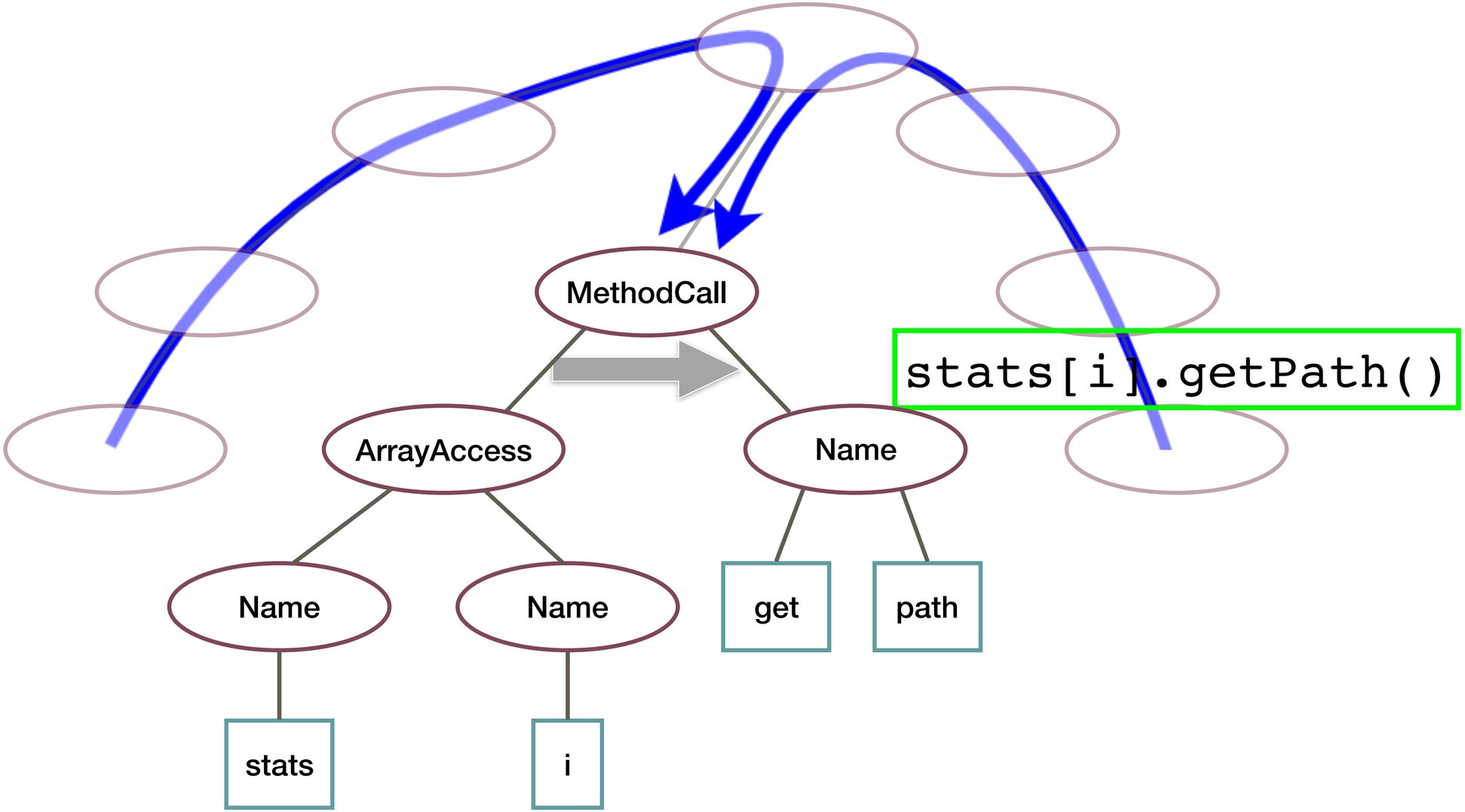
Any-Code Completion

```
public static Path[] stat2Paths(FileStatus[] stats) {
    if (stats == null)
        return null;
    Path[] ret = new Path[stats.length];
    for (int i = 0; i < stats.length; ++i){
        ret[i] =
    }
    return ret;
}
```

