Character> i =
        cells.keySet().iterator();
    int size = 0;
    while(i.hasNext()) {
        Cell e = at(c);
        ...
    }
    return size;
}
```

# Interesting Code Generated by PLBART

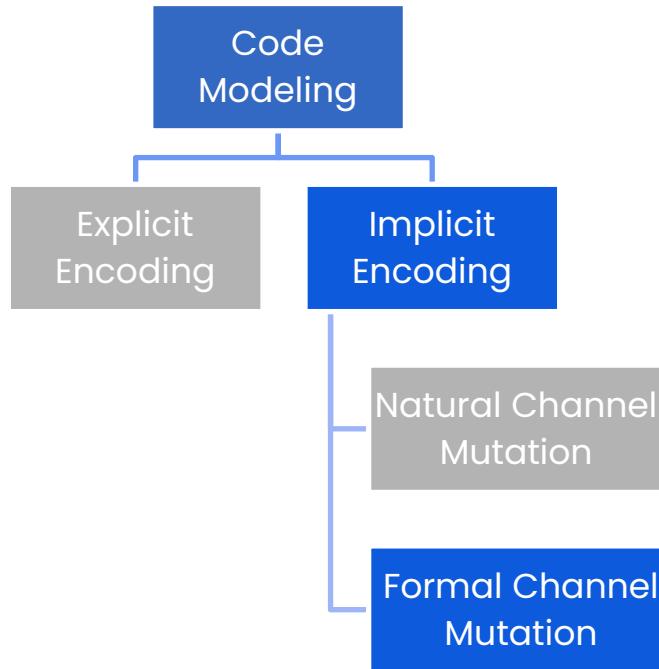
## PLBART generated

```
public int getCells(){
    Iterator<Character> i =
        cells.keySet().iterator();
    int size = 0;
    for ( ; i.hasNext() ; ) {
        Cell e = at(c);
        ...
    }
    return size;
}
```

## Better Code

```
public int getCells(){
    Iterator<Character> i =
        cells.keySet().iterator();
    int size = 0;
    while(i.hasNext()) {
        Cell e = at(c);
        ...
    }
    return size;
}
```

# Implicit Encoding



```
public int getCells() {  
    Iterator<Character> i =  
        cells.keySet().iterator();  
    int size = 0;  
    for ( ; i.hasNext() ; ) {  
        Cell e = at(c);  
        ...  
    }  
    return size;  
}
```



```
public int getCells() {  
    Iterator<Character> i =  
        cells.keySet().iterator();  
    int size = 0;  
    while(i.hasNext()) {  
        Cell e = at(c);  
        ...  
    }  
    return size;  
}
```

# NatGen: Generative Pretraining by “Naturalizing” Source Code

- FSE 2022

## Findings

Forcing a model to edit unnatural code to re-write in a more natural way embeds developers' natural way of writing code into the model.

## Contribution

Designed a large-scale pretrained model which learned natural coding patterns from developers, with demonstrated higher performance in few shot code generation.

# NatGen: Generative pre-training by “Naturalizing” source code

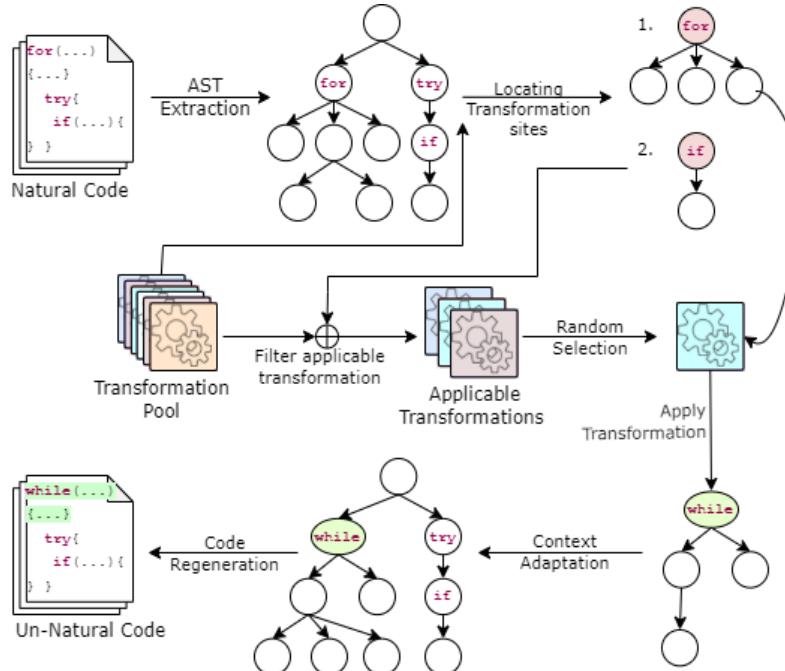
```
public int getCells() {
    Iterator<Character> i =
        cells.keySet().iterator();
    int size = 0;
    for ( ; i.hasNext() ; ) {
        Cell e = at(c);
        ...
    }
    return size;
}
```

```
public int getCells() {
    Iterator<Character> i =
        cells.keySet().iterator();
    int size = 0;
    while(i.hasNext()) {
        Cell e = at(c);
        ...
    }
    return size;
}
```

Write Semantic / Functional Equivalent Code in  
“More Natural” way

# NatGen: Generative pre-training by “Naturalizing” source code

De-Naturalizing Transformation



# NatGen: De-Naturalizing Transformations

# NatGen: De-Naturalizing Transformations

```
public int getCells() {
    Iterator<Character> i =
        cells.keySet().iterator();
    int size = 0;
    while(size > 0) {
        Cell e = at(size);
    }
    while(i.hasNext()) {
        Cell e = at(c);
        ...
    }
    return size;
}
Dead Code Insertion
```

# NatGen: De-Naturalizing Transformations

```
public int getCells() {
    Iterator<Character> i =
        cells.keySet().iterator();
    int size = 0;
    while(size > 0) {
        Cell e = at(size);
    }
    while(i.hasNext()) {
        Cell e = at(c);
        ...
    }
    return size;
}
```

Dead Code Insertion

```
- if (x > 0) {
-     Cell e = at(x);
- }
- else {
-     Cell c = at(x+1);
- }
+ Cell c = (x > 0) ? at(x) : at(x+1);
```

Confusing Statements [17]

# NatGen: De-Naturalizing Transformations

```
public int getCells() {
    Iterator<Character> i =
        cells.keySet().iterator();
    int size = 0;
    while(size > 0) {
        Cell e = at(size);
    }
    while(i.hasNext()) {
        Cell e = at(c);
        ...
    }
    return size;
}
```

Dead Code Insertion

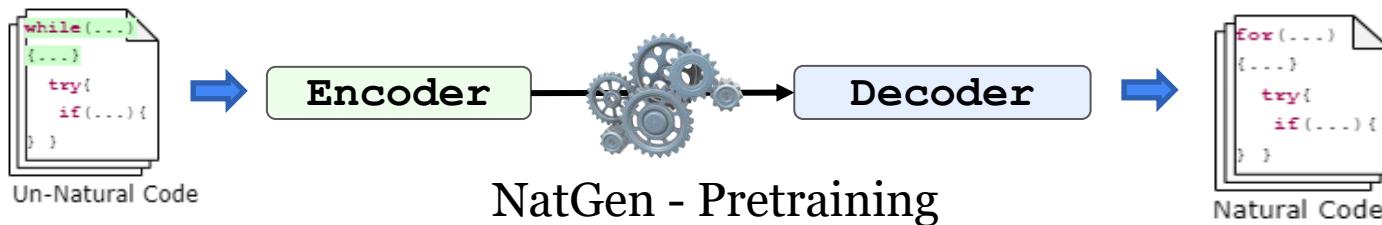
```
- if (x > 0) {
-     Cell e = at(x);
-
- } else {
-     Cell c = at(x+1);
-
+ Cell c = (x > 0) ? at(x) : at(x+1);
```

Confusing Statements [17]

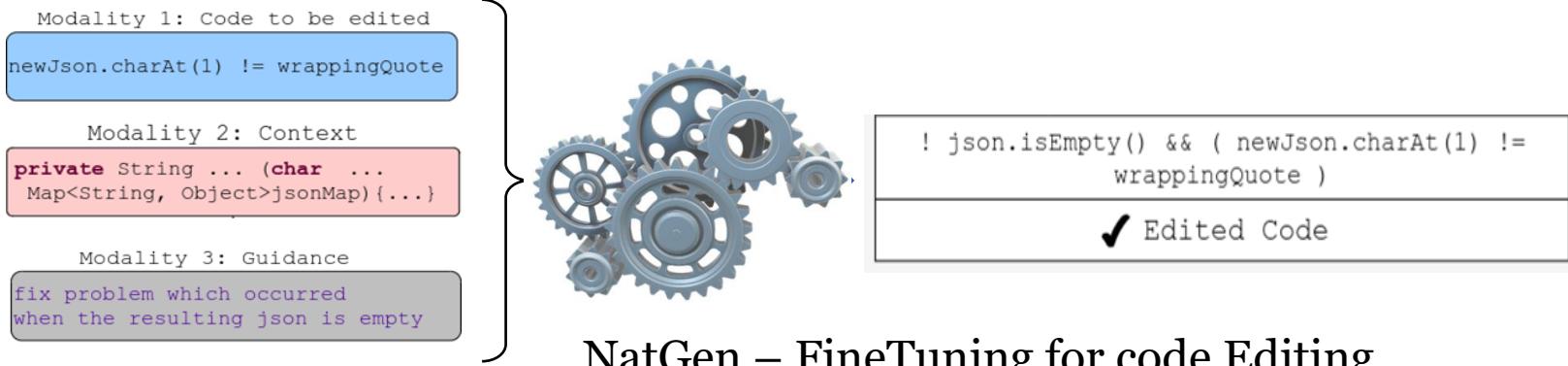
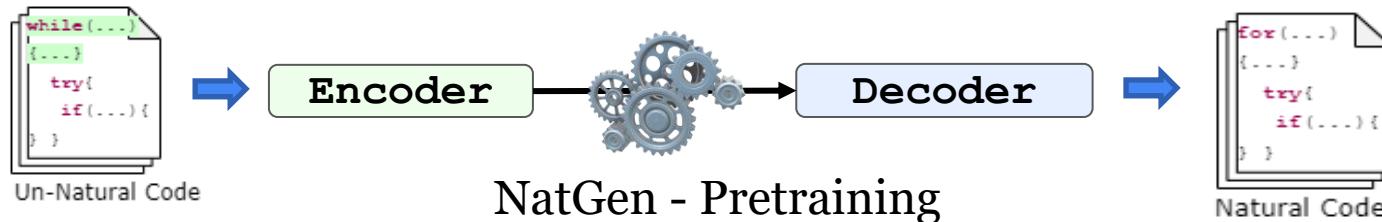
```
- if (x > 0) {
+ if (0 < x) {
    Cell e = at(x);
}
```

Operand Swap

# NatGen: Generative pre-training by “Naturalizing” source code



# NatGen: Generative pre-training by “Naturalizing” source code



NatGen – FineTuning for code Editing

# NatGen Evaluation

# NatGen: Code Generation By Pretrained Models

| Model      | Syntax Match (%) | Dataflow Match (%) | CodeBLEU (%) | Direct Copy (%) | Avg. Edit Distance |
|------------|------------------|--------------------|--------------|-----------------|--------------------|
| CodeT5[18] | 13.83            | 23.67              | 10.87        | 0               | 65                 |
| PLBART     | 73.17            | 75.95              | 74.56        | 7.05            | 3                  |
| NatGen     | 98.16            | 96.85              | 96.82        | 0.01            | 10                 |

# NatGen: Code Generation By Pretrained Models

| Model      | Syntax Match (%) | Dataflow Match (%) | CodeBLEU (%) | Direct Copy (%) | Avg. Edit Distance |
|------------|------------------|--------------------|--------------|-----------------|--------------------|
| CodeT5[18] | 13.83            | 23.67              | 10.87        | 0               | 65                 |
| PLBART     | 73.17            | 75.95              | 74.56        | 7.05            | 3                  |
| NatGen     | <b>98.16</b>     | <b>96.85</b>       | <b>96.82</b> | 0.01            | 10                 |

## 1. Input

```
protected SDV iam(SDV in,...){  
    if(i < i){  
        return new IAM(...);  
    }  
    return new IAM(...);  
}
```

# NatGen: Code Generation By Pretrained Models

| Model      | Syntax Match (%) | Dataflow Match (%) | CodeBLEU (%) | Direct Copy (%) | Avg. Edit Distance |
|------------|------------------|--------------------|--------------|-----------------|--------------------|
| CodeT5[18] | 13.83            | 23.67              | 10.87        | 0               | 65                 |
| PLBART     | 73.17            | 75.95              | 74.56        | 7.05            | 3                  |
| NatGen     | <b>98.16</b>     | <b>96.85</b>       | <b>96.82</b> | 0.01            | 10                 |

1. Input

```
protected SDV iam(SDV in,...){    SDV iam(SDV in, ...){  
    if(i < i){                      if(i < i){  
        return new IAM(...);          return new IAM(...);  
    }                                }  
    return new IAM(...);            return new IAM(...);  
}
```

2. PLBART output

# NatGen: Code Generation By Pretrained Models

| Model      | Syntax Match (%) | Dataflow Match (%) | CodeBLEU (%) | Direct Copy (%) | Avg. Edit Distance |
|------------|------------------|--------------------|--------------|-----------------|--------------------|
| CodeT5[18] | 13.83            | 23.67              | 10.87        | 0               | 65                 |
| PLBART     | 73.17            | 75.95              | 74.56        | 7.05            | 3                  |
| NatGen     | <b>98.16</b>     | <b>96.85</b>       | <b>96.82</b> | 0.01            | 10                 |

1. Input

```
protected SDV iam(SDV in,...){  
    if(i < i){  
        return new IAM(...);  
    }  
    return new IAM(...);  
}
```

2. PLBART output

```
SDV iam(SDV in, ...){  
    if(i < i){  
        return new IAM(...);  
    }  
    return new IAM(...);  
}
```

3. NATGEN output

```
protected SDV iam(SDV in,...){  
    return new IAM(...);  
}
```

# NatGen: Code Generation By Pretrained Models

| Model      | Syntax Match (%) | Dataflow Match (%) | CodeBLEU (%) | Direct Copy (%) | Avg. Edit Distance |
|------------|------------------|--------------------|--------------|-----------------|--------------------|
| CodeT5[18] | 13.83            | 23.67              | 10.87        | 0               | 65                 |
| PLBART     | 73.17            | 75.95              | 74.56        | 7.05            | 3                  |
| NatGen     | <b>98.16</b>     | <b>96.85</b>       | <b>96.82</b> | 0.01            | 10                 |

1. Input

```
protected SDV iam(SDV in,...){  
    if(i < i){  
        return new IAM(...);  
    }  
    return new IAM(...);  
}
```

2. PLBART output

```
SDV iam(SDV in, ...){  
    if(i < i){  
        return new IAM(...);  
    }  
    return new IAM(...);  
}
```

3. NATGEN output

```
protected SDV iam(SDV in,...){  
    if (in) {  
        return  
    }  
}
```

4. CodeT5 output

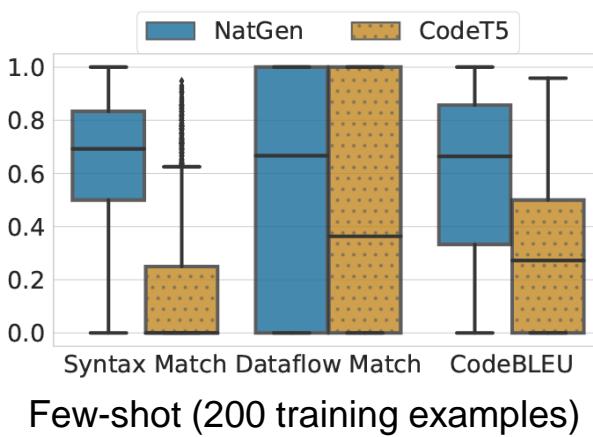
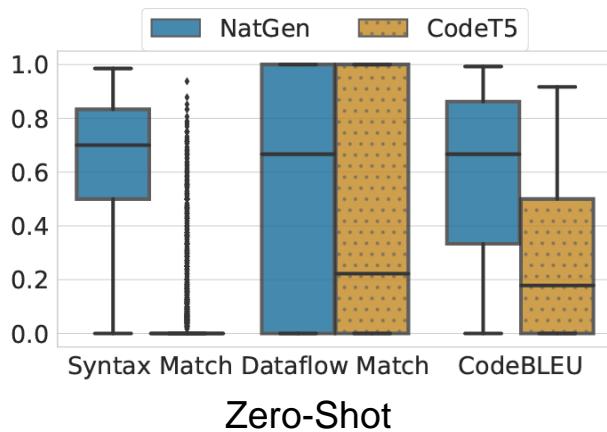
# NatGen: Automated Code Editing Result

| Approach | BugFix <sub>small</sub> |              | BugFix <sub>medium</sub> |              |
|----------|-------------------------|--------------|--------------------------|--------------|
|          | Unimodal                | Multimodal   | Unimodal                 | Multimodal   |
| MODIT    | 20.35                   | 21.57        | 8.35                     | 13.18        |
| CodeT5   | 21.79                   | 22.97        | 12.59                    | <b>14.94</b> |
| NATGEN   | <b>22.26</b>            | <b>23.43</b> | <b>13.32</b>             | 14.93        |

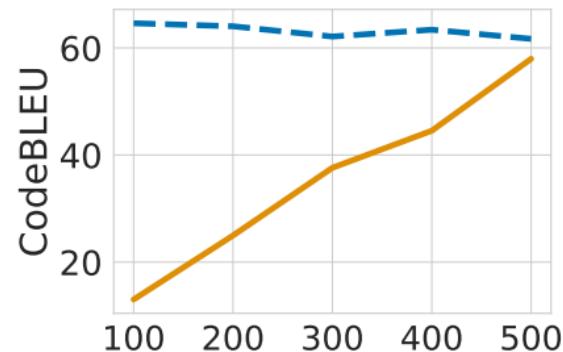
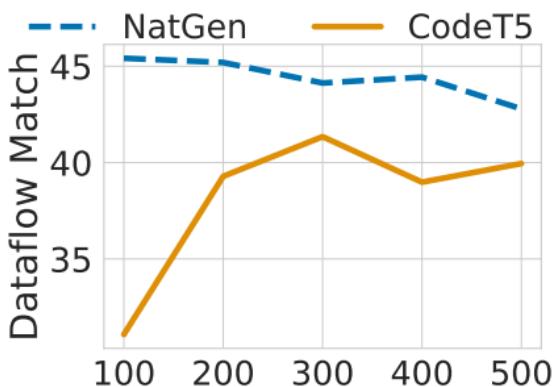
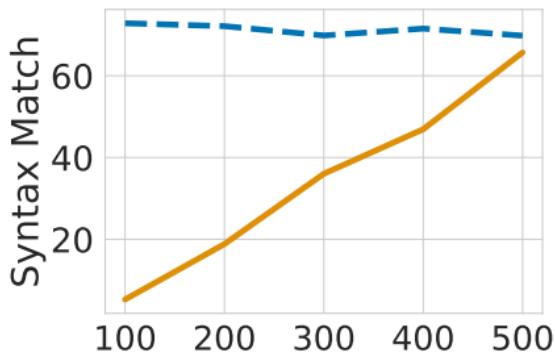
BugFix dataset proposed by Tufano et al. (ICSE 2019) [7]

Exact Match accuracy (%) at Top 1 recommendation.

# NatGen: Automated Code Editing Result

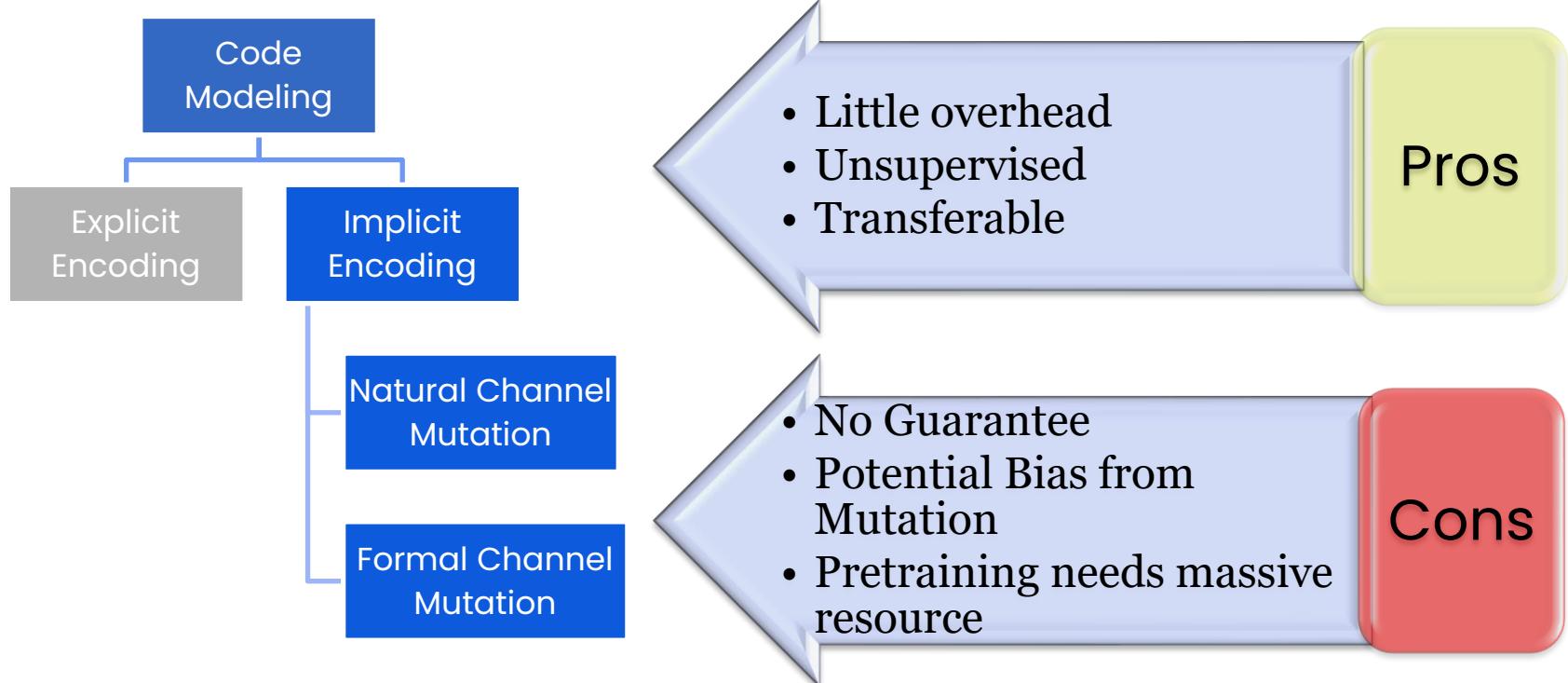


# NatGen: Automated Code Editing Result



Number of Training Examples →

# Explicit Encoding – Take Aways



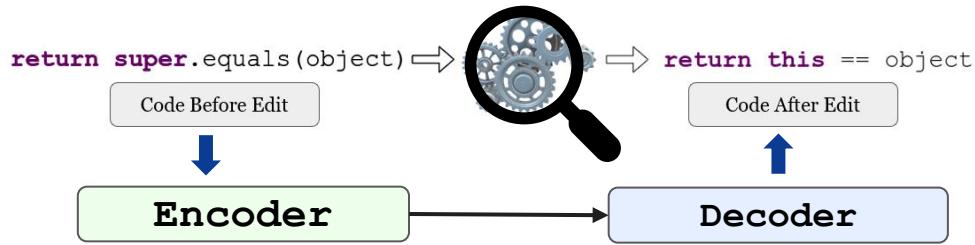
# Dissertation Summary

`return super.equals(object) ⇒`  `⇒ return this == object`

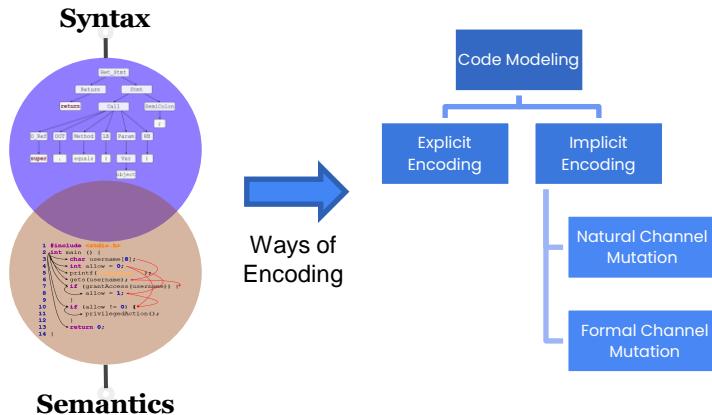
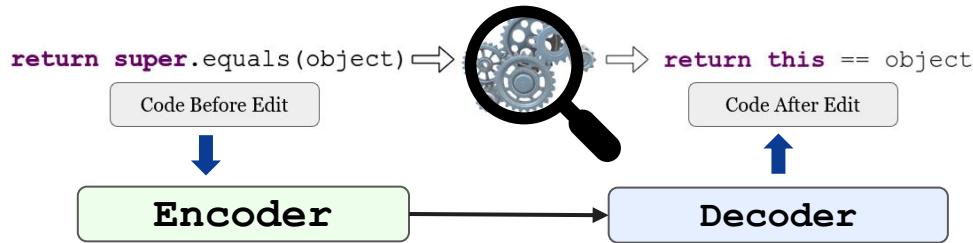
Code Before Edit

Code After Edit

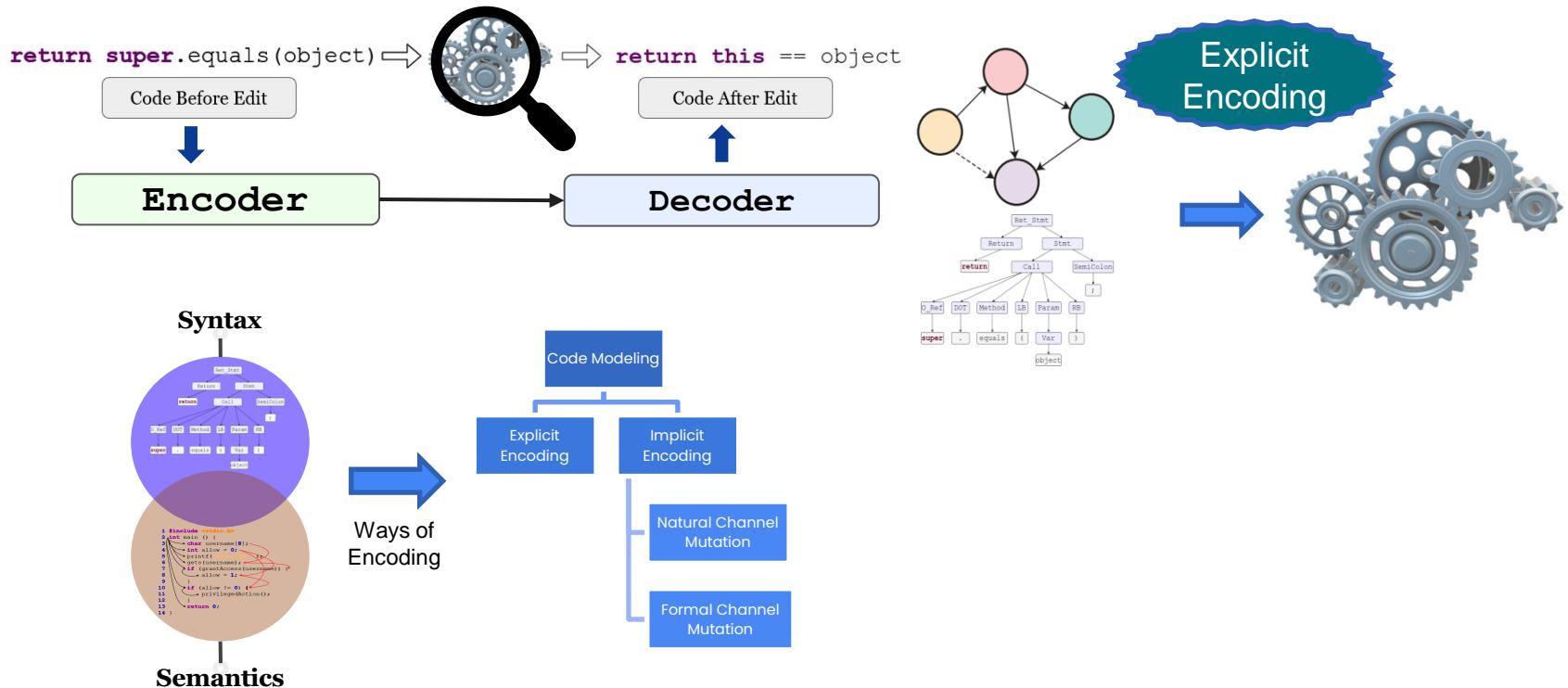
# Dissertation Summary



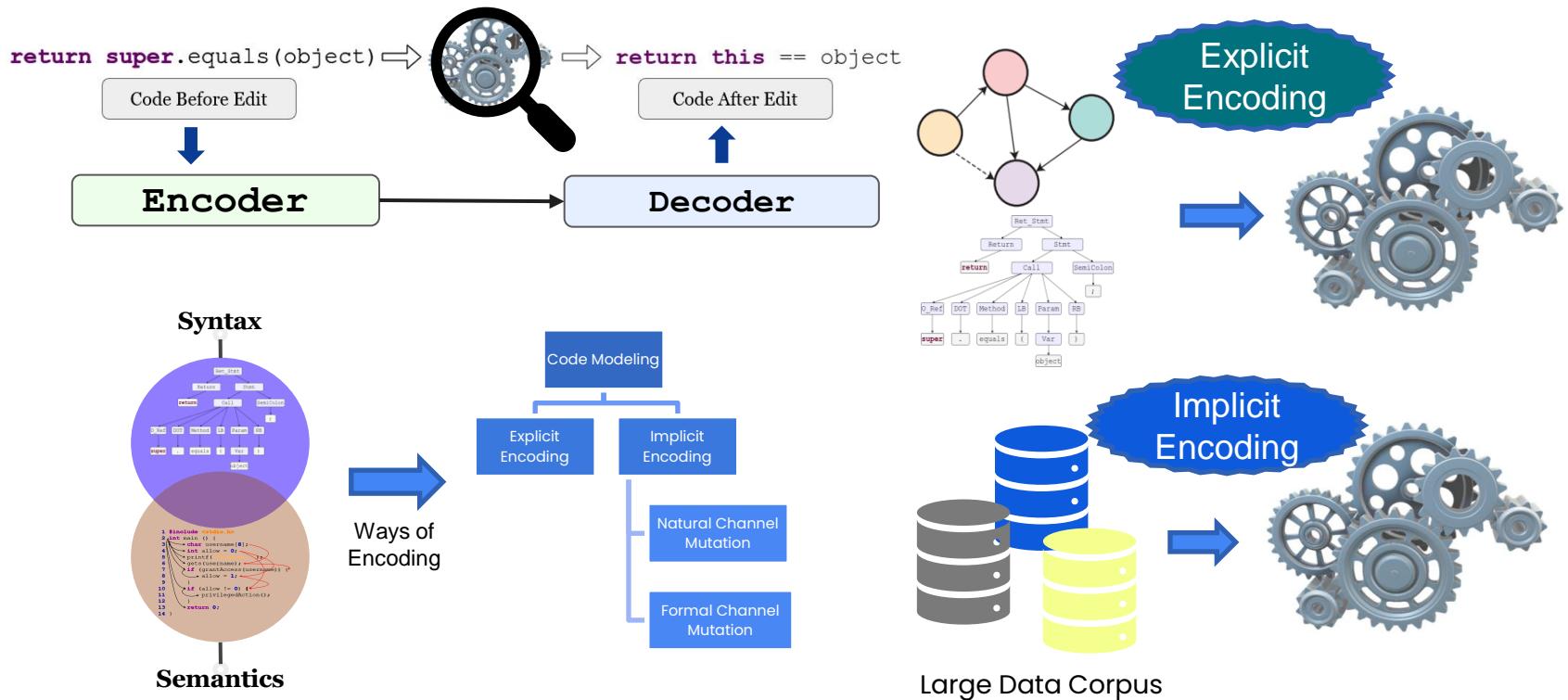
# Dissertation Summary



# Dissertation Summary



# Dissertation Summary

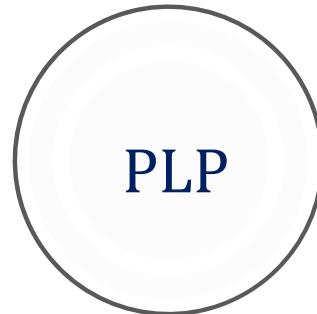


# List of Publications

1. **NatGen: Generative pre-training by" Naturalizing" source code** - S Chakraborty, T Ahmed, Y Ding, P Devanbu, B Ray. The ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2022).
2. Towards Learning (Dis)-Similarity of Source Code from Program Contrasts - Y Ding, L Buratti, S Pujar, A Morari, B Ray, S Chakraborty Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL'22).
3. **On Multi-Modal Learning of Editing Source Code** - S Chakraborty, B Ray 2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE'21).
4. Retrieval Augmented Code Generation and Summarization - MR Parvez, WU Ahmad, S Chakraborty, B Ray, KW Chang Findings of the Association for Computational Linguistics 2021 (ENMLP).
5. **Deep learning-based vulnerability detection: Are we there yet?** - S Chakraborty, R Krishna, Y Ding, B Ray IEEE Transactions on Software Engineering (TSE'21).
6. **Unified Pre-training for Program Understanding and Generation** - WU Ahmad, S Chakraborty, B Ray, KW Chang Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL'21).
7. **CODIT: Code editing with tree-based neural models** - S Chakraborty, Y Ding, M Allamanis, B Ray IEEE Transactions on Software Engineering (TSE'20).
8. A transformer-based Approach for Source Code Summarization - W Ahmad, S Chakraborty, B Ray, KW Chang Association for Computational Linguistics (ACL'20).
9. Toward optimal selection of information retrieval models for software engineering tasks - MM Rahman, S Chakraborty, G Kaiser, B Ray 2019 19th International Working Conference on Source Code Analysis and Maintenance (SCAM'19).
10. Building language models for text with named entities - MR Parvez, S Chakraborty, B Ray, KW Chang 56th Annual Meeting of the Association for Computational Linguistics (ACL'18).
11. Which similarity metric to use for software documents? A study on information retrieval-based software engineering tasks - MM Rahman, S Chakraborty, B Ray Proceedings of the 40th International Conference on Software Engineering (ICSE'18).

# My Research Applications

## Programming Language Processing



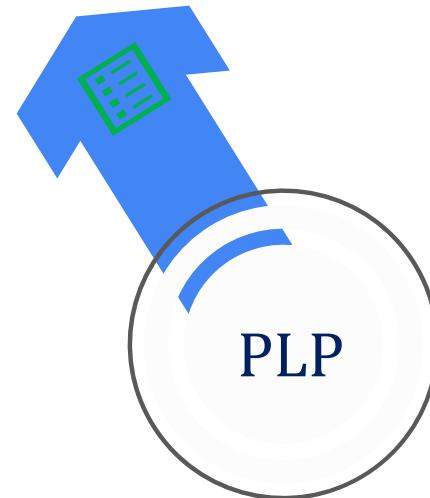
# My Research Applications

## Programming Language Processing



### Code Summarization

NeuralCodeSum (ACL'20), PLBART (NAACL'21)



# My Research Applications

## Programming Language Processing



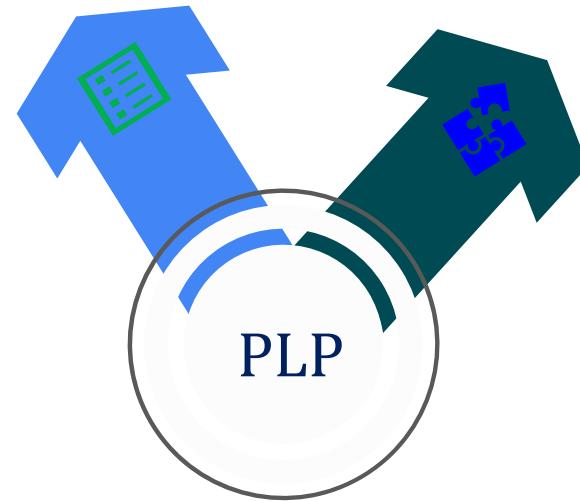
### Code Summarization

NeuralCodeSum (ACL'20), PLBART (NAACL'21)



### Vulnerability Detection

ReVeal(TSE'21), DISCO(ACL'22)



# My Research Applications

## Programming Language Processing



### Code Summarization

NeuralCodeSum (ACL'20), PLBART (NAACL'21)



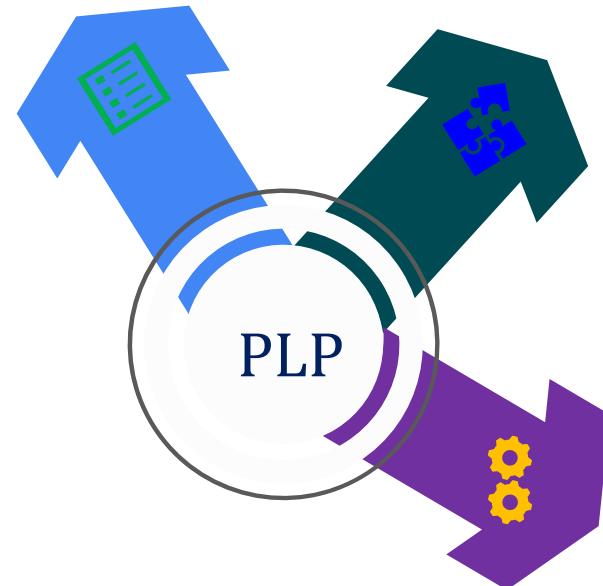
### Vulnerability Detection

ReVeal(TSE'21), DISCO(ACL'22)



### Code Editing

CODIT (TSE'20), MODIT(ASE'21), DiffBERT  
(Facebook)



# My Research Applications

## Programming Language Processing



### Code Summarization

NeuralCodeSum (ACL'20), PLBART (NAACL'21)



### Vulnerability Detection

ReVeal(TSE'21), DISCO(ACL'22)



### Code Editing

CODIT (TSE'20), MODIT(ASE'21), DiffBERT  
(Facebook)



### Code Generation

PLBART (NAACL'21), DataTypeLM ForCode (ACL'18).



# My Research Applications

## Programming Language Processing

 **Code Summarization**  
NeuralCodeSum (ACL'20), PLBART (NAACL'21)

 **Vulnerability Detection**  
ReVeal(TSE'21), DISCO(ACL'22)

 **Code Editing**  
CODIT (TSE'20), MODIT(ASE'21), DiffBERT  
(Facebook)

 **Code Generation**  
PLBART (NAACL'21), DataTypeLM ForCode (ACL'18).

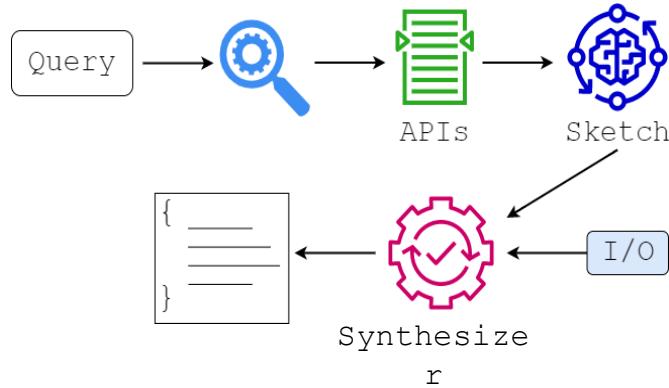
 **Code Search and Synthesis**  
RedCoder (EMNLP'21), CodePanda (W.I.P)



# Future Plan (Short Term Goal)

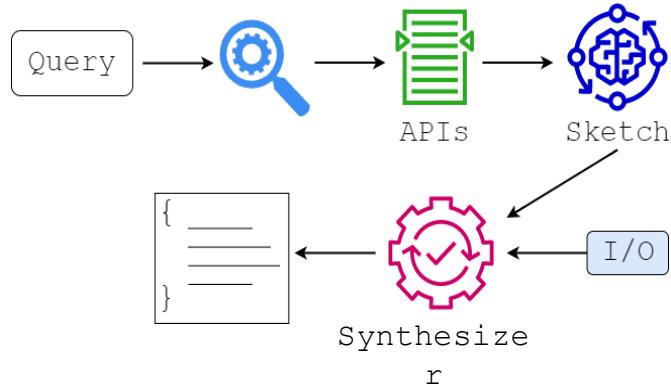
# Future Plan (Short Term Goal)

- API driven Program Synthesis

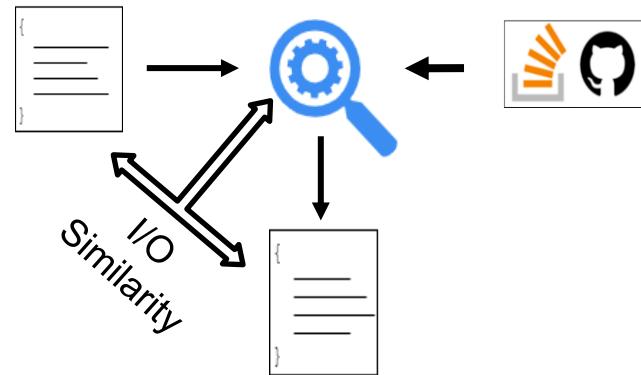


# Future Plan (Short Term Goal)

- API driven Program Synthesis



- Improving Semantic Code Search with RL



# Future Plan (Short Term Goal)

- Learning Code Syntax and Semantics with Reinforcement Learning (RL)

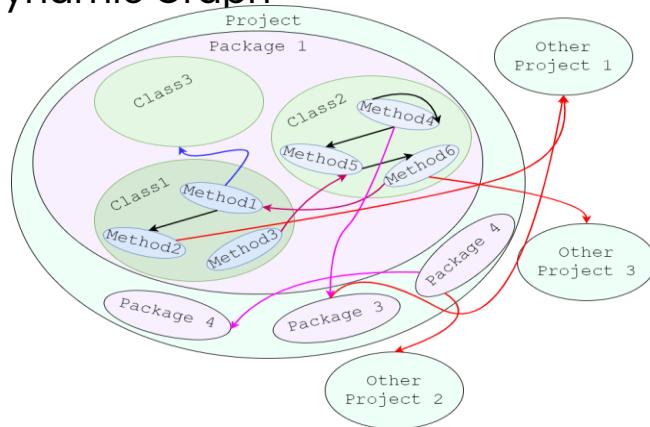
```
void foo(Scanner sc) {  
    String s = "";  
    while(sc.hasNext().get()) {  
        s += sc.nextInt();  
    }  
    return s;  
}
```

# Future Plan (Short Term Goal)

- Learning Code Syntax and Semantics with Reinforcement Learning (RL)

```
void foo(Scanner sc) {  
    String s = "";  
    while(sc.hasNext().get()) {  
        s += sc.nextInt();  
    }  
    return s;  
}
```

- Representing Code Context as Dynamic Graph

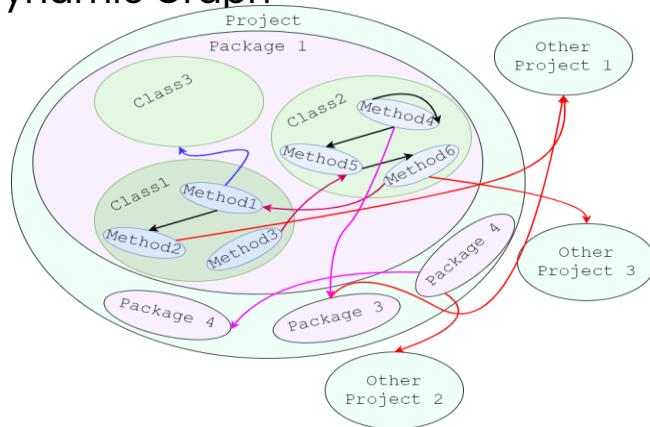


# Future Plan (Short Term Goal)

- Learning Code Syntax and Semantics with Reinforcement Learning (RL)