Generated: (Java)

`stats[i].getPath()`	(25.2%)
`new Path(stats[i])`	(3.3%)
`new Path(stats[i], charset)`	(2.5%)

Overview: a ***Structural*** Language Model



Any Code Gen

STRUCTURAL LANGUAGE MODELS OF CODE

Source: soon Paper

TO APPEAR IN ICML'2020

EXAMPLES: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

```java
public static Path[] stat2Paths(FileStatus[] stats) {
    if (??) return null;
    Path[] ret = new Path[??];
    for (int i = 0; i < stats.length; ++i) {
        ret[i] = stats[i].getPath();
    }
    return ret;
}
```

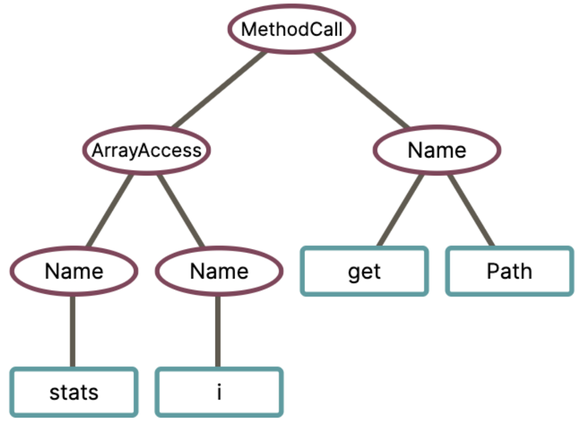
JAVA

- **stats[i].getPath()** 12.56%
- Path(stats[i]) 4.54%
- stat(stats[i], ret) 1.35%
- new Path(stats[i], charset) 1.15%
- statPath(stats[i]) 0.87%

Replace a code expression with "??". Then, hover over the "??" or press the green button.

AST

stats[i].getPath()



Tip: the tree is zoomable and movable.

http://AnyCodeGen.org

Structural Language Models of Code

ICML'2020

Uri Alon
Technion

Roy Sadaka
Technion

Omer Levy
Tel-Aviv University
Facebook AI Research

Eran Yahav
Technion

Language modeling of code

- Code completion
- Validate existing code, detect unlikely code.

```
public static Path[] stat2Paths(FileStatus[] stats) {
    if (stats == null)
        return null;
    Path[] ret = new Path[stats.size()];
    for (int i = 0; i < stats.length; ++i){
        ret[i] = stats[i].getPath();
    }
    return ret;
}
```

Key Idea #1: predict a missing subtree

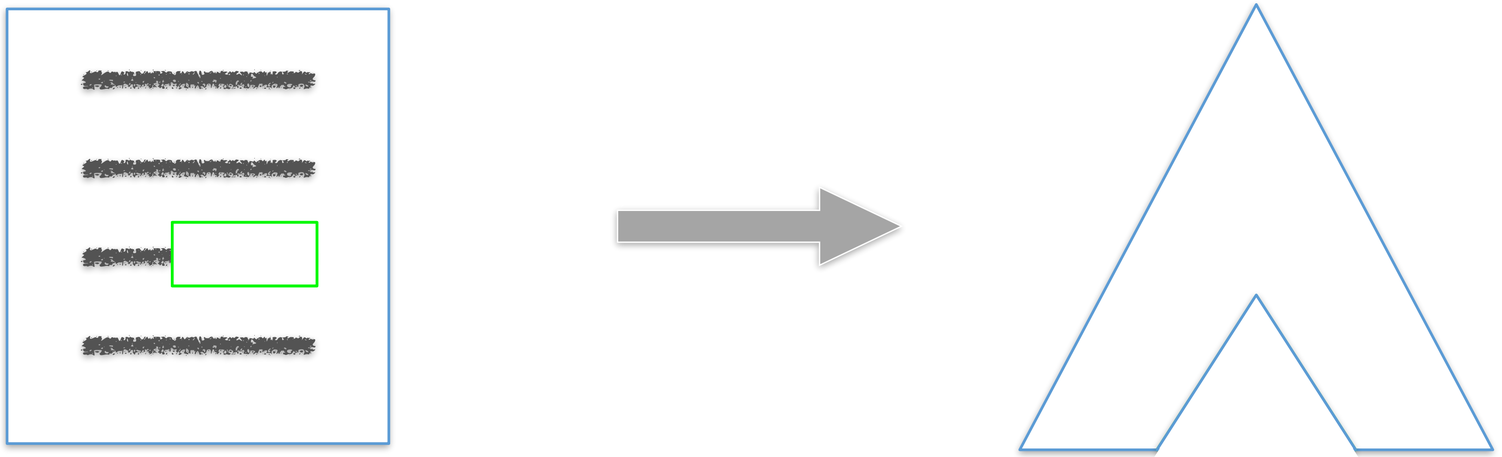
Instead of representing the task as:

"predict a missing **sentence** in a **text**"

Represent the task as:

"predict a missing **subtree** in a **tree**".

Learn **syntactic** patterns, instead of **sequential** patterns



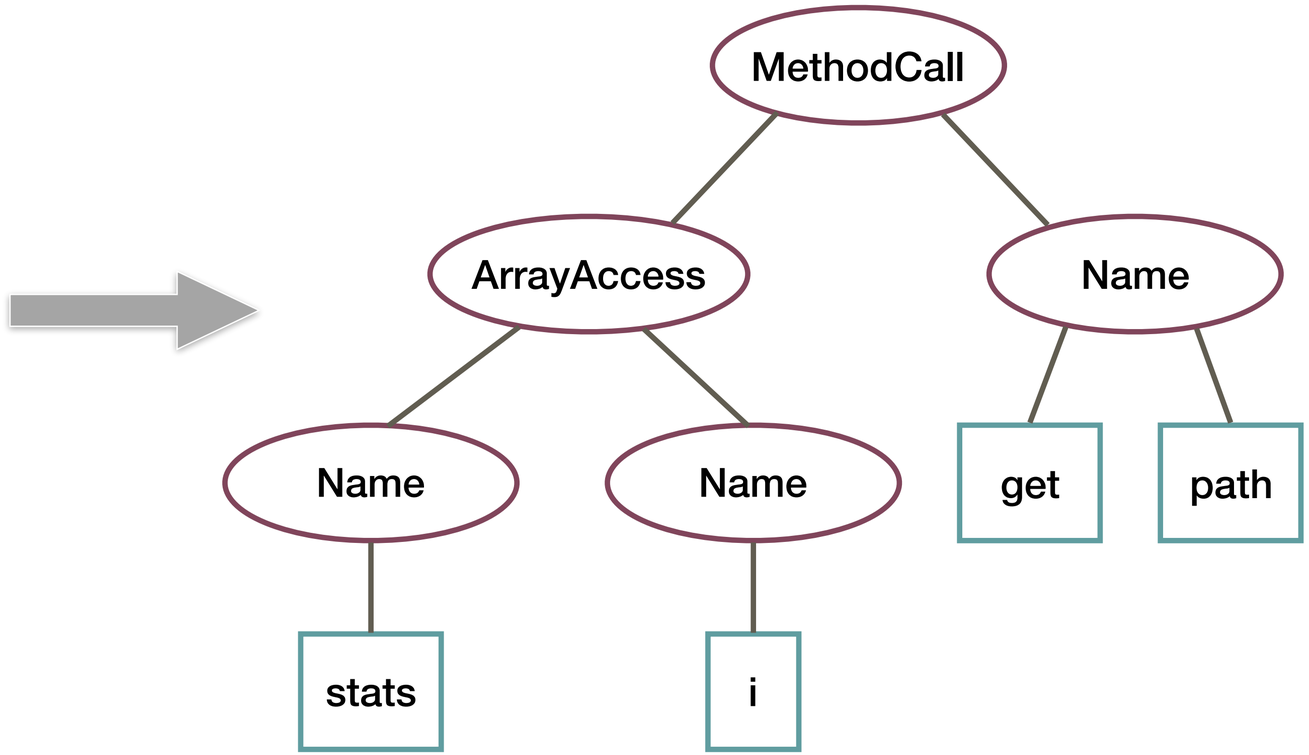
Abstract Syntax Tree

Any valid code snippet can be parsed into an Abstract Syntax Tree (AST).

The AST is composed of nodes

and user-defined values in its leaves.

```
stats[i].getPath()
```