```
void foo(Scanner sc) {  
    String s = "";  
    while(sc.hasNext().get()) {  
        s += sc.nextInt();  
    }  
    return s;  
}
```

- Representing Code Context as Dynamic Graph



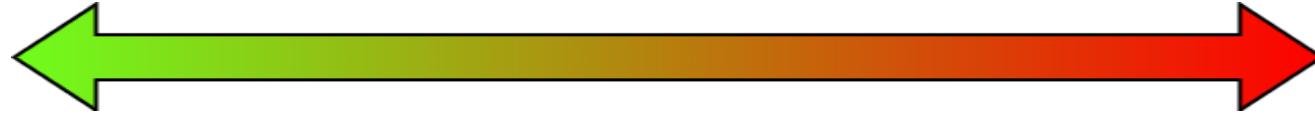
- Other Information Modalities

1. Other Software Metadata – Comments, Commits, Code Evolution metadata.
2. Analysis of Program Binaries.
3. Dynamic Analysis of Program – based on execution behavior.

# Future Plan (Long Term Goal)

# Future Plan (Long Term Goal)

## ➤ Code Generation



Formal Analysis

Probabilistic models

- ✓ **Guarantee** for the Analysis
- ✗ Noise Intolerant

- ✓ **Scalable and Transferrable**
- ✗ No theoretical Guarantee

# Future Plan (Long Term Goal)

## ➤ Code Generation



Formal Analysis

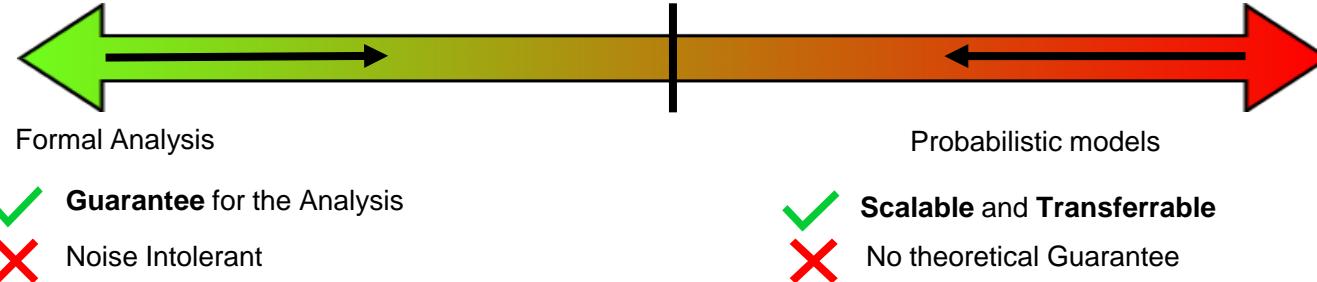
Probabilistic models

- ✓ **Guarantee** for the Analysis
- ✗ Noise Intolerant

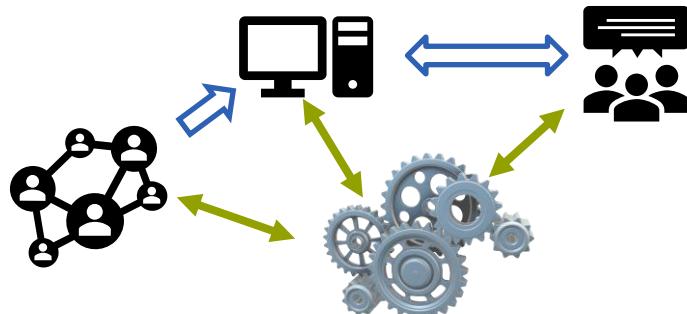
- ✓ **Scalable and Transferrable**
- ✗ No theoretical Guarantee

# Future Plan (Long Term Goal)

## ➤ Code Generation



## ➤ Developer Feedback Oriented Automation





Wasi Ahmad  
UCLA / Amazon



Miltos Allanmanis  
MSR



Baishakhi Ray  
Adviser



Kai-Wei Chang  
UCLA



Gail Kaiser  
Columbia University



Prem Devanbu  
UC Davis



Masudur Rahman  
Uva/Purdue



Rahul Krishna  
Columbia / IBM



Toufiq Parag  
UC Davis



Rizwan Parvez  
UCLA

# References

- [1] Meng, Na, Miryung Kim, and Kathryn S. McKinley. "Systematic editing: generating program transformations from an example." ACM SIGPLAN Notices 46.6 (2011): 329-342.
- [2] Meng, Na, Miryung Kim, and Kathryn S. McKinley. "LASE: locating and applying systematic edits by learning from examples." 2013 35th International Conference on Software Engineering (ICSE). IEEE, 2013.
- [3] Ray, Baishakhi, et al. "The uniqueness of changes: Characteristics and applications." 2015 IEEE/ACM 12th Working Conference on Mining Software Repositories. IEEE, 2015.
- [4] Rolim, Reudismam, et al. "Learning syntactic program transformations from examples." 2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE). IEEE, 2017.
- [5] Dinella, Elizabeth, et al. "Hoppity: Learning graph transformations to detect and fix bugs in programs." International Conference on Learning Representations (ICLR). 2020.
- [6] Tufano, Michele, et al. "An empirical investigation into learning bug-fixing patches in the wild via neural machine translation." Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering. 2018.
- [7] Tufano, Michele, et al. "On learning meaningful code changes via neural machine translation." 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE). IEEE, 2019.
- [8] Tufano, Michele, et al. "An empirical study on learning bug-fixing patches in the wild via neural machine translation." ACM Transactions on Software Engineering and Methodology (TOSEM) 28.4 (2019): 1-29.
- [9] Chen, Zimin, et al. "Sequencer: Sequence-to-sequence learning for end-to-end program repair." IEEE Transactions on Software Engineering 47.9 (2019): 1943-1959.
- [10] Tufano, Rosalia, et al. "Towards automating code review activities." 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE). IEEE, 2021.
- [11] Feng, Zhangyin, et al. "CodeBERT: A Pre-Trained Model for Programming and Natural Languages." Findings of the Association for Computational Linguistics: EMNLP 2020. 2020.

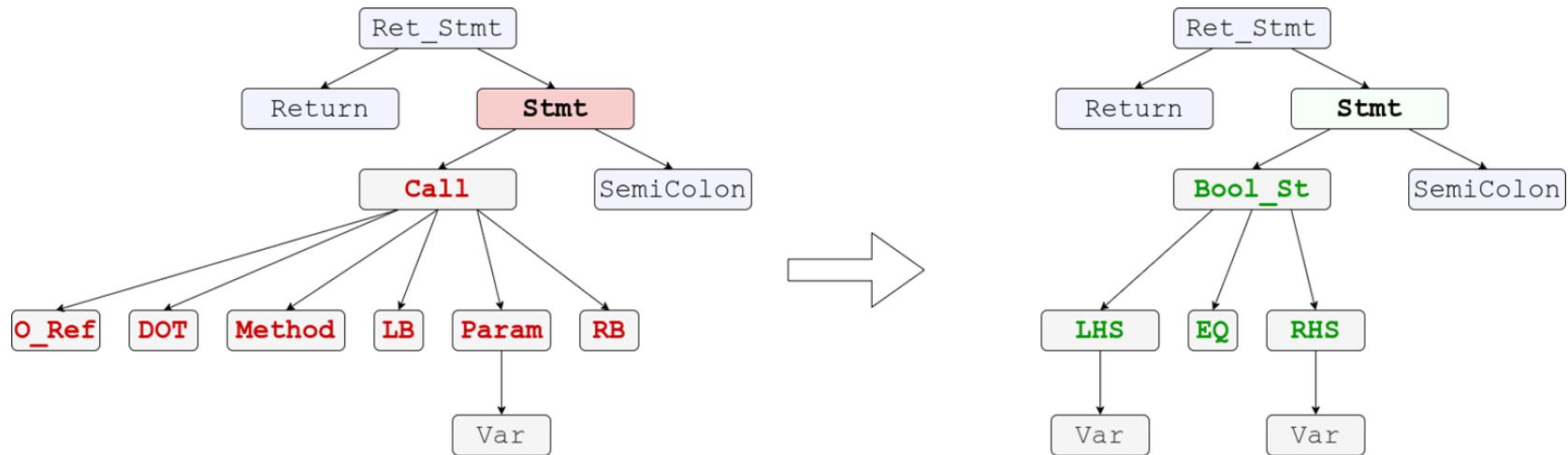
# References

- [12] Lutellier, Thibaud, et al. "Coconut: combining context-aware neural translation models using ensemble for program repair." Proceedings of the 29th ACM SIGSOFT international symposium on software testing and analysis. 2020.
- [13] Allamanis, Miltiadis, Marc Brockschmidt, and Mahmoud Khademi. "Learning to Represent Programs with Graphs." International Conference on Learning Representations. 2018.
- [14] Yin, Pengcheng, et al. "Learning to Represent Edits." International Conference on Learning Representations. 2018.
- [15] Zhou, Yaqin, et al. "Devign: Effective vulnerability identification by learning comprehensive program semantics via graph neural networks." Advances in neural information processing systems 32 (2019).
- [16] Guo, Daya, et al. "GraphCodeBERT: Pre-training Code Representations with Data Flow." International Conference on Learning Representations. 2020.
- [17] Chen, Mark, et al. "Evaluating Large Language Models Trained on Code." (2021).
- [18] Wang, Yue, et al. "CodeT5: Identifier-aware Unified Pre-trained Encoder-Decoder Models for Code Understanding and Generation." Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing. 2021.
- [19] Casalnuovo, Casey, et al. "A theory of dual channel constraints." 2020 IEEE/ACM 42nd International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER). IEEE, 2020.
- [20] Karampatsis, Rafael-Michael, et al. "Big code!= big vocabulary: Open-vocabulary models for source code." 2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE). IEEE, 2020.
- [21] Gopstein, Dan, et al. "Thinking aloud about confusing code: A qualitative investigation of program comprehension and atoms of confusion." Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 2020.

*Thanks!*

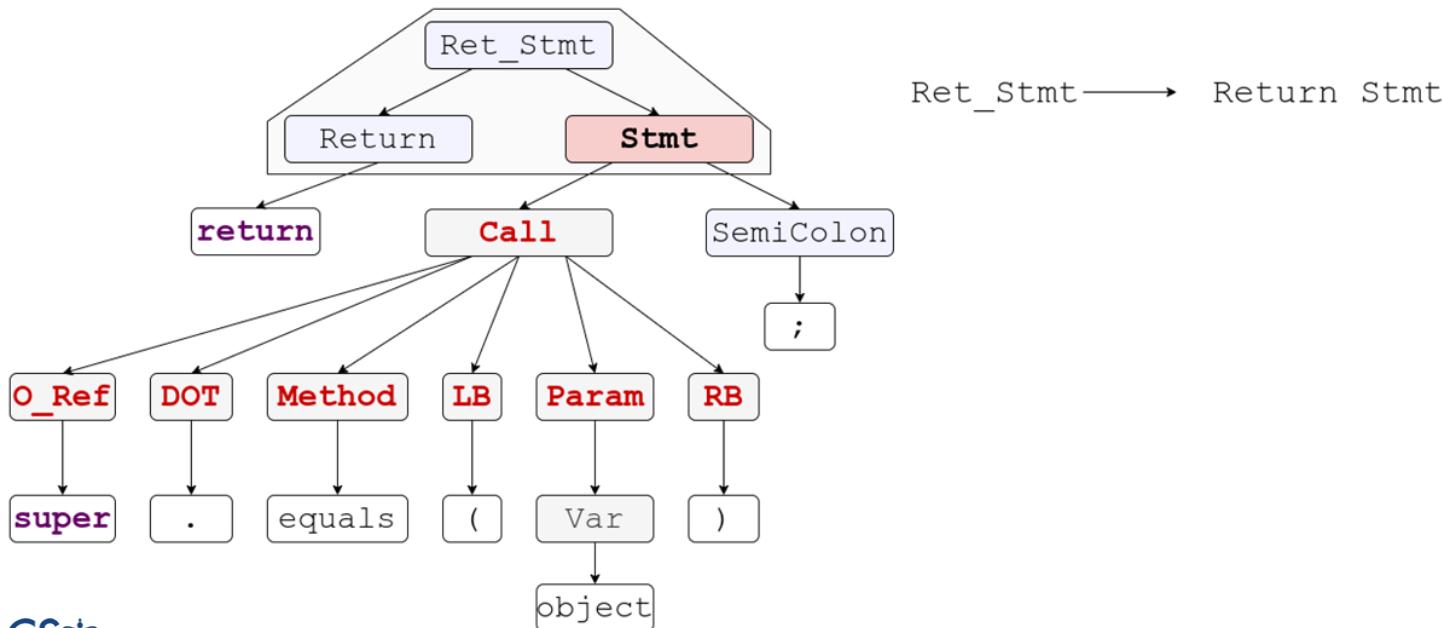
# Backup Slides

# CODIT Step 1 : Tree Translation



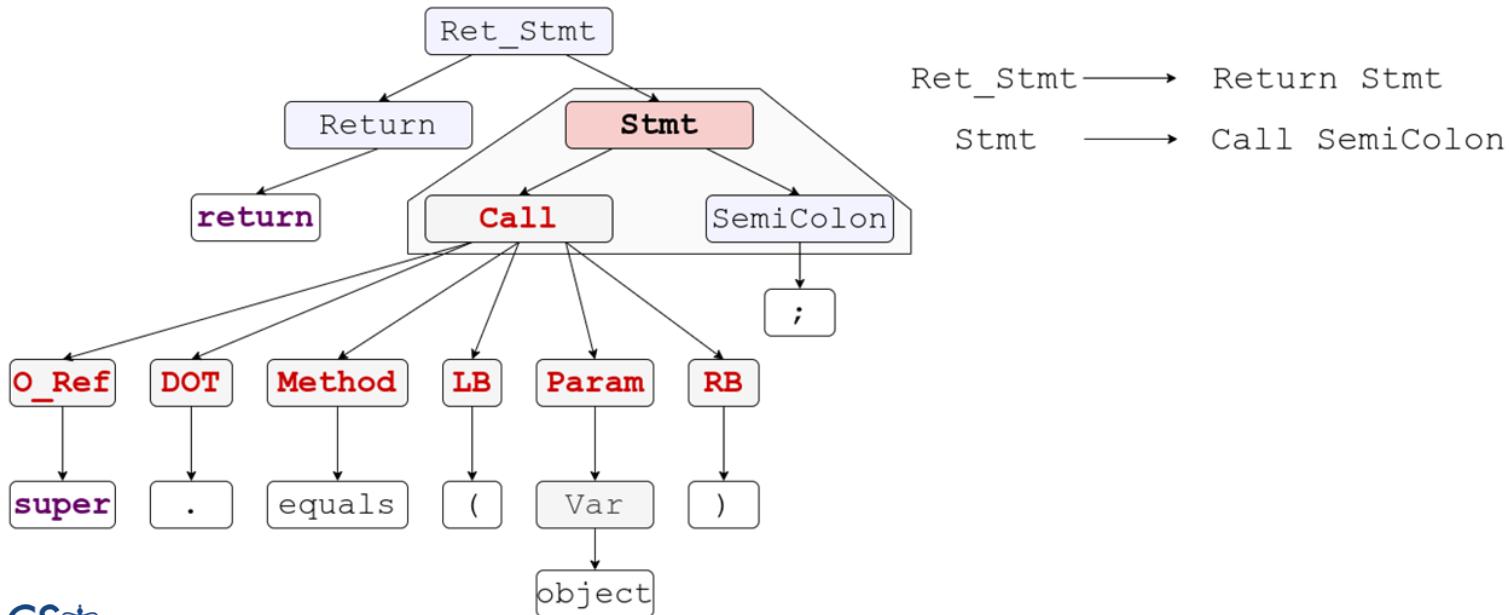
# CODIT Step 1 : Tree Translation

```
return super.equals(object);
```



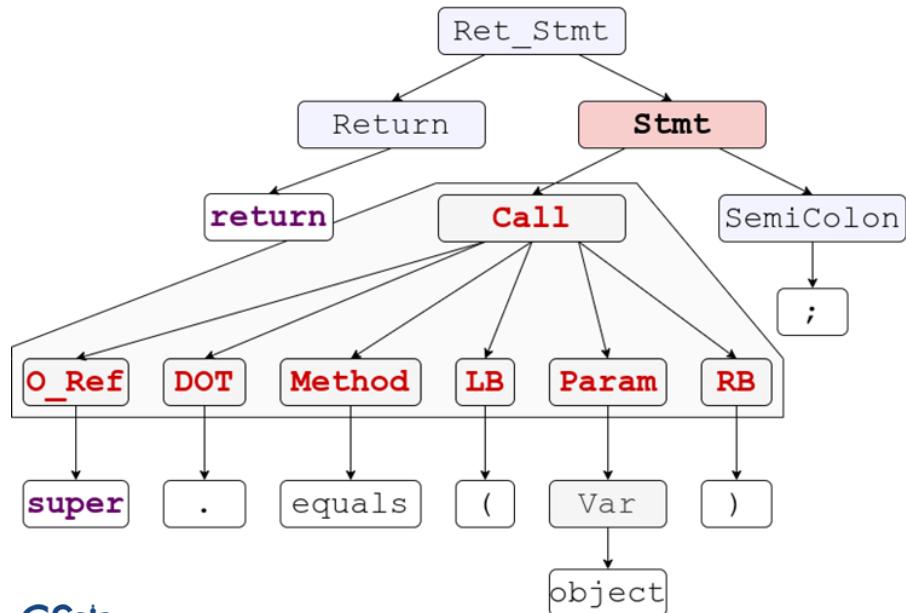
# CODIT Step 1 : Tree Translation

```
return super.equals(object);
```



# CODIT Step 1 : Tree Translation

```
return super.equals(object);
```



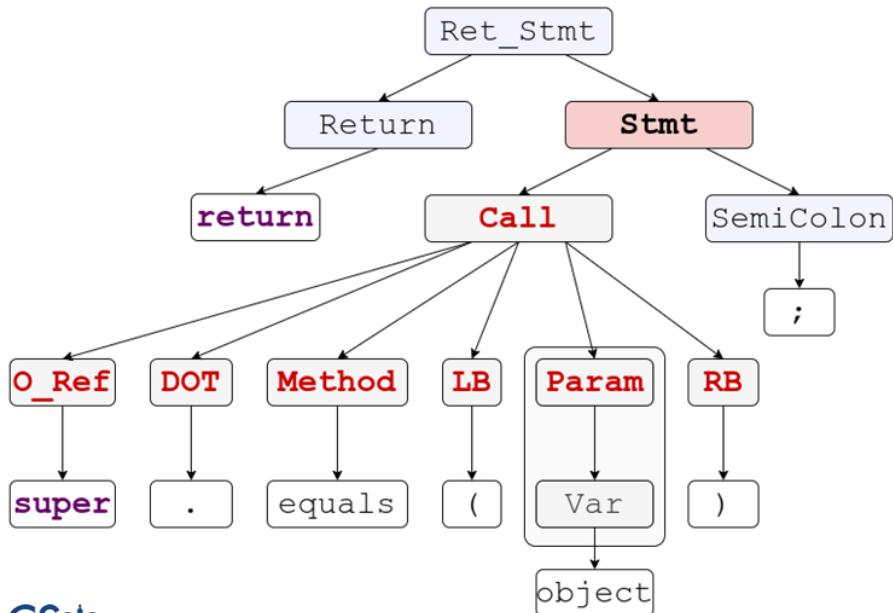
Ret\_Stmt → Return Stmt

Stmt → Call SemiColon

Call → O\_Ref DOT Method LB Param RB

# CODIT Step 1 : Tree Translation

```
return super.equals(object);
```



Ret\_Stmt → Return Stmt

Stmt → Call SemiColon

Call → O\_Ref DOT Method LB Param RB

Param → Var

# CODIT Step 1 : Tree Translation

Rules sequence of  
Syntax Tree before edit

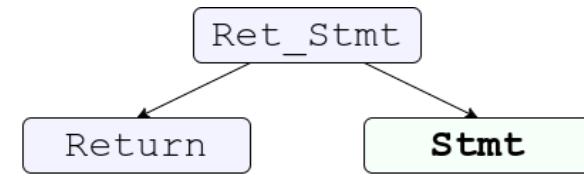
```
Ret_Stmt → Return Stmt  
Stmt → Call Semicolon  
Call → O_Ref DOT Method  
      LB Param RB  
Param → Var
```



Rules sequence of  
Syntax Tree after edit

```
Ret_Stmt → Return Stmt  
Stmt → Bool_St Semicolon  
Bool_St → LHS EQ RHS  
LHS → VAR  
RHS → VAR
```

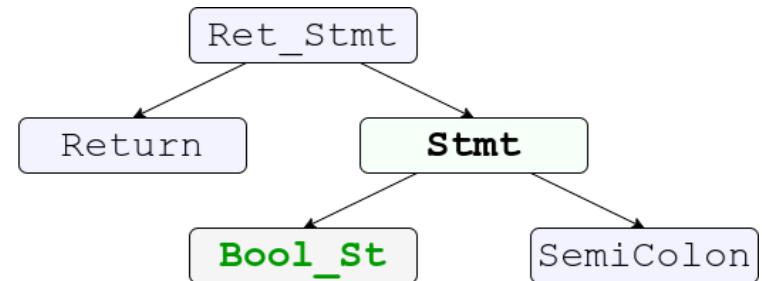
# CODIT Step 1 : Tree Translation



Ret\_Stmt → Return Stmt

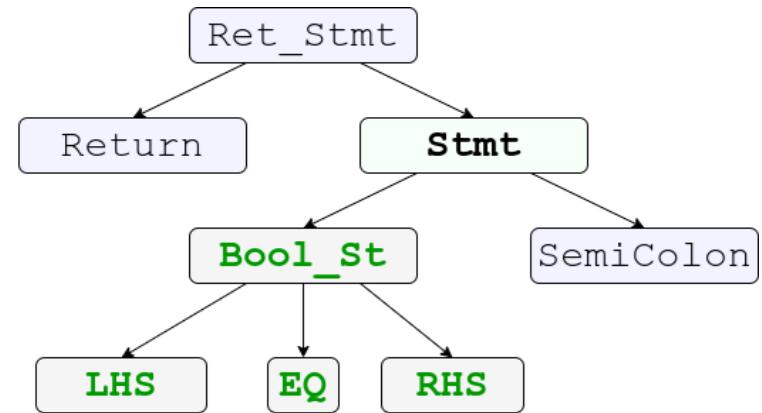
# CODIT Step 1 : Tree Translation

Ret\_Stmt → Return Stmt  
Stmt → Bool\_St Semicolon



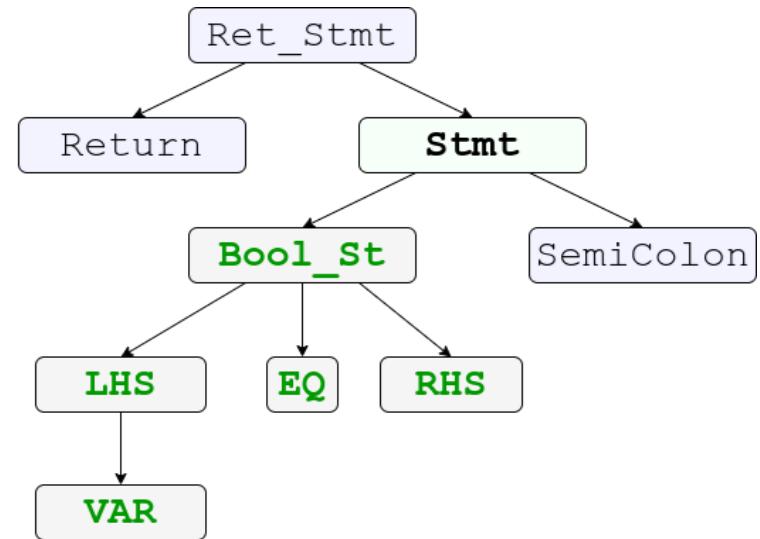
# CODIT Step 1 : Tree Translation

Ret\_Stmt → Return Stmt  
Stmt → Bool\_St Semicolon  
Bool\_St → LHS EQ RHS



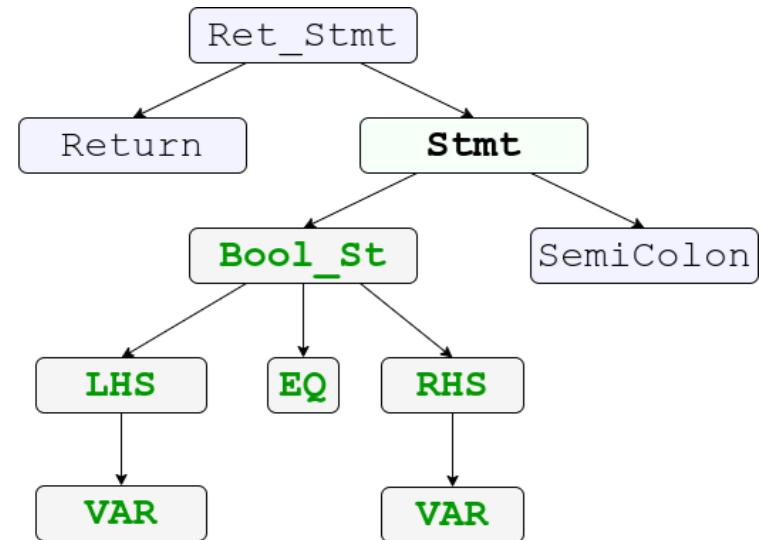
# CODIT Step 1 : Tree Translation

Ret\_Stmt → Return Stmt  
Stmt → Bool\_St Semicolon  
Bool\_St → LHS EQ RHS  
LHS → VAR



# CODIT Step 1 : Tree Translation

Ret\_Stmt → Return Stmt  
Stmt → Bool\_St Semicolon  
Bool\_St → LHS EQ RHS  
LHS → VAR  
RHS → VAR



# ReVeal: Program Understanding through Explicit Program Encoding

TSE - 2021

## Findings

Graph-based models are better equipped to understand semantic relationships between code components

## Contribution

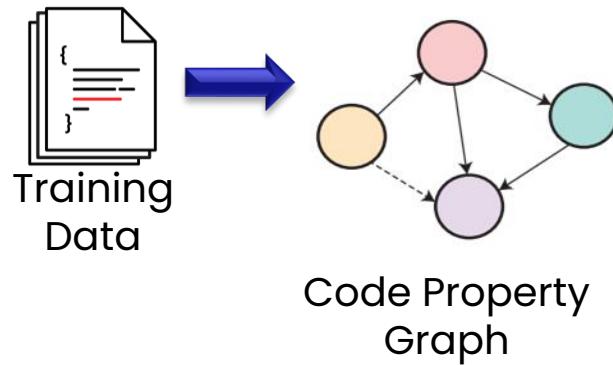
Designed Code Understanding Framework using Graph-Based Models.

# ReVeal : Explicit Encoding for Program Understanding

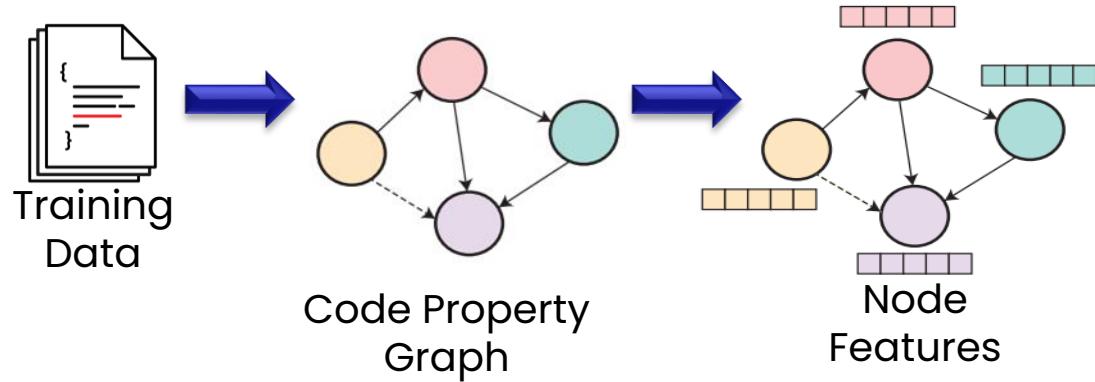


Training  
Data

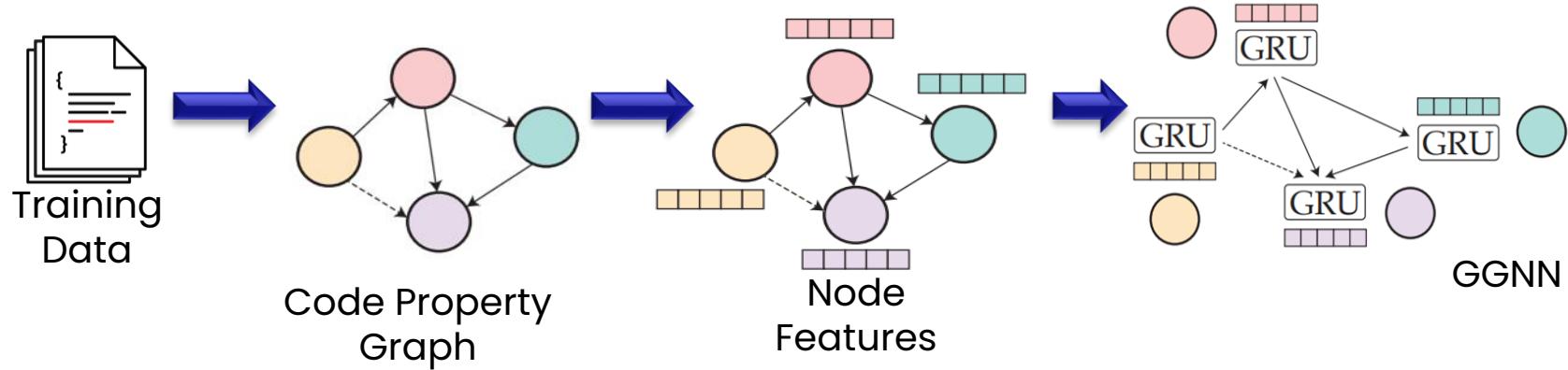
# ReVeal : Explicit Encoding for Program Understanding



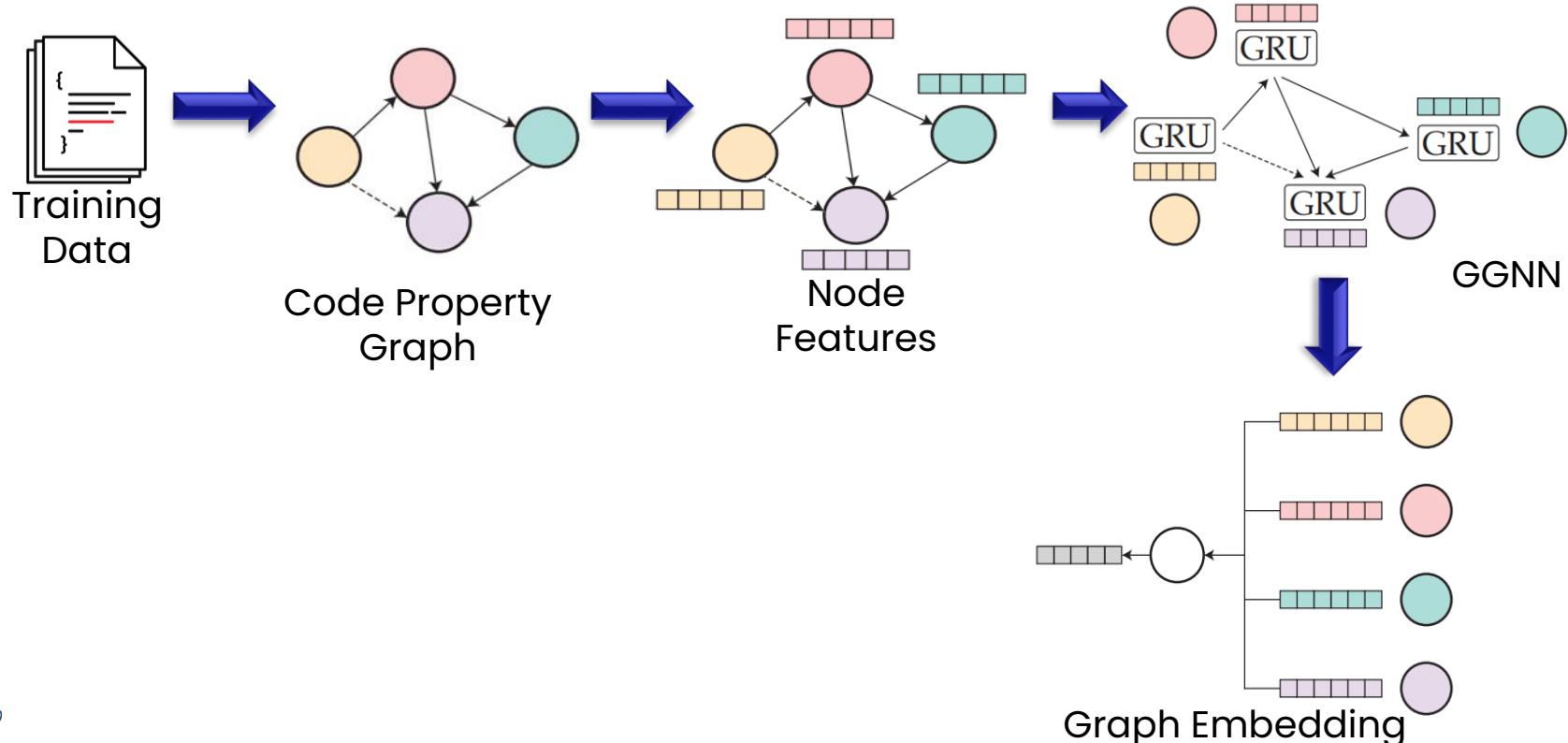
# ReVeal : Explicit Encoding for Program Understanding



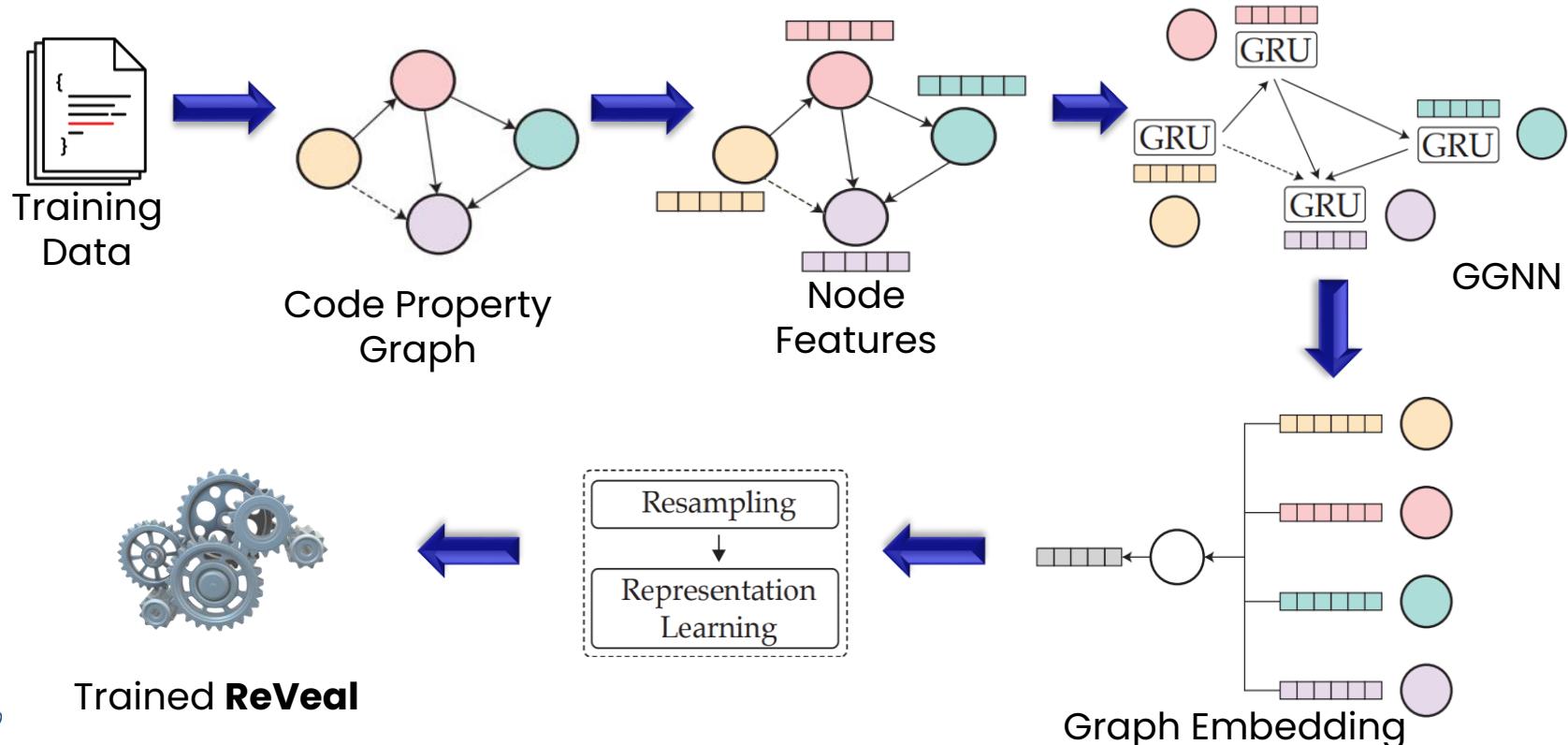
# ReVeal : Explicit Encoding for Program Understanding



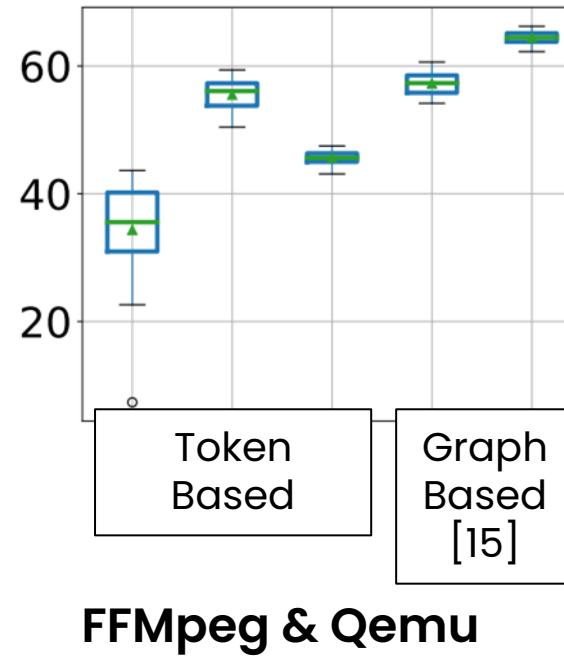
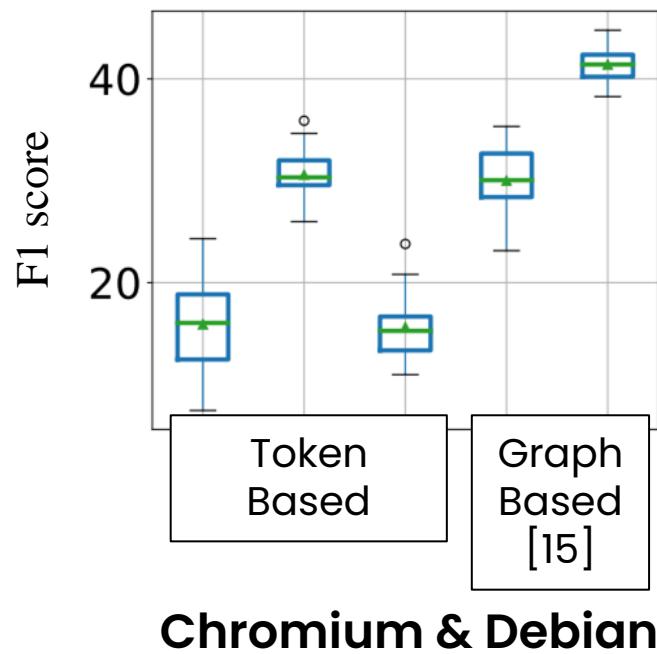
# ReVeal : Explicit Encoding for Program Understanding



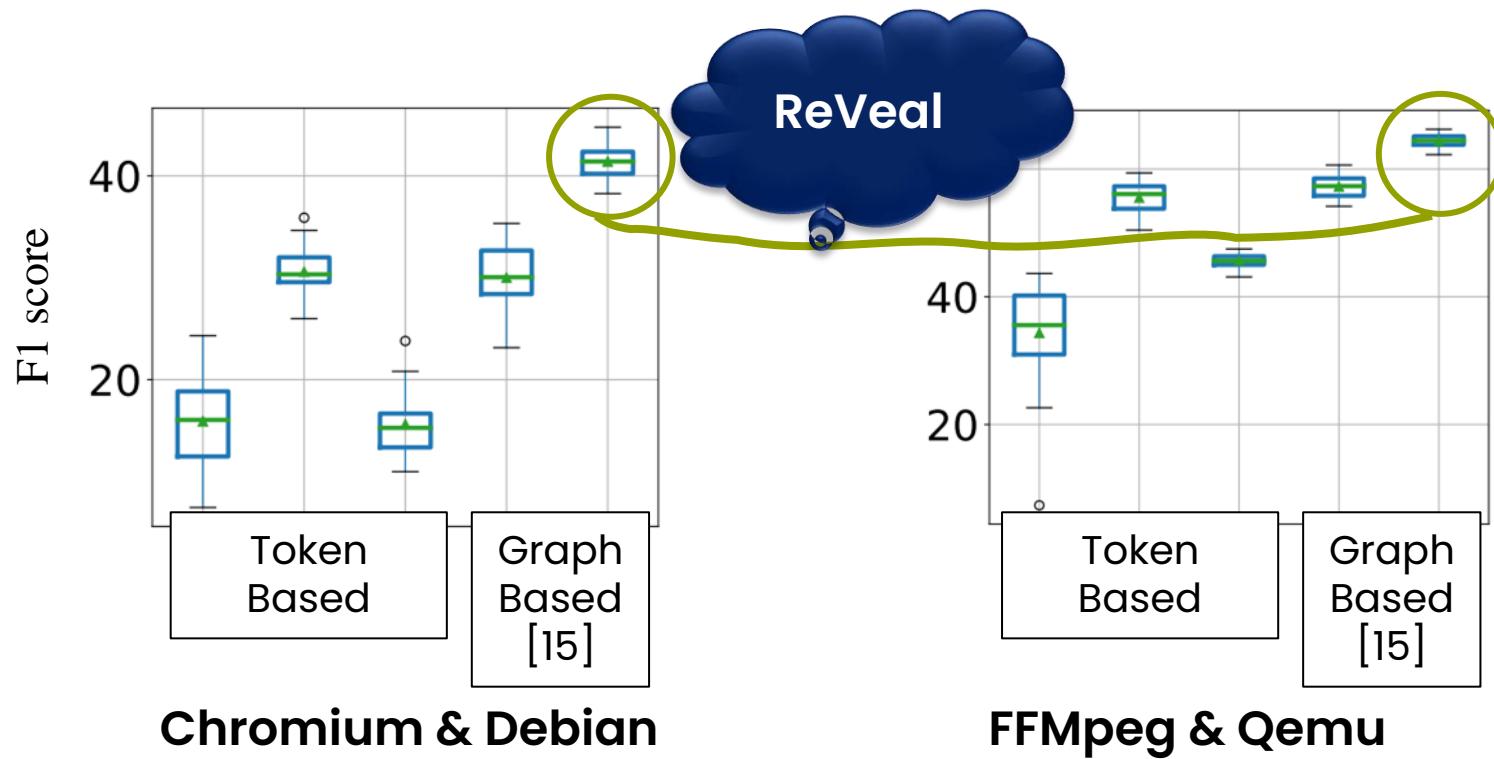
# ReVeal : Explicit Encoding for Program Understanding



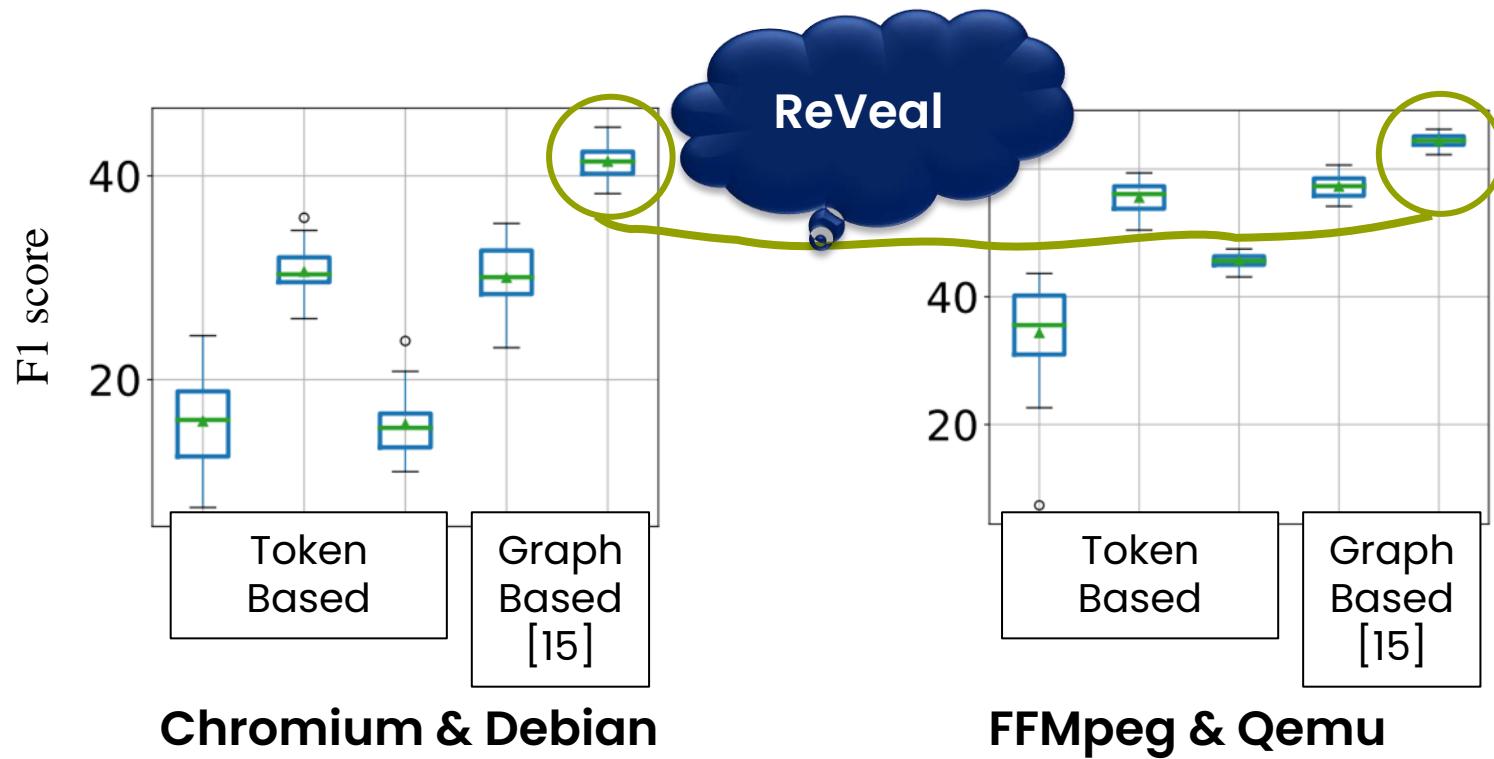
# ReVeal : Explicit Encoding for Program Understanding



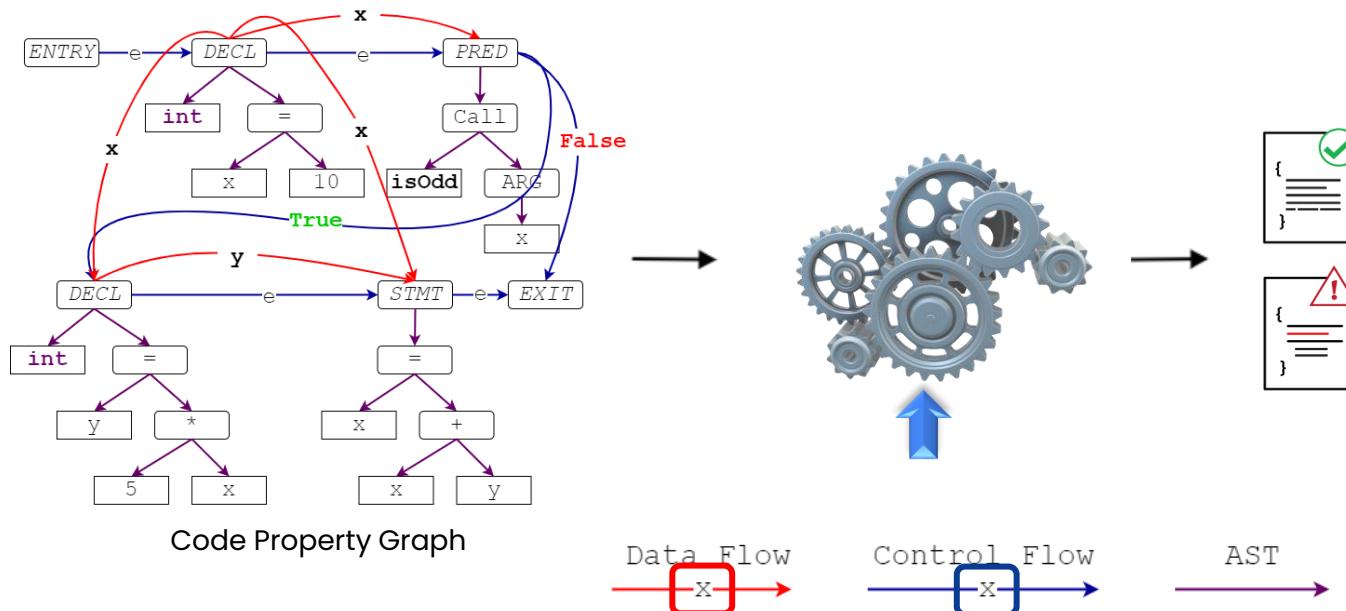
# ReVeal : Explicit Encoding for Program Understanding



# ReVeal : Explicit Encoding for Program Understanding



# Explicit Encoding PL knowledge into model



# PLBART Results

- Code Summarization

| Methods     | Ruby         | Javascript   | Go           | Python       | Java         | PHP          | Overall      |
|-------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
| Seq2Seq     | 9.64         | 10.21        | 13.98        | 15.93        | 15.09        | 21.08        | 14.32        |
| Transformer | 11.18        | 11.59        | 16.38        | 15.81        | 16.26        | 22.12        | 15.56        |
| RoBERTa     | 11.17        | 11.90        | 17.72        | 18.14        | 16.47        | 24.02        | 16.57        |
| CodeBERT    | 12.16        | 14.90        | 18.07        | 19.06        | 17.65        | <b>25.16</b> | 17.83        |
| PLBART      | <b>14.11</b> | <b>15.56</b> | <b>18.91</b> | <b>19.30</b> | <b>18.45</b> | 23.58        | <b>18.32</b> |

# PLBART Results

- Code Generation from Natural Language

| Methods               | EM           | BLEU         | CodeBLEU     |
|-----------------------|--------------|--------------|--------------|
| Seq2Seq               | 3.05         | 21.31        | 26.39        |
| Guo et al. (2019)     | 10.05        | 24.40        | 29.46        |
| Iyer et al. (2019)    | 12.20        | 26.60        | -            |
| GPT-2                 | 17.35        | 25.37        | 29.69        |
| CodeGPT-2             | 18.25        | 28.69        | 32.71        |
| CodeGPT-adapted       | <b>20.10</b> | 32.79        | 35.98        |
| PLBART                | 18.75        | <b>36.69</b> | <b>38.52</b> |
| PLBART <sub>10K</sub> | 17.25        | 31.40        | 33.32        |
| PLBART <sub>20K</sub> | 18.45        | 34.00        | 35.75        |
| PLBART <sub>50K</sub> | 17.70        | 35.02        | 37.11        |

# PLBART Results

- Code Translation

| Methods        | Java to C#   |              |              | C# to Java   |              |              |
|----------------|--------------|--------------|--------------|--------------|--------------|--------------|
|                | BLEU         | EM           | CodeBLEU     | BLEU         | EM           | CodeBLEU     |
| Naive Copy     | 18.54        | 0            | 34.20        | 18.69        | 0            | 43.04        |
| PBSMT          | 43.53        | 12.50        | 42.71        | 40.06        | 16.10        | 43.48        |
| Transformer    | 55.84        | 33.00        | 63.74        | 50.47        | 37.90        | 61.59        |
| RoBERTa (code) | 77.46        | 56.10        | 83.07        | 71.99        | 57.90        | 80.18        |
| CodeBERT       | 79.92        | 59.00        | 85.10        | 72.14        | 58.80        | 79.41        |
| GraphCodeBERT  | 80.58        | 59.40        | -            | 72.64        | 58.80        | -            |
| PLBART         | <b>83.02</b> | <b>64.60</b> | <b>87.92</b> | <b>78.35</b> | <b>65.00</b> | <b>85.27</b> |

# PLBART Results

- Code Classification - % Accuracy for Vulnerability, F1 score for Clone Detection.

| Tasks         | Vulnerability<br>Detection | Clone<br>Detection |
|---------------|----------------------------|--------------------|
| Transformer   | 61.64                      | -                  |
| CodeBERT      | 62.08                      | 96.5               |
| GraphCodeBERT | -                          | 97.1               |
| PLBART        | <b>63.18</b>               | <b>97.2</b>        |

## Code Translation Example

Input Code : C#