Key Idea #2: a structural language model (SLM)

In a **natural-language model**:

$$Pr(Y) = Pr(y_1, y_2, \ldots, y_n) = \prod_{t=1}^{n} Pr\left(y_t \mid y < t\right)$$

But how can we compute the probability of a ***tree***?

Key Idea #2: a structural language model (SLM)

Given a tree $\mathcal{A}$ (can be an arbitrary graph)

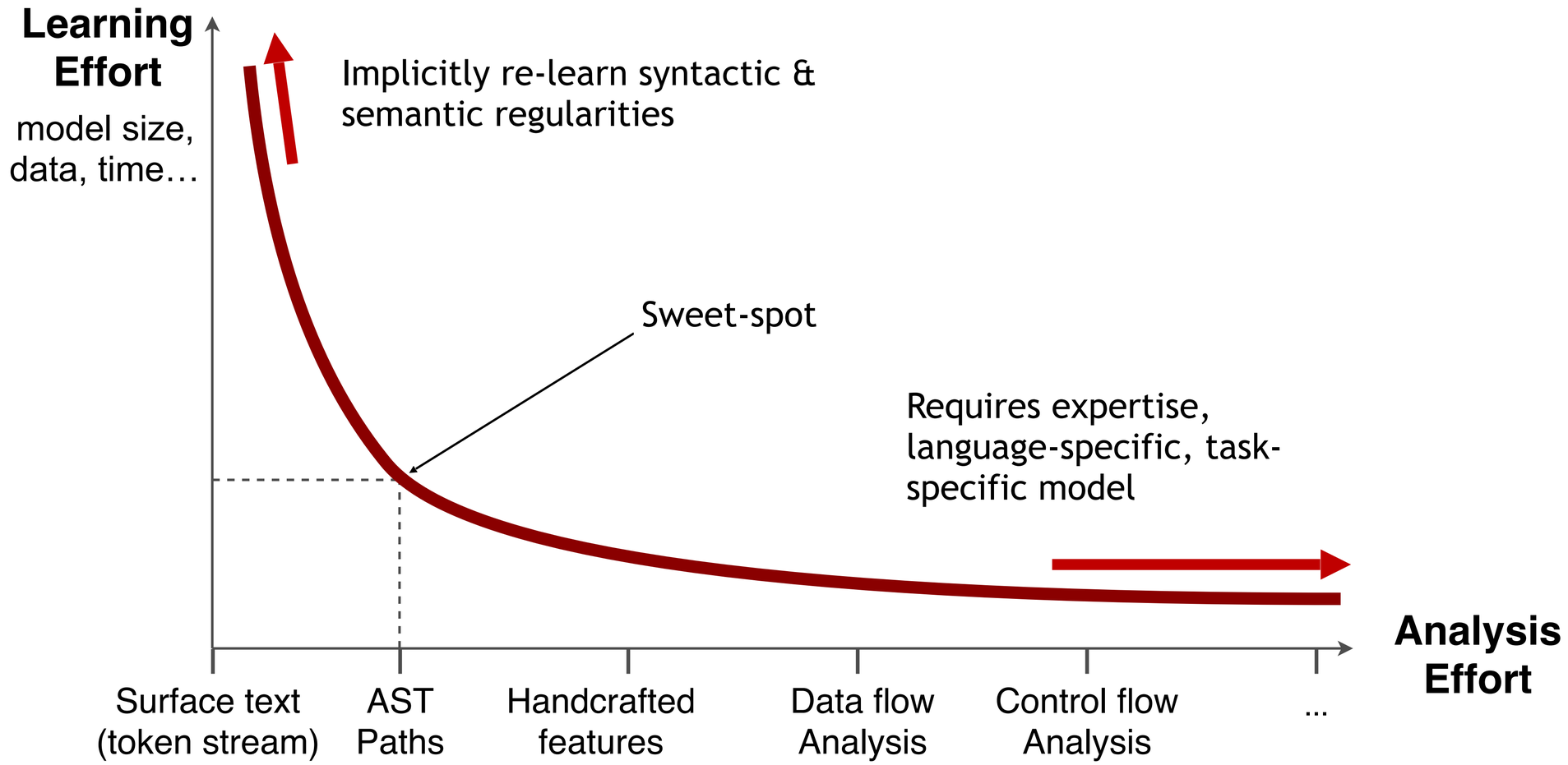
Induce an ordering over its nodes: $a_0, a_1, \ldots, a_n \in \mathcal{A}$ (in practice: DFS)