```
1 public int GetCells() {
2     int size = 0;
3     foreach (char c in cells.Keys) {
4         Cell e = At(c);
5         if (e.cmd >= 0 || e.@ref >= 0) {
6             size++;
7         }
8     }
9     return size;
10 }
```

Generated Code : Java

```
1 public int getCells() {
2     Iterator<Character> i =
3         cells.keySet().iterator();
4     int size = 0;
5     for (; i.hasNext();) {
6         Character c = i.next();
7         Cell e = at(c);
8         if (e.cmd >= 0 || e.ref >= 0) {
9             size++;
10        }
11    }
12    return size;
13 }
```

## Code Translation Example

Input Code : C#

```
1 public int GetCells() {
2     int size = 0;
3     foreach (char c in cells.Keys) {
4         Cell e = At(c);
5         if (e.cmd >= 0 || e.@ref >= 0) {
6             size++;
7         }
8     }
9     return size;
10 }
```

Generated Code : Java

```
1 public int getCells() {
2     Iterator<Character> i =
3         cells.keySet().iterator();
4     int size = 0;
5     for (; i.hasNext()); {
6         Character c = i.next();
7         Cell e = at(c);
8         if (e.cmd >= 0 || e.ref >= 0) {
9             size++;
10        }
11    }
12    return size;
13 }
```

## Code Translation Example

Input Code : C#

```
1 public int GetCells() {
2     int size = 0;
3     foreach (char c in cells.Keys) {
4         Cell e = At(c);
5         if (e.cmd >= 0 || e.@ref >= 0) {
6             size++;
7         }
8     }
9     return size;
10 }
```

Generated Code : Java

```
1 public int getCells() {
2     Iterator<Character> i =
3         cells.keySet().iterator();
4     int size = 0;
5     for (; i.hasNext()); {
6         Character c = i.next();
7         Cell e = at(c);
8         if (e.cmd >= 0 || e.ref >= 0) {
9             size++;
10        }
11    }
12    return size;
13 }
```

## Code Translation Example

Input Code : C#

```
1 public int GetCells() {
2     int size = 0;
3     foreach (char c in cells.Keys) {
4         Cell e = At(c);
5         if (e.cmd >= 0 || e.ref >= 0) {
6             size++;
7         }
8     }
9     return size;
10 }
```

Generated Code : Java

```
1 public int getCells() {
2     Iterator<Character> i =
3         cells.keySet().iterator();
4     int size = 0;
5     for (; i.hasNext();) {
6         Character c = i.next();
7         Cell e = at(c);
8         if (e.cmd >= 0 || e.ref >= 0) {
9             size++;
10        }
11    }
12    return size;
13 }
```



```
public void addPicture (String picture) {  
    if ((pictures) == null) {  
        pictures = new ArrayList<>();  
  
    }  
    pictures.add(picture);  
}
```



```
public void addPicture (String picture) {  
    if ((pictures) == null) {  
        pictures = new ArrayList<>();  
  
    }  
    pictures.add(picture);  
}
```



```
public void addPicture (String picture) {
    if ((pictures) == null) {
-        pictures = new ArrayList<>();
    }
    pictures.add(picture);
}
```

```
public void addPicture (String picture) {
    if ((pictures) == null) {
-        pictures = new ArrayList<>();
+        pictures = new HashSet<>();
    }
    pictures.add(picture);
}
```



```
public void addPicture (String picture) {
    if ((pictures) == null) {
-        pictures = new ArrayList<>();
    }
    pictures.add(picture);
}
```

```
public void addPicture (String picture) {
    if ((pictures) == null) {
-        pictures = new ArrayList<>();
+        pictures = new HashSet<>();
    }
    pictures.add(picture);
}
```

```
public void addPicture (String picture) {
    if ((pictures) == null) {
-        pictures = new ArrayList<>();
+        pictures = new LinkedList<>();
    }
    pictures.add(picture);
}
```



```
public void addPicture (String picture) {
    if ((pictures) == null) {
-        pictures = new ArrayList<>();
    }
    pictures.add(picture);
}
```

```
public void addPicture (String picture) {
    if ((pictures) == null) {
-        pictures = new ArrayList<>();
+        pictures = new HashSet<>();
    }
    pictures.add(picture);
}
```

```
public void addPicture (String picture) {
    if ((pictures) == null) {
-        pictures = new ArrayList<>();
+        pictures = new LinkedList<>();
    }
    pictures.add(picture);
}
```



```
public void addPicture (String picture) {  
    if ((pictures) == null) {  
-        pictures = new ArrayList<>();  
  
    }  
    pictures.add(picture);  
}
```

## use LinkedList and fix sublist problem

```
public void addPicture (String picture){  
    if ((pictures) == null) {  
-        pictures = new ArrayList<>();  
+        pictures = new HashSet<>();  
    }  
    pictures.add(picture);  
}
```

```
public void addPicture (String picture){  
    if ((pictures) == null) {  
-        pictures = new ArrayList<>();  
+        pictures = new LinkedList<>();  
    }  
    pictures.add(picture);  
}
```



```
public void addPicture (String picture) {
    if ((pictures) == null) {
-     pictures = new ArrayList<>();
}
    pictures.add(picture);
}
```

## use LinkedList and fix sublist problem

```
public void addPicture (String picture){
    if ((pictures) == null) {
-     pictures = new ArrayList<>();
+     pictures = new HashSet<>();
}
    pictures.add(picture);
}
```

```
public void addPicture (String picture) {
    if ((pictures) == null) {
-     pictures = new ArrayList<>();
+     pictures = new LinkedList<>();
}
    pictures.add(picture);
}
```



```
public void addPicture (String picture) {
    if ((pictures) == null) {
-     pictures = new ArrayList<>();
}
    pictures.add(picture);
}
```

## use LinkedList and fix sublist problem

```
public void addPicture (String picture){
    if ((pictures) == null) {
-     pictures = new ArrayList<>();
+     pictures = new HashSet<>();
}
    pictures.add(picture);
}
```

```
public void addPicture (String picture) {
    if ((pictures) == null) {
-     pictures = new ArrayList<>();
+     pictures = new LinkedList<>();
}
    pictures.add(picture);
}
```

# Transformers can learn syntax

| Model Name               | # of params (M) | Multi-Modal | Accuracy (%) |         |
|--------------------------|-----------------|-------------|--------------|---------|
|                          |                 |             | $B2F_s$      | $B2F_m$ |
| <i>Transformer-base</i>  | 139.22          | ✓           | 11.18        | 6.61    |
| <i>Transformer-large</i> | 406.03          | ✓           | 13.40        | 8.63    |
| CODIT                    | 105.43          | ✗           | 6.53         | 4.79    |

# Transformers can learn syntax

| Model Name               | # of params (M) | Multi-Modal | Accuracy (%) |         |
|--------------------------|-----------------|-------------|--------------|---------|
|                          |                 |             | $B2F_s$      | $B2F_m$ |
| <i>Transformer-base</i>  | 139.22          | ✓           | 11.18        | 6.61    |
| <i>Transformer-large</i> | 406.03          | ✓           | 13.40        | 8.63    |
| CODIT                    | 105.43          | ✗           | 6.53         | 4.79    |

# Pretrained models improves Code editing

| Training Type | Model Name               | # of params (M) | Multi-Modal | Accuracy (%) |              |
|---------------|--------------------------|-----------------|-------------|--------------|--------------|
|               |                          |                 |             | $B2F_s$      | $B2F_m$      |
| From Scratch  | LSTM                     | 82.89           | ✓           | 6.14         | 1.04         |
|               | <i>Transformer-base</i>  | 139.22          | ✓           | 11.18        | 6.61         |
|               | <i>Transformer-large</i> | 406.03          | ✓           | 13.40        | 8.63         |
|               | CODIT                    | 105.43          | ✗           | 6.53         | 4.79         |
| Fine-tuned    | CodeBERT                 | 172.50          | ✗           | 24.28        | 16.76        |
|               |                          |                 | ✓           | 26.05        | 17.13        |
|               | GraphCodeBERT            | 172.50          | ✗           | 24.44        | 16.85        |
|               |                          |                 | ✓           | 25.67        | 18.31        |
|               | CodeGPT                  | 124.44          | ✗           | 28.13        | 16.35        |
|               |                          |                 | ✓           | 28.43        | 17.64        |
|               | MODIT                    | 139.22          | ✗           | 26.67        | 19.79        |
|               |                          |                 | ✓           | <b>29.99</b> | <b>23.02</b> |

# Pretrained models improves Code editing

| Training Type | Model Name               | # of params (M) | Multi-Modal | Accuracy (%) |              |
|---------------|--------------------------|-----------------|-------------|--------------|--------------|
|               |                          |                 |             | $B2F_s$      | $B2F_m$      |
| From Scratch  | LSTM                     | 82.89           | ✓           | 6.14         | 1.04         |
|               | <i>Transformer-base</i>  | 139.22          | ✓           | 11.18        | 6.61         |
|               | <i>Transformer-large</i> | 406.03          | ✓           | 13.40        | 8.63         |
|               | CODIT                    | 105.43          | ✗           | 6.53         | 4.79         |
| Fine-tuned    | CodeBERT                 | 172.50          | ✗           | 24.28        | 16.76        |
|               |                          |                 | ✓           | 26.05        | 17.13        |
|               | GraphCodeBERT            | 172.50          | ✗           | 24.44        | 16.85        |
|               |                          |                 | ✓           | 25.67        | 18.31        |
|               | CodeGPT                  | 124.44          | ✗           | 28.13        | 16.35        |
|               |                          |                 | ✓           | 28.43        | 17.64        |
|               | MODIT                    | 139.22          | ✗           | 26.67        | 19.79        |
|               |                          |                 | ✓           | <b>29.99</b> | <b>23.02</b> |

# Pretrained models improves Code editing

| Training Type | Model Name               | # of params (M) | Multi-Modal | Accuracy (%) |              |
|---------------|--------------------------|-----------------|-------------|--------------|--------------|
|               |                          |                 |             | $B2F_s$      | $B2F_m$      |
| From Scratch  | LSTM                     | 82.89           | ✓           | 6.14         | 1.04         |
|               | <i>Transformer-base</i>  | 139.22          | ✓           | 11.18        | 6.61         |
|               | <i>Transformer-large</i> | 406.03          | ✓           | 13.40        | 8.63         |
|               | CODIT                    | 105.43          | ✗           | 6.53         | 4.79         |
| Fine-tuned    | CodeBERT                 | 172.50          | ✗           | 24.28        | 16.76        |
|               |                          |                 | ✓           | 26.05        | 17.13        |
|               | GraphCodeBERT            | 172.50          | ✗           | 24.44        | 16.85        |
|               |                          |                 | ✓           | 25.67        | 18.31        |
|               | CodeGPT                  | 124.44          | ✗           | 28.13        | 16.35        |
|               |                          |                 | ✓           | 28.43        | 17.64        |
|               | MODIT                    | 139.22          | ✗           | 26.67        | 19.79        |
|               |                          |                 | ✓           | <b>29.99</b> | <b>23.02</b> |

# Multi Modality Improves Code Editing

| Training Type | Model Name    | # of params (M) | Multi-Modal | Accuracy (%) |              |
|---------------|---------------|-----------------|-------------|--------------|--------------|
|               |               |                 |             | $B2F_s$      | $B2F_m$      |
| Fine-tuned    | CodeBERT      | 172.50          | ✗           | 24.28        | 16.76        |
|               |               |                 | ✓           | 26.05        | 17.13        |
|               | GraphCodeBERT | 172.50          | ✗           | 24.44        | 16.85        |
|               |               |                 | ✓           | 25.67        | 18.31        |
|               | CodeGPT       | 124.44          | ✗           | 28.13        | 16.35        |
|               |               |                 | ✓           | 28.43        | 17.64        |
|               | MODIT         | 139.22          | ✗           | 26.67        | 19.79        |
|               |               |                 | ✓           | <b>29.99</b> | <b>23.02</b> |

# Multi Modality Improves Code Editing

| Training Type | Model Name    | # of params (M) | Multi-Modal | Accuracy (%) |              |
|---------------|---------------|-----------------|-------------|--------------|--------------|
|               |               |                 |             | $B2F_s$      | $B2F_m$      |
| Fine-tuned    | CodeBERT      | 172.50          | ✗           | 24.28        | 16.76        |
|               |               |                 | ✓           | 26.05        | 17.13        |
|               | GraphCodeBERT | 172.50          | ✗           | 24.44        | 16.85        |
|               |               |                 | ✓           | 25.67        | 18.31        |
|               | CodeGPT       | 124.44          | ✗           | 28.13        | 16.35        |
|               |               |                 | ✓           | 28.43        | 17.64        |
|               | MODIT         | 139.22          | ✗           | 26.67        | 19.79        |
|               |               |                 | ✓           | <b>29.99</b> | <b>23.02</b> |

# Multi Modality Improves Code Editing

| Training Type | Model Name    | # of params (M) | Multi-Modal | Accuracy (%) |              |
|---------------|---------------|-----------------|-------------|--------------|--------------|
|               |               |                 |             | $B2F_s$      | $B2F_m$      |
| Fine-tuned    | CodeBERT      | 172.50          | ✗           | 24.28        | 16.76        |
|               |               |                 | ✓           | 26.05        | 17.13        |
|               | GraphCodeBERT | 172.50          | ✗           | 24.44        | 16.85        |
|               |               |                 | ✓           | 25.67        | 18.31        |
|               | CodeGPT       | 124.44          | ✗           | 28.13        | 16.35        |
|               |               |                 | ✓           | 28.43        | 17.64        |
|               | MODIT         | 139.22          | ✗           | 26.67        | 19.79        |
|               |               |                 | ✓           | <b>29.99</b> | <b>23.02</b> |

# Multi Modality Improves Code Editing

| Training Type | Model Name    | # of params (M) | Multi-Modal | Accuracy (%) |              |
|---------------|---------------|-----------------|-------------|--------------|--------------|
|               |               |                 |             | $B2F_s$      | $B2F_m$      |
| Fine-tuned    | CodeBERT      | 172.50          | ✗           | 24.28        | 16.76        |
|               |               |                 | ✓           | 26.05        | 17.13        |
|               | GraphCodeBERT | 172.50          | ✗           | 24.44        | 16.85        |
|               |               |                 | ✓           | 25.67        | 18.31        |
|               | CodeGPT       | 124.44          | ✗           | 28.13        | 16.35        |
|               |               |                 | ✓           | 28.43        | 17.64        |
|               | MODIT         | 139.22          | ✗           | 26.67        | 19.79        |
|               |               |                 | ✓           | <b>29.99</b> | <b>23.02</b> |

# Multi Modality Improves Code Editing

| Training Type | Model Name    | # of params (M) | Multi-Modal | Accuracy (%) |              |
|---------------|---------------|-----------------|-------------|--------------|--------------|
|               |               |                 |             | $B2F_s$      | $B2F_m$      |
| Fine-tuned    | CodeBERT      | 172.50          | ✗           | 24.28        | 16.76        |
|               |               |                 | ✓           | 26.05        | 17.13        |
|               | GraphCodeBERT | 172.50          | ✗           | 24.44        | 16.85        |
|               |               |                 | ✓           | 25.67        | 18.31        |
|               | CodeGPT       | 124.44          | ✗           | 28.13        | 16.35        |
|               |               |                 | ✓           | 28.43        | 17.64        |
|               | MODIT         | 139.22          | ✗           | 26.67        | 19.79        |
|               |               |                 | ✓           | <b>29.99</b> | <b>23.02</b> |

# Impact of Different Information Modality.

| Technique                   | Accuracy (%) |              |
|-----------------------------|--------------|--------------|
|                             | $B2F_s$      | $B2F_m$      |
| MODIT                       | <b>29.99</b> | <b>23.02</b> |
| — Context                   | 28.76        | 21.63        |
| — Guidance                  | 29.79        | 21.40        |
| — Both Context and Guidance | 26.67        | 19.79        |

# Impact of Different Information Modality.

| Technique                 | Accuracy (%) |              |
|---------------------------|--------------|--------------|
|                           | $B2F_s$      | $B2F_m$      |
| MODIT                     | <b>29.99</b> | <b>23.02</b> |
| Context                   | 28.76        | 21.63        |
| Guidance                  | 29.79        | 21.40        |
| Both Context and Guidance | 26.67        | 19.79        |

# Impact of Different Information Modality.

| Technique                 | Accuracy (%) |              |
|---------------------------|--------------|--------------|
|                           | $B2F_s$      | $B2F_m$      |
| MODIT                     | <b>29.99</b> | <b>23.02</b> |
| Context                   | 28.76        | 21.63        |
| Guidance                  | 29.79        | 21.40        |
| Both Context and Guidance | 26.67        | 19.79        |

# Impact of Different Information Modality.

| Technique                 | Accuracy (%) |              |
|---------------------------|--------------|--------------|
|                           | $B2F_s$      | $B2F_m$      |
| MODIT                     | <b>29.99</b> | <b>23.02</b> |
| Context                   | 28.76        | 21.63        |
| Guidance                  | 29.79        | 21.40        |
| Both Context and Guidance | 26.67        | 19.79        |

# Impact of Different Information Modality.

| Technique                               | Accuracy (%) |              |
|---|--------------|--------------|
|   | $B2F_s$      | $B2F_m$      |
| MODIT                                   | <b>29.99</b> | <b>23.02</b> |
| Context                                 | 28.76        | 21.63        |
| Guidance                                | 29.79        | 21.40        |
| Both Context and Guidance               | 26.67        | 19.79        |
| SequqnceR [Chen <i>et al.</i> , TSE 19] | 13.03        | 4.53         |
| SequenceR + Guidance                    | 17.90        | 4.60         |

# Impact of Different Information Modality.

## Impact of Guidance

```
// Guidance: fixed some bugs in type checking
// improved performance by caching types of expressions
private TypeCheckInfo getType(SadlUnionType expression) {
    ...
    return new TypeCheckInfo(
        - declarationConceptName, declarationConceptName
        /* MODIT generated patch with guidance */
        + declarationConceptName, declarationConceptName,
        + this, expression
        /* MODIT generated patch without guidance */
        + this.declarationConceptName,
        + this.declarationConceptName
    );
}
```

# Impact of Different Information Modality.

## Impact of Guidance