A **structural language model (SLM)** computes the probability of the tree $\mathcal{A}$:

$$Pr(\mathcal{A}) = \prod_{t=0}^{n} Pr\left(a_t \mid a_{<t}\right)$$

But, how can we represent the partial tree $a_{<t}$ when computing $Pr\left(a_t \mid a_{<t}\right)$?

The fundamental tradeoff in code representation



["A General Path-based Representation …", PLDI'2018]

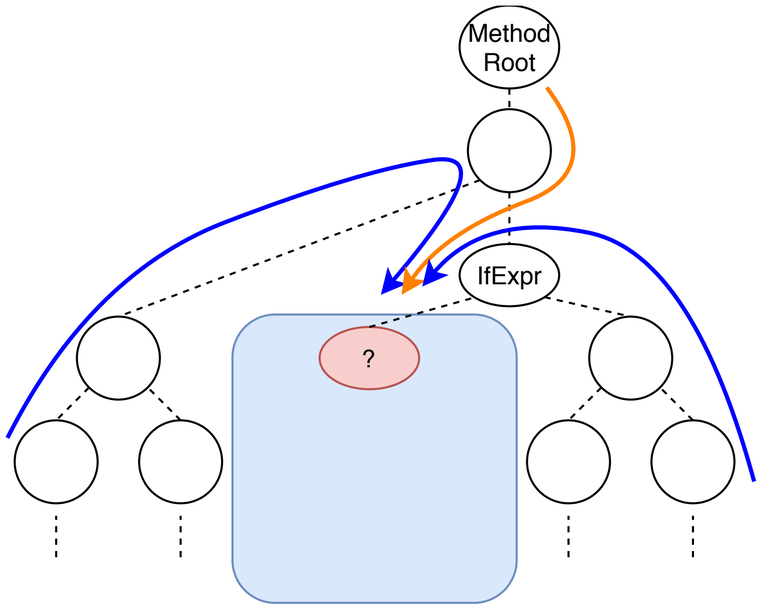
["code2vec", POPL'2019]