```
// Guidance: fixed some bugs in type checking
// improved performance by caching types of expressions
private TypeCheckInfo getType(SadlUnionType expression) {
    ...
    return new TypeCheckInfo(
        - declarationConceptName, declarationConceptName
        /* MODIT generated patch with guidance */
        + declarationConceptName, declarationConceptName,
        + this, expression
        /* MODIT generated patch without guidance */
        + this.declarationConceptName,
        + this.declarationConceptName
    );
}
```

# Impact of Different Information Modality.

## Impact of Context

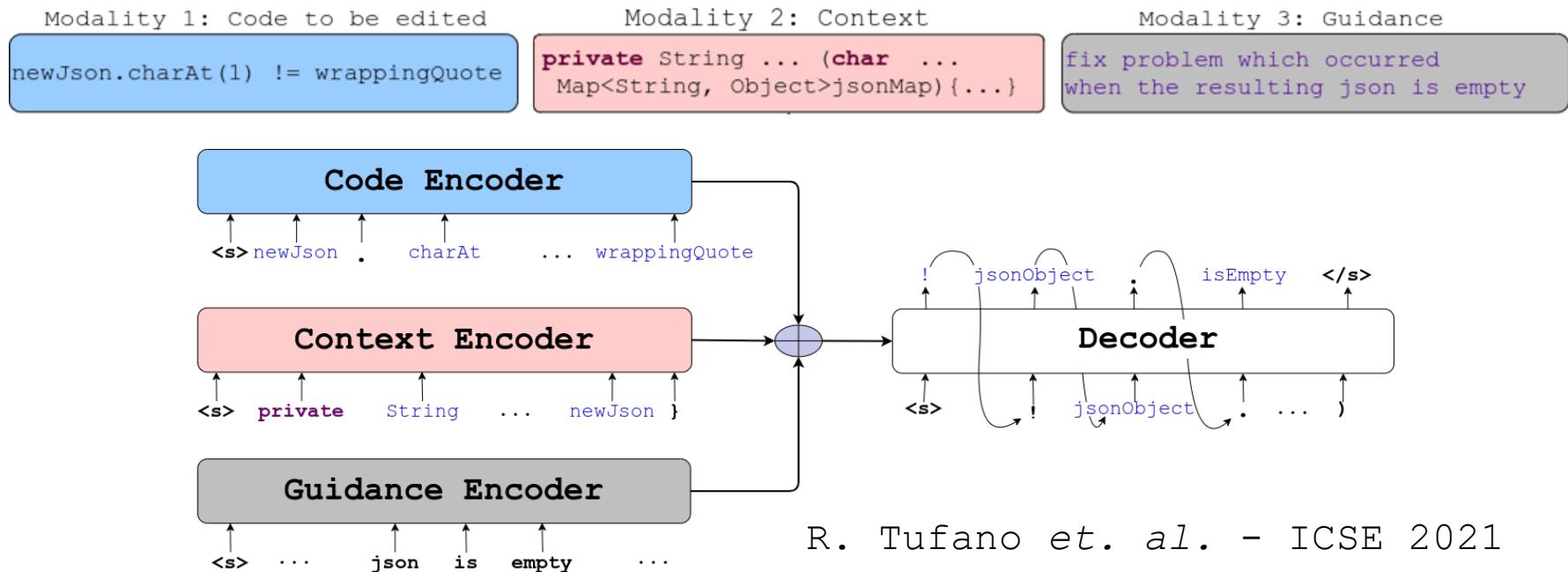
```
// Guidance: Fix bug of sending wrong message
public void setPredecessor (model.Message m) {
    this.predecessor = Integer.valueOf(m.Content);
    model.Message sent = new model.Message();
    sent.To = m.Origin;
    - sendMessage(m);
    /* MODIT generates with the context. */
    + sendMessage(sent);
    /* MODIT generates without context as input. */
    + sendMessage(m.toString());
}
```

# Impact of Different Information Modality.

## Impact of Context

```
// Guidance: Fix bug of sending wrong message
public void setPredecessor (model.Message m) {
    this.predecessor = Integer.valueOf(m.Content);
    model.Message sent = new model.Message();
    sent.To = m.Origin;
    - sendMessage(m);
    /* MODIT generates with the context. */
    + sendMessage(sent);
    /* MODIT generates without context as input. */
    + sendMessage(m.toString());
}
```

# Best way to encode input Modalities.



R. Tufano et. al. - ICSE 2021

# Best way to encode input Modalities.

| # of<br>Modalities | # of<br>Encoders | Accuracy (%) |         |
|--------------------|------------------|--------------|---------|
|                    |                  | $B2F_s$      | $B2F_m$ |
|                    |                  |              |         |
|                    |                  |              |         |
|                    |                  |              |         |

# Best way to encode input Modalities.

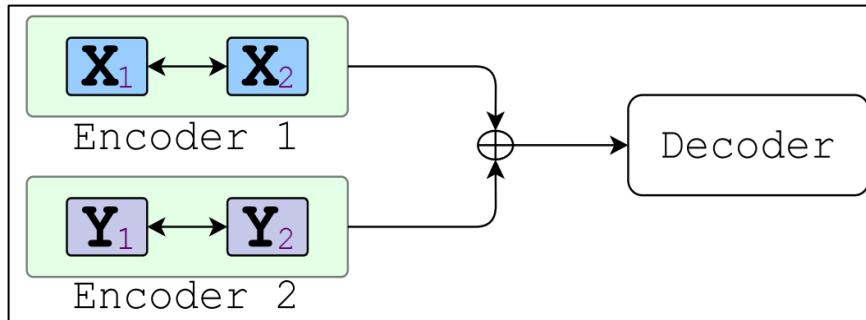
| # of<br>Modalities                | # of<br>Encoders | Accuracy (%) |              |
|-----------------------------------|------------------|--------------|--------------|
|                                   |                  | $B2F_s$      | $B2F_m$      |
| 3 ( $e_p$ , $\mathcal{G}$ , $C$ ) | 3                | 20.63        | 11.69        |
|                                   | 1                | <b>26.05</b> | <b>17.13</b> |

# Best way to encode input Modalities.

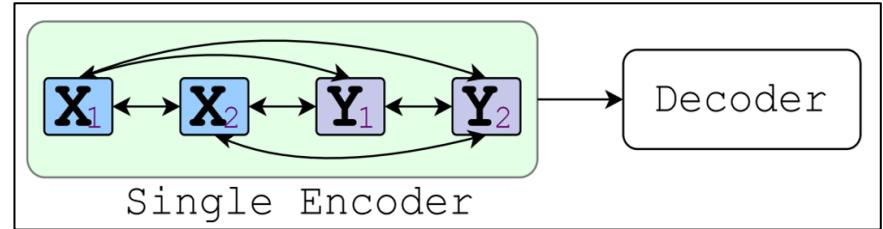
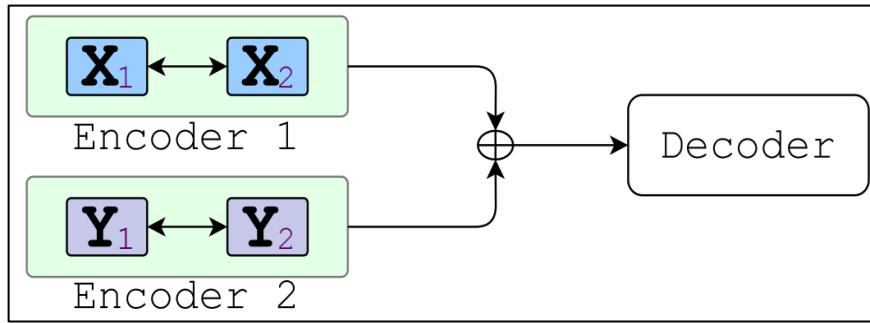
| # of<br>Modalities       | # of<br>Encoders | Accuracy (%) |              |
|--------------------------|------------------|--------------|--------------|
|                          |                  | $B2F_s$      | $B2F_m$      |
| 2 ( $e_p, \mathcal{G}$ ) | 2                | 23.12        | 15.49        |
|                          | 1                | <b>23.81</b> | <b>17.46</b> |

# Best way to encode input Modalities.

# Best way to encode input Modalities.



# Best way to encode input Modalities.



# NatGen Results

- NL to Code Generation

| Approach                  | EM                   | SM           | DM           | CB           |
|---------------------------|----------------------|--------------|--------------|--------------|
| Seq2Seq                   | 3.05                 | -            | -            | 26.39        |
| Guo et al. [30]           | 10.05                | -            | -            | 29.46        |
| Iyer et al. [37]          | 12.20                | -            | -            | -            |
| GPT-2                     | 17.30                | -            | -            | 29.69        |
| CodeGPT                   | 20.10                | -            | -            | 35.98        |
| PLBART                    | 18.75                | -            | -            | 38.52        |
| CodeT5-base<br>(reported) | <b>22.30</b>         | -            | -            | 43.20        |
| CodeT5*                   | $\mathcal{M}_{last}$ | 21.85        | 44.34        | 44.52        |
|                           | $\mathcal{M}_{best}$ | 21.55        | 41.08        | 43.71        |
| NATGEN                    | $\mathcal{M}_{last}$ | 22.25        | <b>45.59</b> | <b>46.87</b> |
|                           | $\mathcal{M}_{best}$ | <b>22.30</b> | 44.38        | 45.64        |

\* Our reproduced result using CodeT5's publicly available pre-trained model.

# NatGen Results

- Code Translations

| Approach             | Java → C#   |             |             |             | C# → Java   |             |             |             |
|----------------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
|                      | EM          | SM          | DM          | CB          | EM          | SM          | DM          | CB          |
| PBSTM                | 12.5        | -           | -           | 42.7        | 16.1        | -           | -           | 43.5        |
| CodeBERT             | 59.0        | -           | -           | 85.1        | 58.8        | -           | -           | 79.4        |
| SPT-Code             | 64.1        | -           | -           | -           | 60.2        | -           | -           | -           |
| PLBART               | 64.6        | -           | -           | 87.9        | 65.0        | -           | -           | <b>85.3</b> |
| CodeT5<br>(reported) | 65.9        | -           | -           | -           | 66.9        | -           | -           | -           |
| CodeT5*              | 65.9        | 90.4        | 91.9        | 87.8        | 66.0        | 90.4        | 88.9        | 84.4        |
| NATGEN               | <b>66.2</b> | <b>91.0</b> | <b>92.0</b> | <b>88.1</b> | <b>67.3</b> | <b>91.0</b> | <b>89.8</b> | 85.2        |

\* Our reproduced result using CodeT5's publicly available pre-trained model.

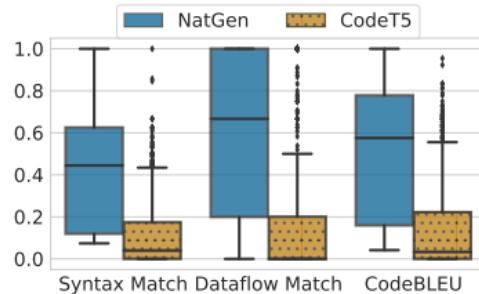
# NatGen Results

- Code Summarization

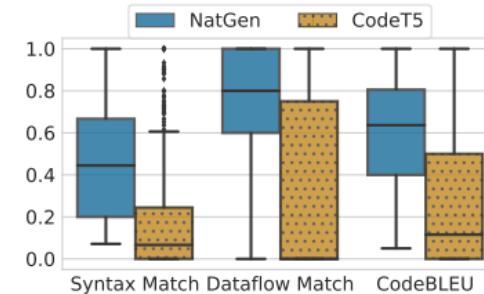
| Approach | Go           | Java         | JS           | Python       | Php          | Ruby         | Overall |
|----------|--------------|--------------|--------------|--------------|--------------|--------------|---------|
| PLBART   | 18.91        | 18.45        | 15.56        | 19.30        | 23.58        | 14.11        | 18.32   |
| CodeT5   | <b>19.56</b> | 20.31        | <b>16.16</b> | 20.01        | <b>26.03</b> | 15.24        | 19.55   |
| NATGEN   | 19.43        | <b>20.38</b> | 16.00        | <b>20.09</b> | 26.00        | <b>15.38</b> | 19.55   |

# NatGen Results

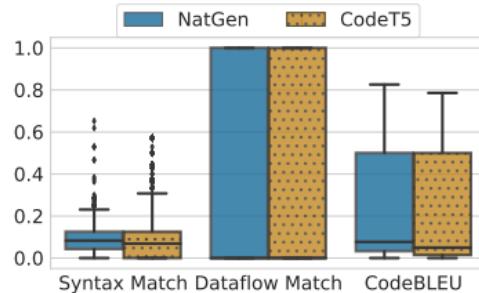
- Zero Shot Learning



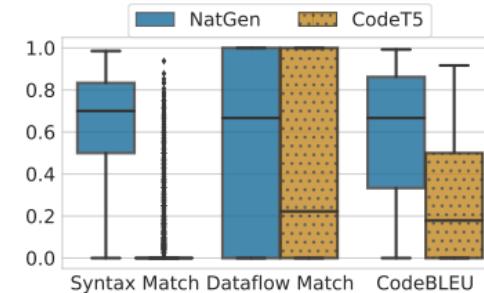
(a) Java to C# Translation



(b) C# to Java Translation



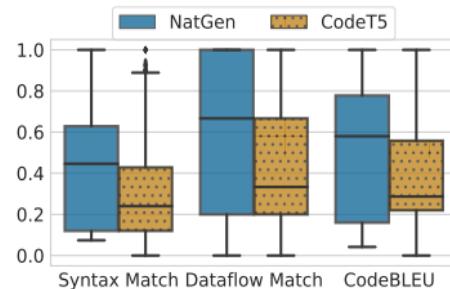
(c) Text to Code Generation



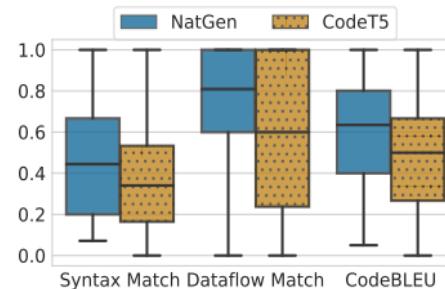
(d) Bug Fix (small, multimodal)

# NatGen Results

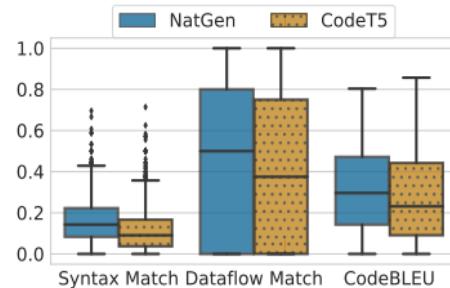
- Few Shot Learning



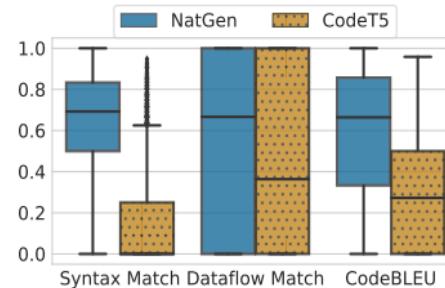
(a) Java to C# Translation



(b) C# to Java Translation



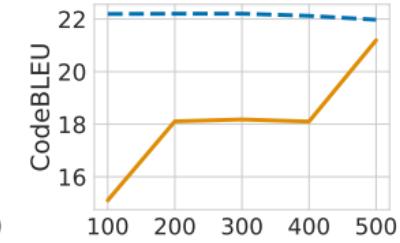
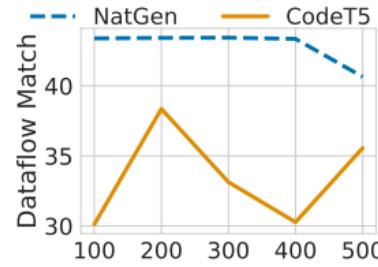
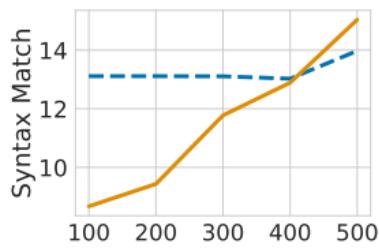
(c) Text to Code Generation



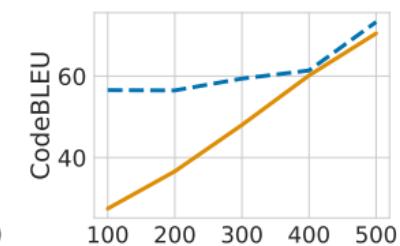
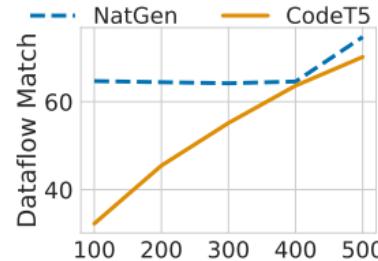
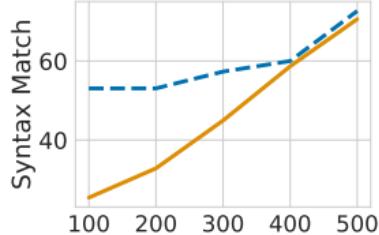
(d) Bug Fix (small, multimodal).

# NatGen Results

- Few Shot - Ablation



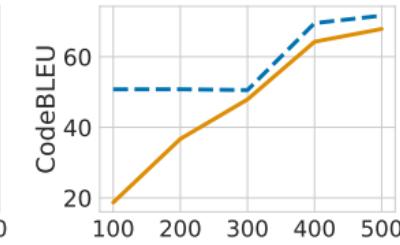
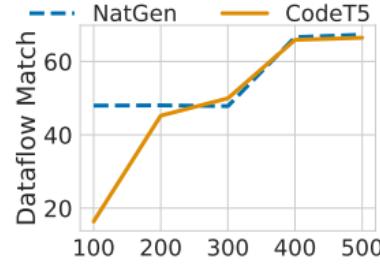
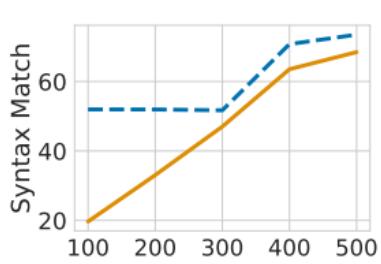
(a) Text to Code Generation.



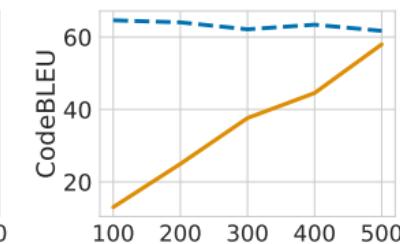
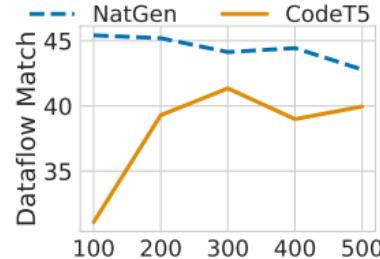
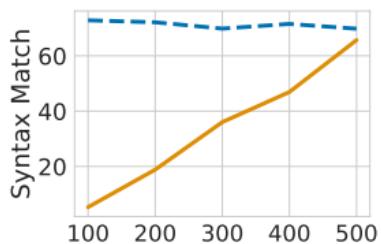
(b) C# to Java Translation.

# NatGen Results

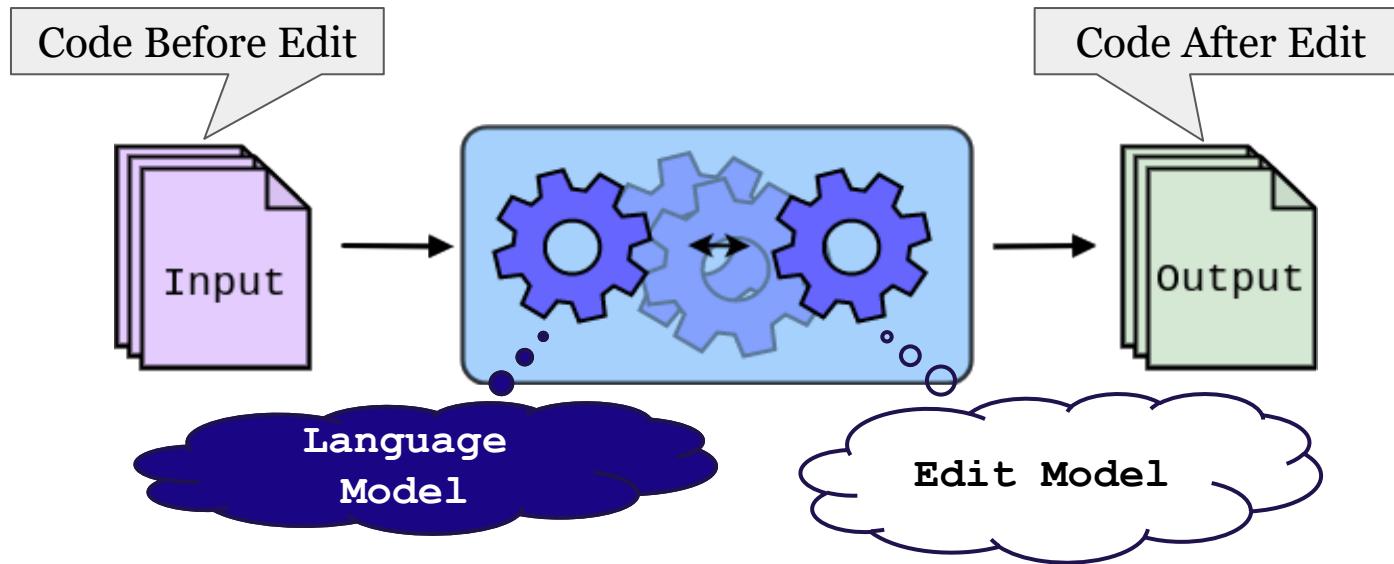
- Few Shot - Ablation

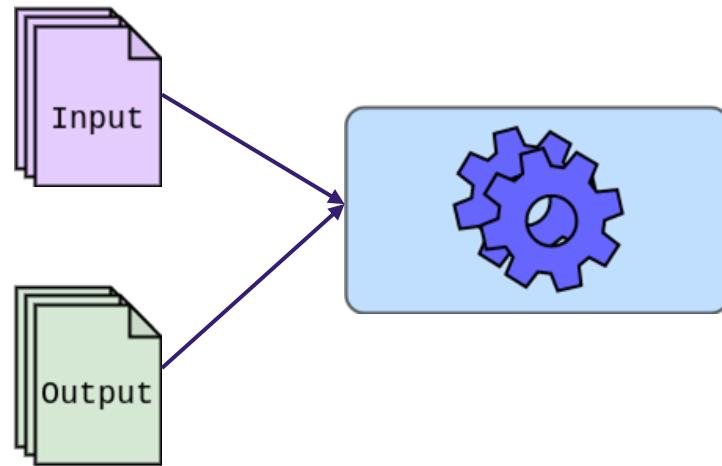


(c) Java to C# Translation.



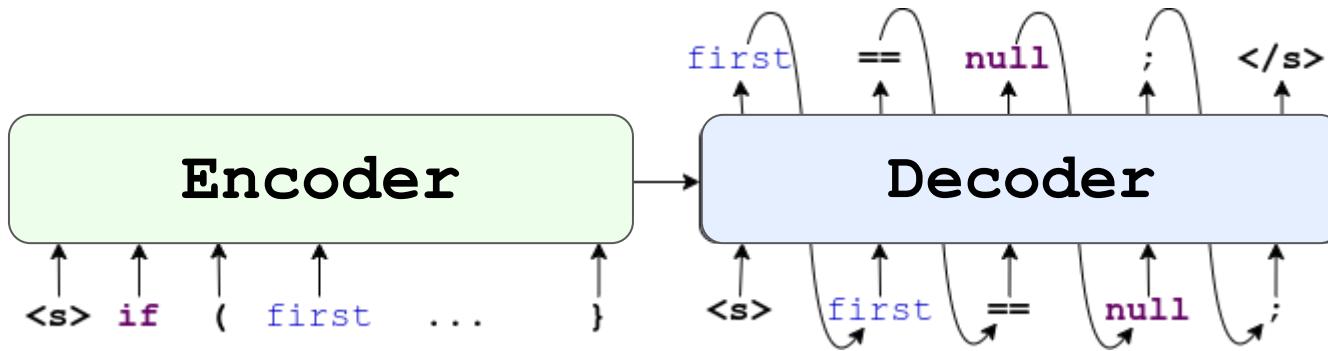
(d) Bug Fix (small, multimodal).



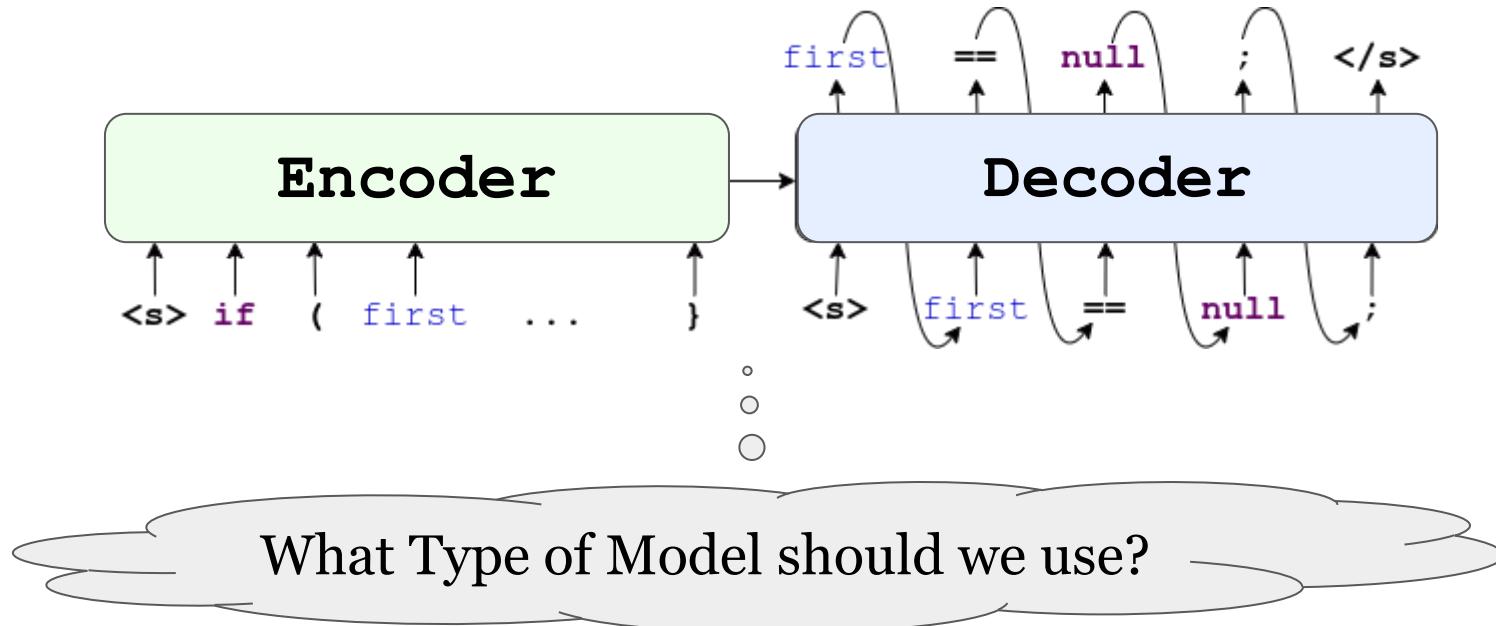


**Pretraining** - **Embed** the knowledge of **input and output** into the model.

# PLBART - Pretraining both encoder and decoder.



# PLBART - Pretraining both encoder and decoder.



# Recurrent Model Vs. Transformer Model

# Recurrent Model Vs. Transformer Model

## 1. Recurrent Model

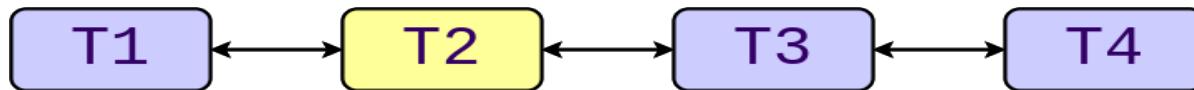
# Recurrent Model Vs. Transformer Model

## 1. Recurrent Model



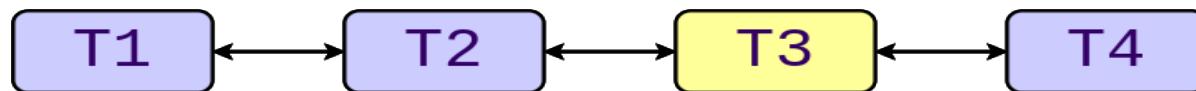
# Recurrent Model Vs. Transformer Model

## 1. Recurrent Model



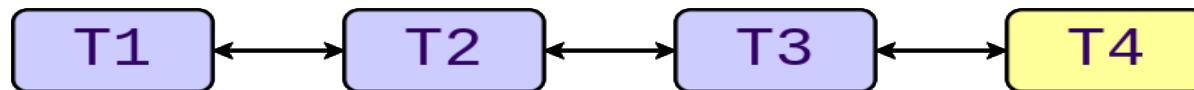
# Recurrent Model Vs. Transformer Model

## 1. Recurrent Model



# Recurrent Model Vs. Transformer Model

## 1. Recurrent Model



# Recurrent Model Vs. Transformer Model

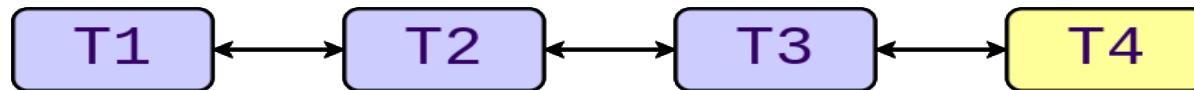
## 1. Recurrent Model



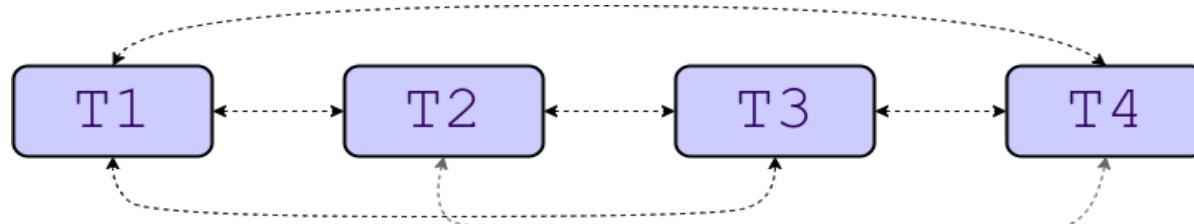
## 2. Transformer Model

# Recurrent Model Vs. Transformer Model

## 1. Recurrent Model

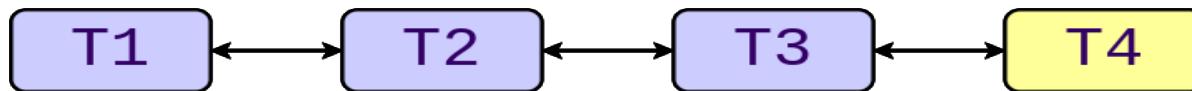


## 2. Transformer Model

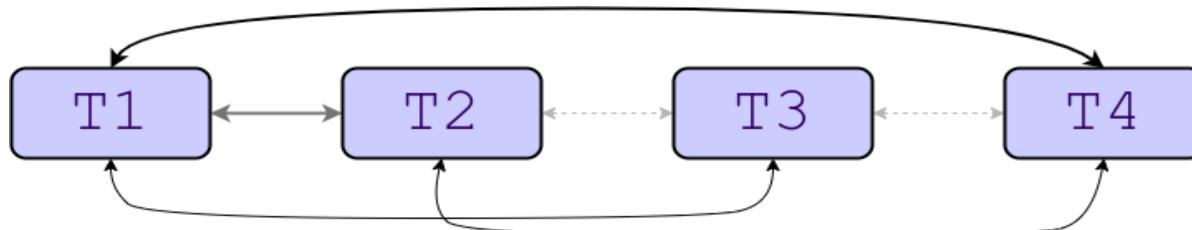


# Recurrent Model Vs. Transformer Model

## 1. Recurrent Model



## 2. Transformer Model

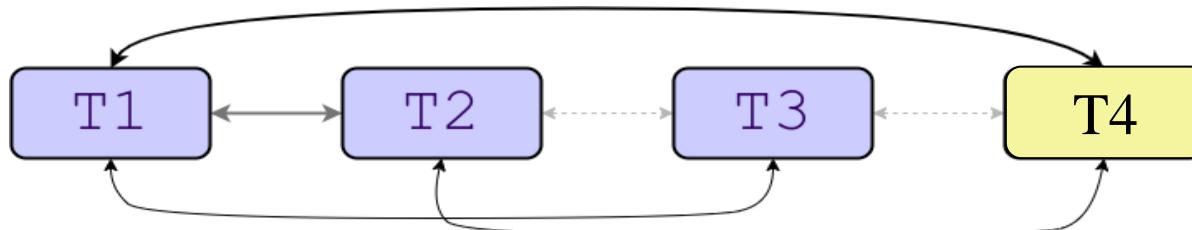


# Recurrent Model Vs. Transformer Model

## 1. Recurrent Model



## 2. Transformer Model

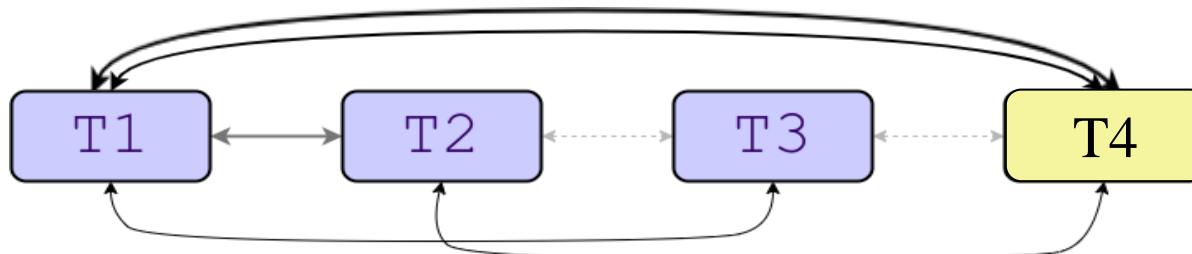


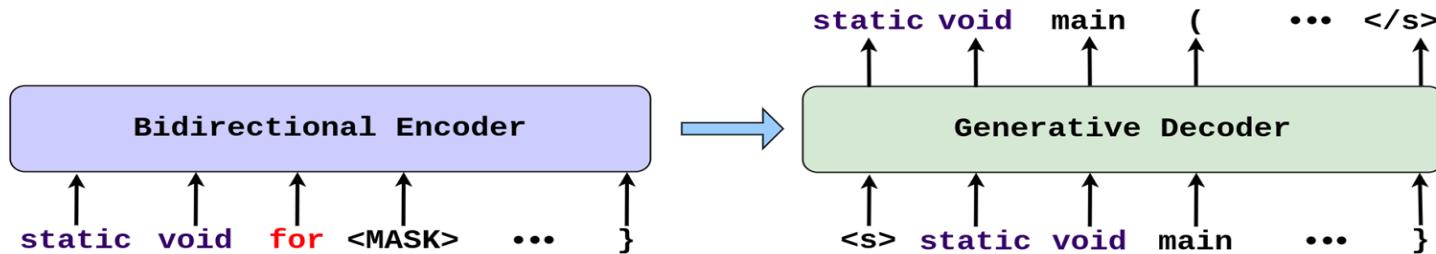
# Recurrent Model Vs. Transformer Model

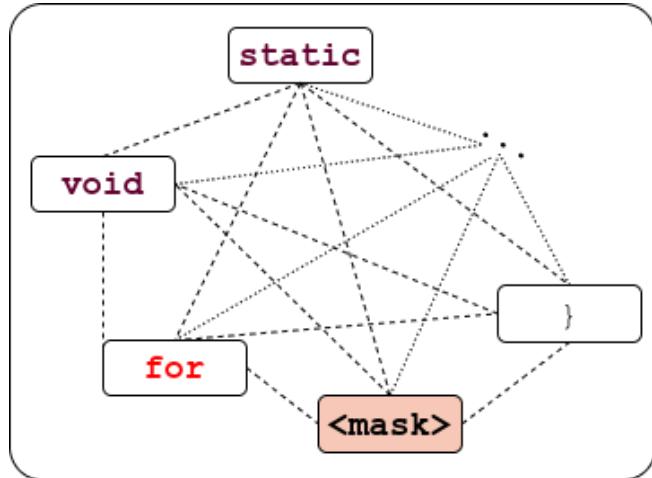
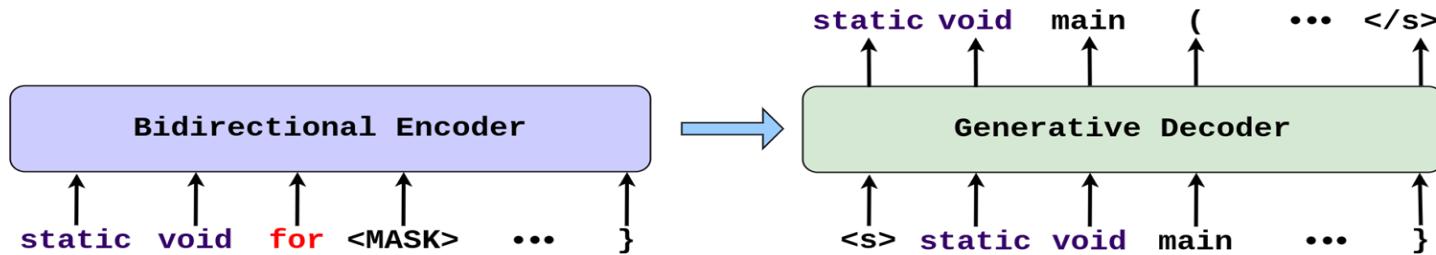
## 1. Recurrent Model

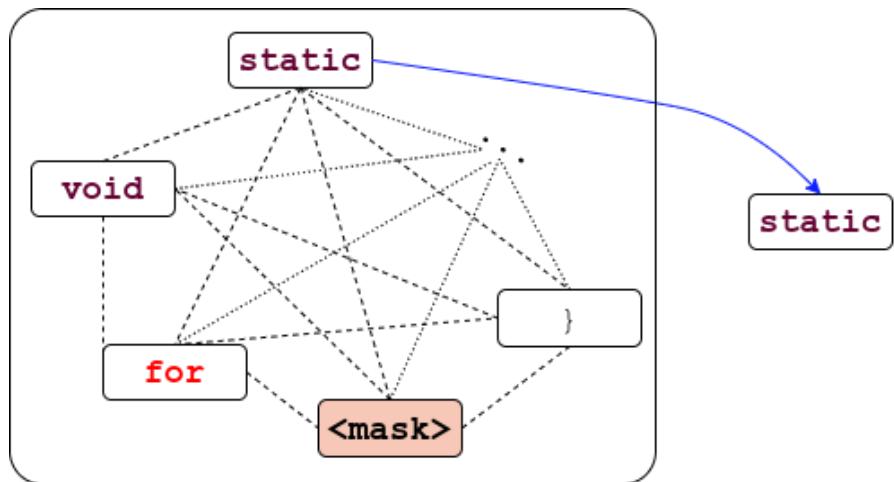
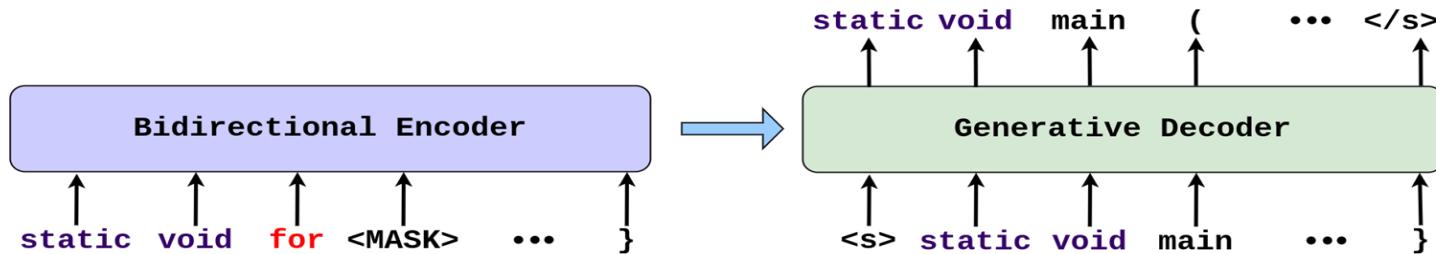


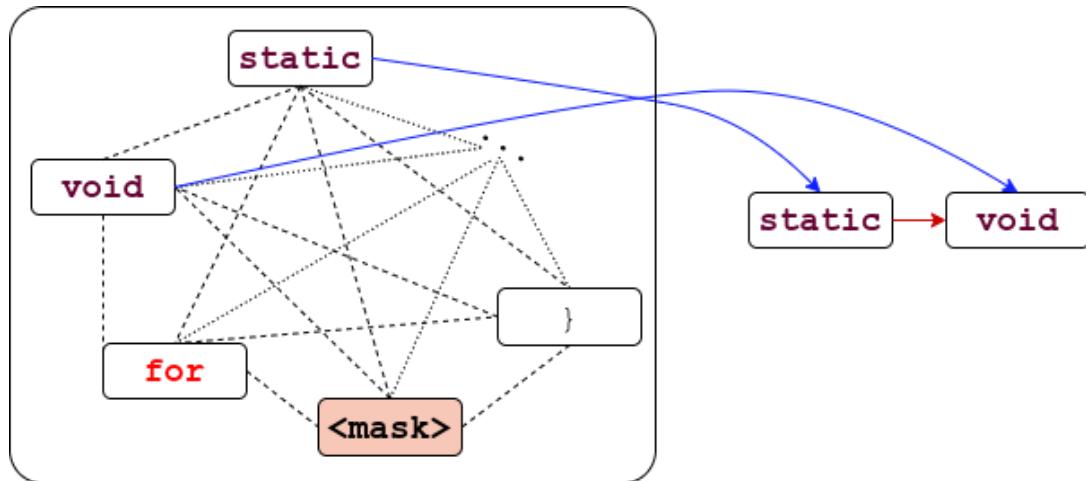
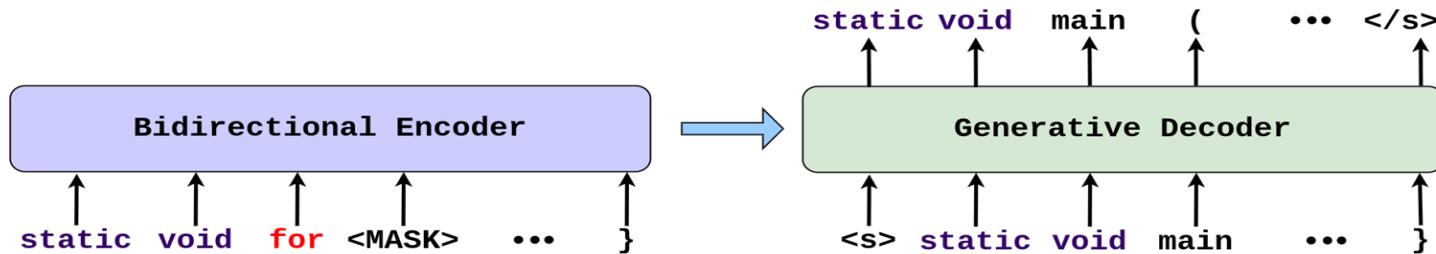
## 2. Transformer Model

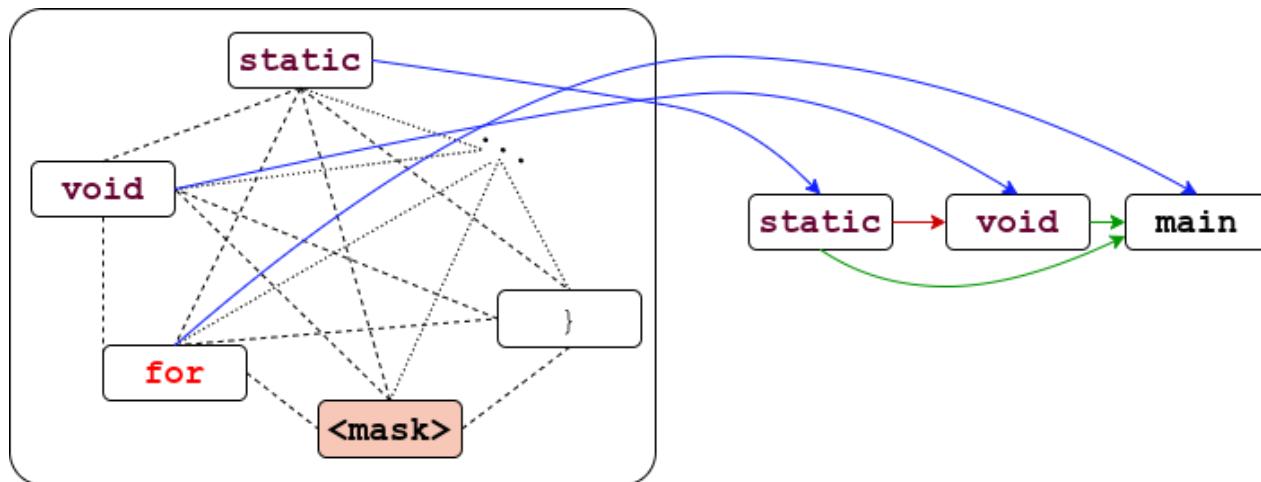
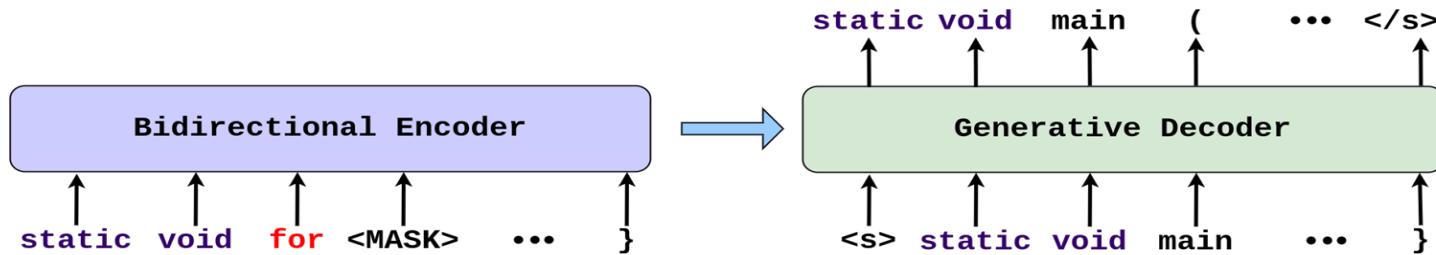


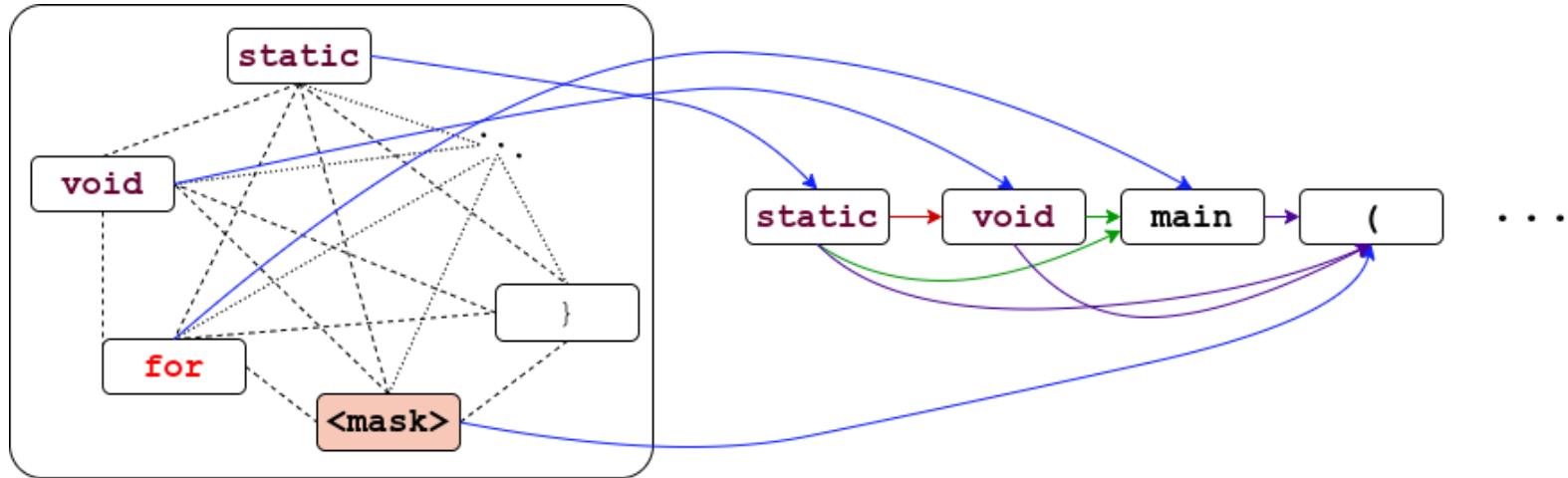
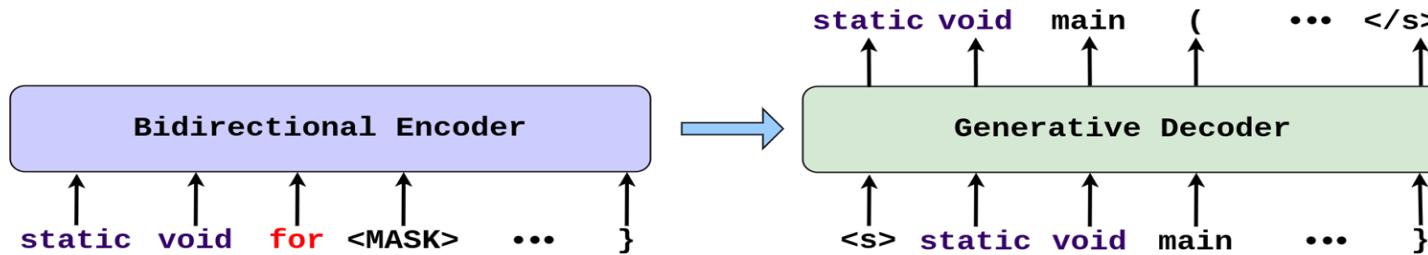


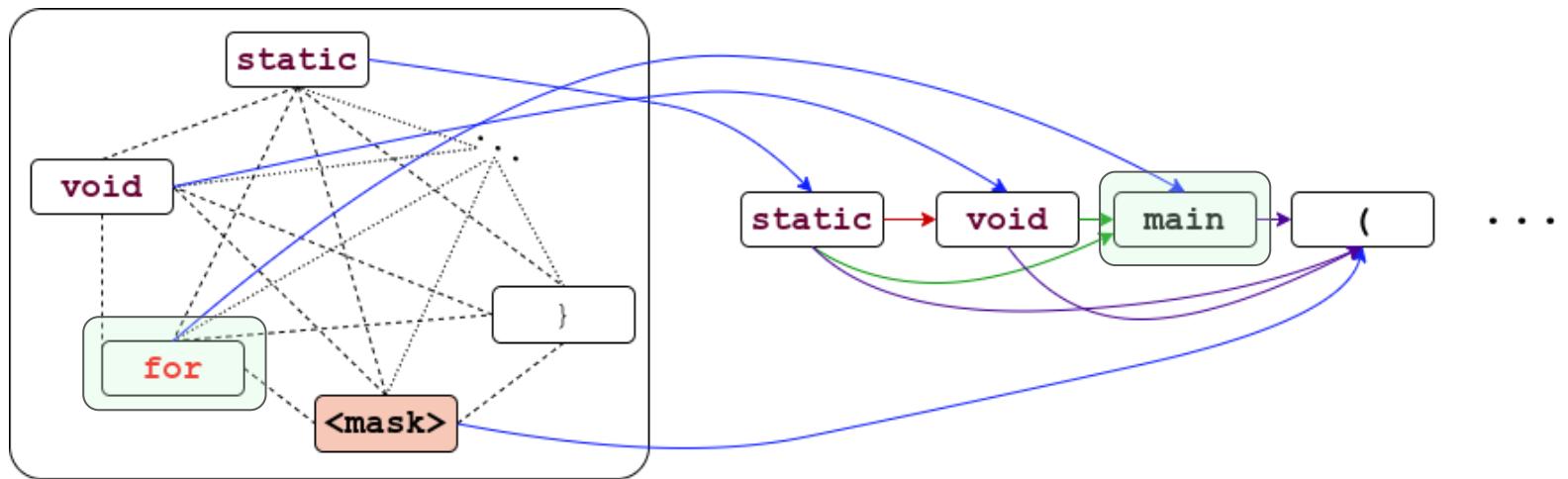
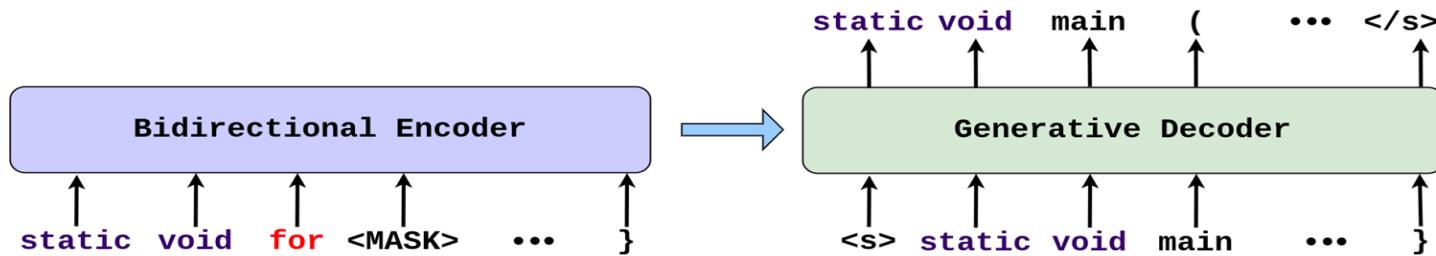




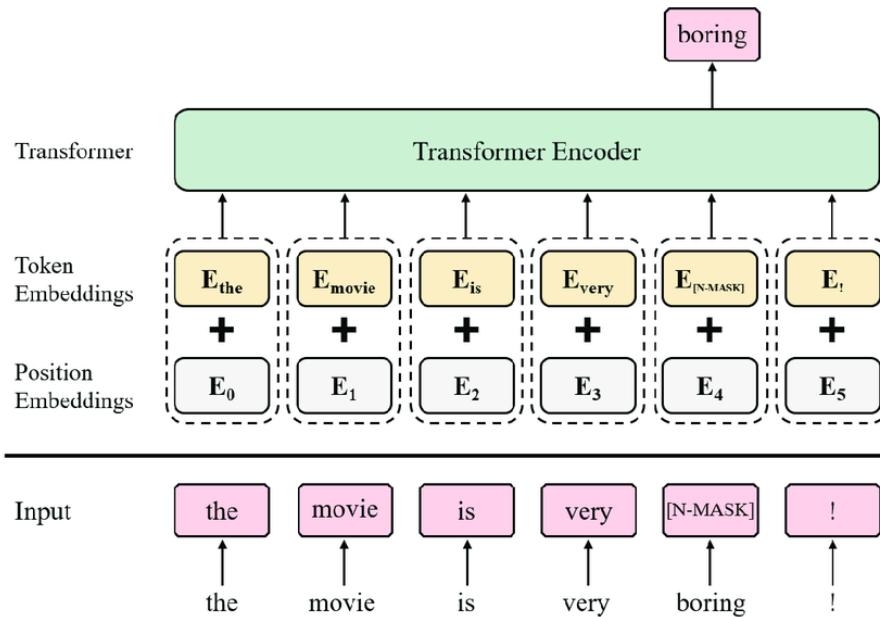




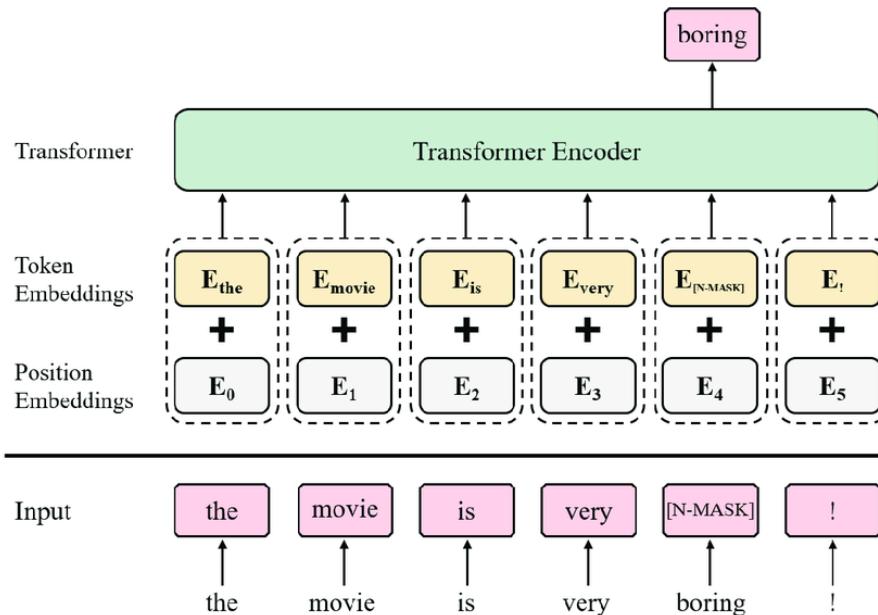




# BERT - Pretrained Transformer Encoder



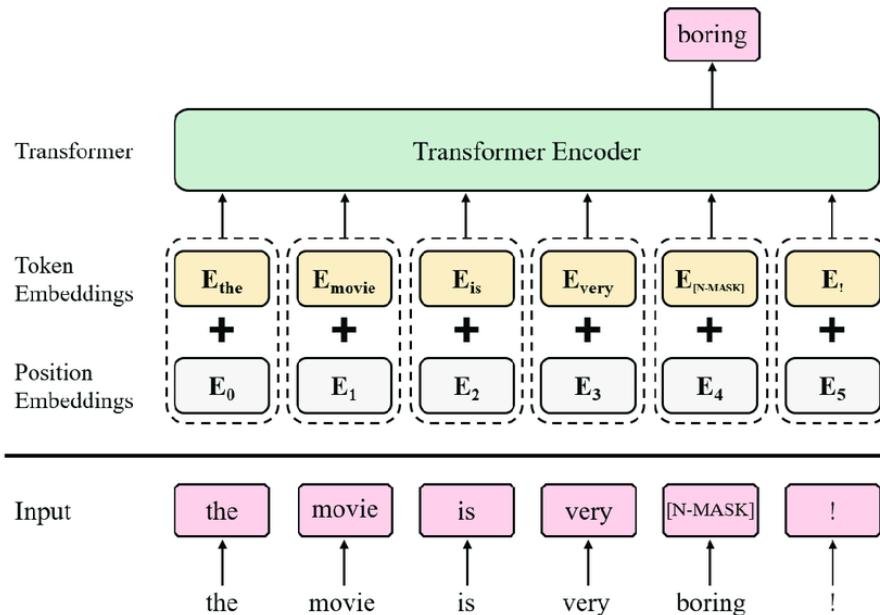
# BERT - Pretrained Transformer Encoder



**Pre-training:**

Task agnostic Masked Language Model.

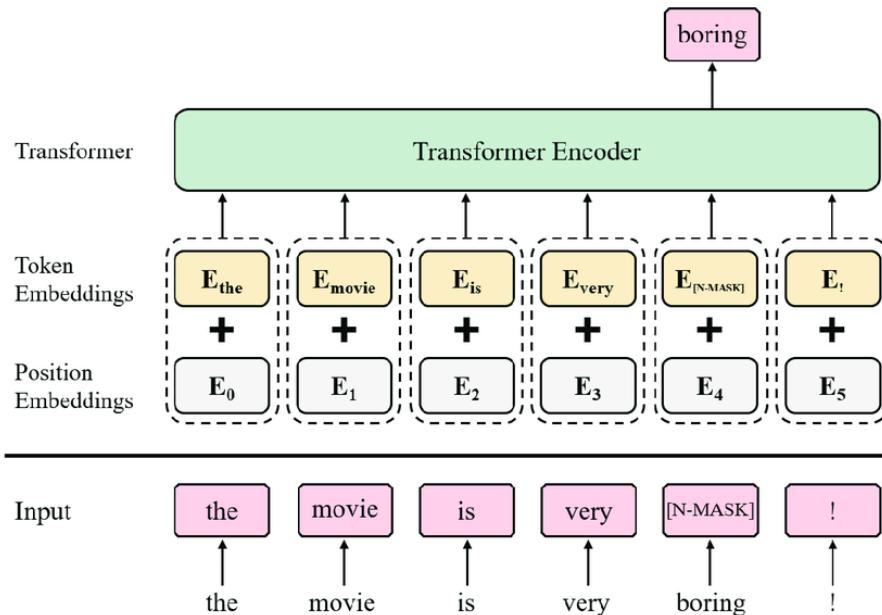
# BERT - Pretrained Transformer Encoder



**Pre-training:**

Task agnostic Masked Language Model.

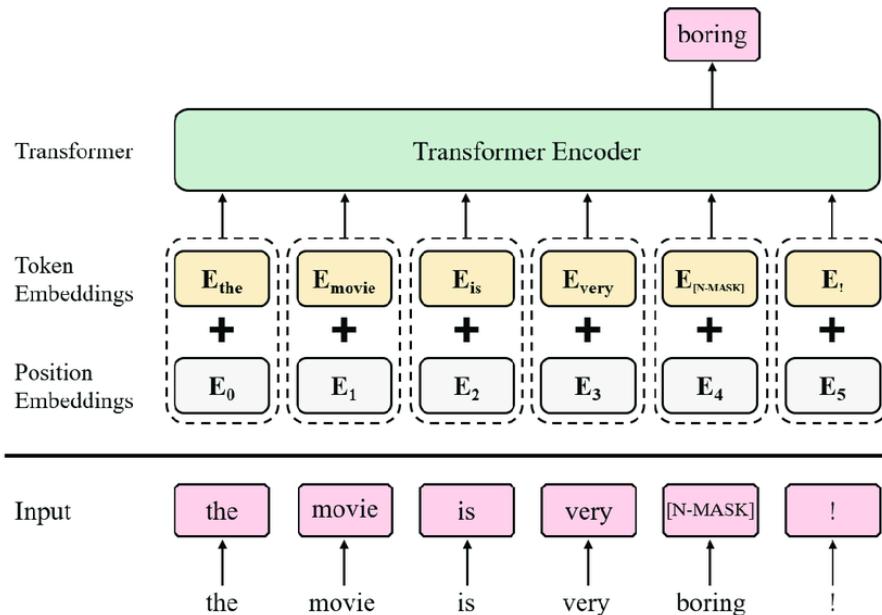
# BERT - Pretrained Transformer Encoder



**Pre-training:**

Task agnostic Masked Language Model.

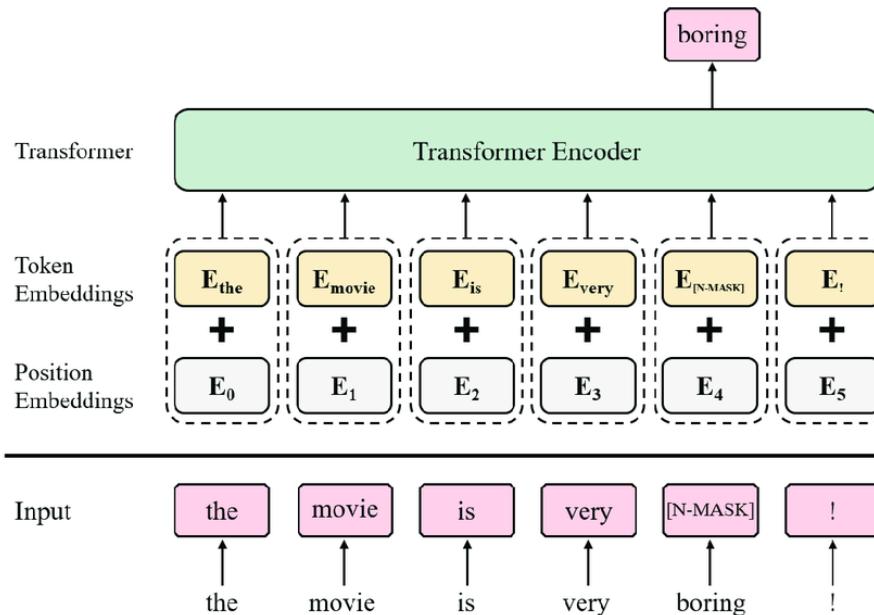
# BERT - Pretrained Transformer Encoder



**Pre-training:**

Task agnostic Masked Language Model.

# BERT - Pretrained Transformer Encoder

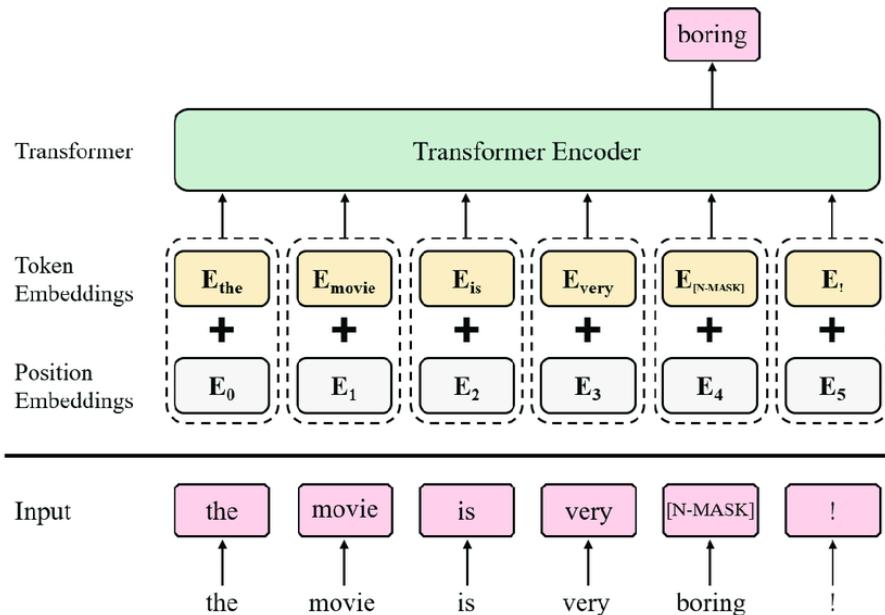


**Pre-training:**

Task agnostic Masked Language Model.

**SE usage:** Better **Suitable** for **Understanding Code**.

# BERT - Pretrained Transformer Encoder

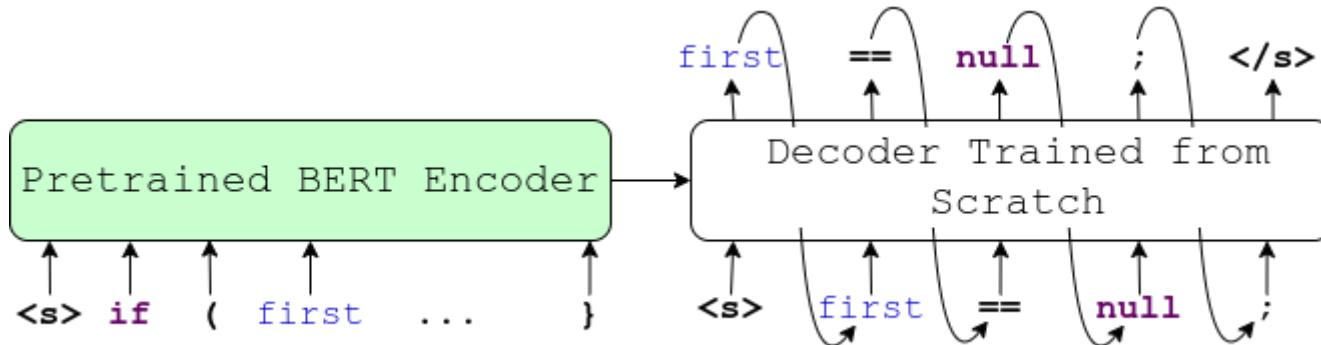


## Pre-training:

Task agnostic Masked Language Model.

1. CuBERT - Kanade *et. al.* 2020.
2. CodeBERT - Feng *et. al.* 2020.
3. GraphCodeBERT - Guo *et. al.* 2021

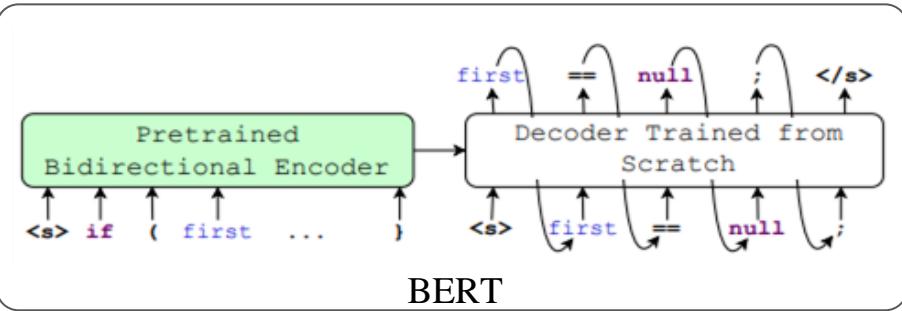
**SE usage:** Better **Suitable** for **Understanding Code**.



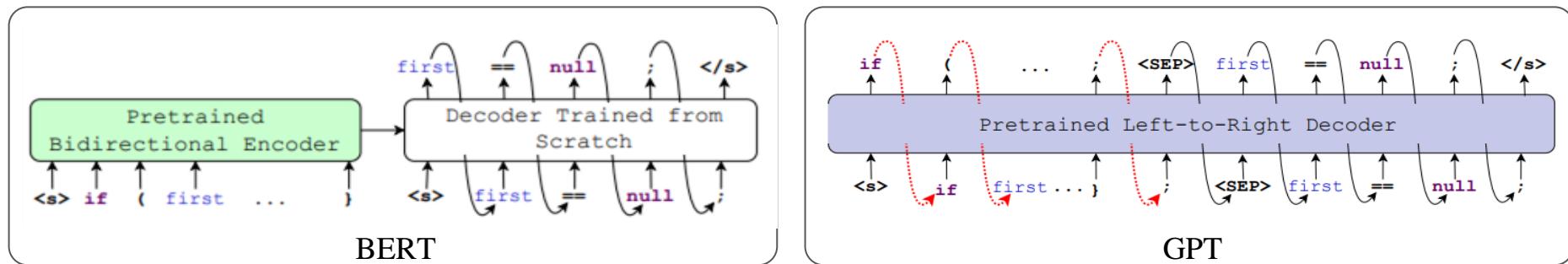
Knowledge about **generation** is **not** embedded in Decoder.

# CodeBERT vs. CodeGPT vs. PLBART

# CodeBERT vs. CodeGPT vs. PLBART



# CodeBERT vs. CodeGPT vs. PLBART



# CodeBERT vs. CodeGPT vs. PLBART