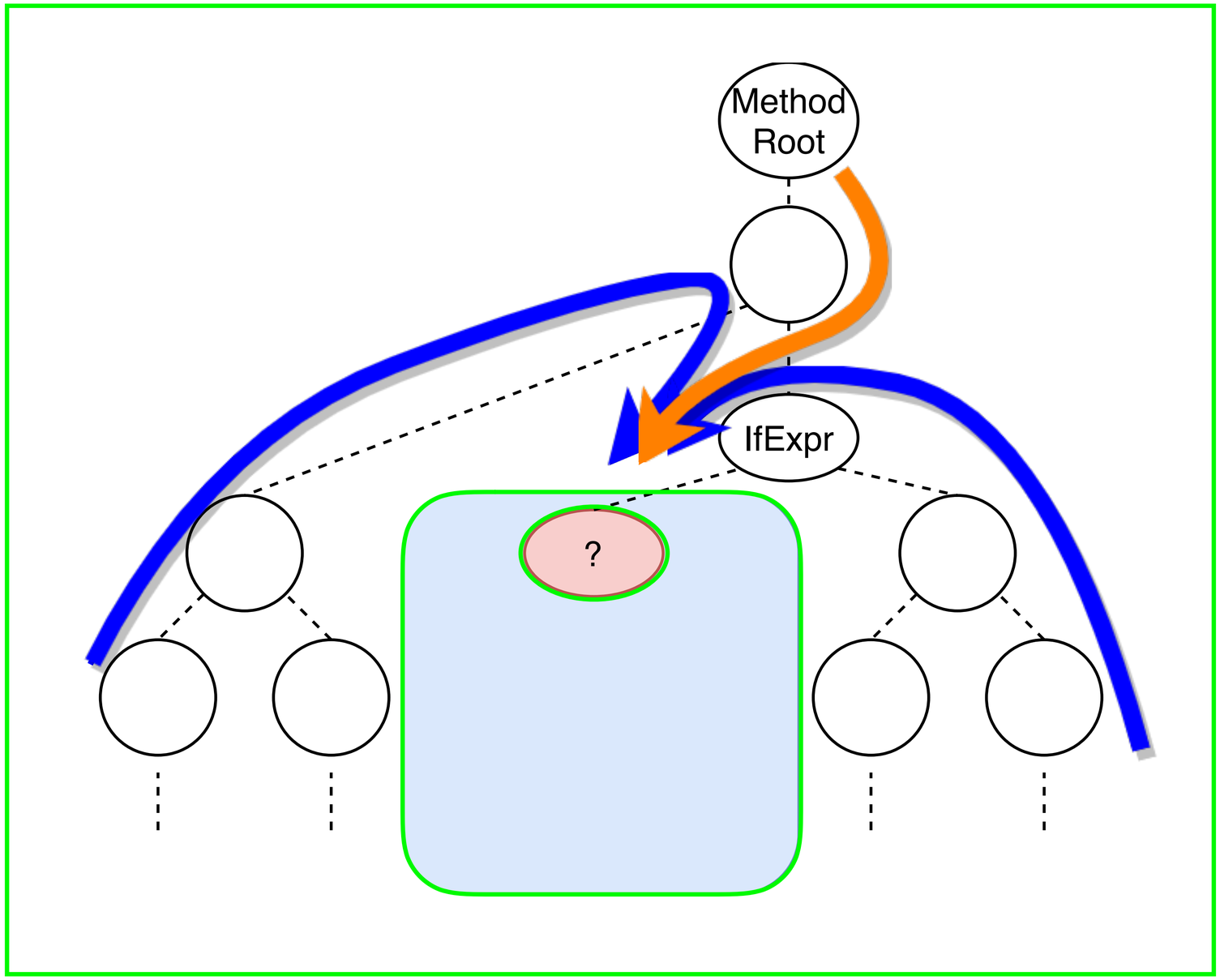
Key Idea #3: a partial tree as AST paths

We compute the probability of a node $Pr\left(a_t \mid a_{<t}\right)$

by considering the paths in the Abstract Syntax Tree (AST)

from all leaves into a_t .




Method
Root
IfExpr
?

AST Paths

AST Paths are simple paths over nodes in the AST.

In previous works, we used AST paths to **read** code.

In this work, we **generate** code by predicting the next node in a set of AST paths.

Code summarization in Java:

```
public boolean ___ (Set<String> set,
                    String value) {
  for (String entry : set) {
    if (entry.equalsIgnoreCase(value))
      return true;
  }
  return false;
}
```

① ② ③

contains① ignore② case③

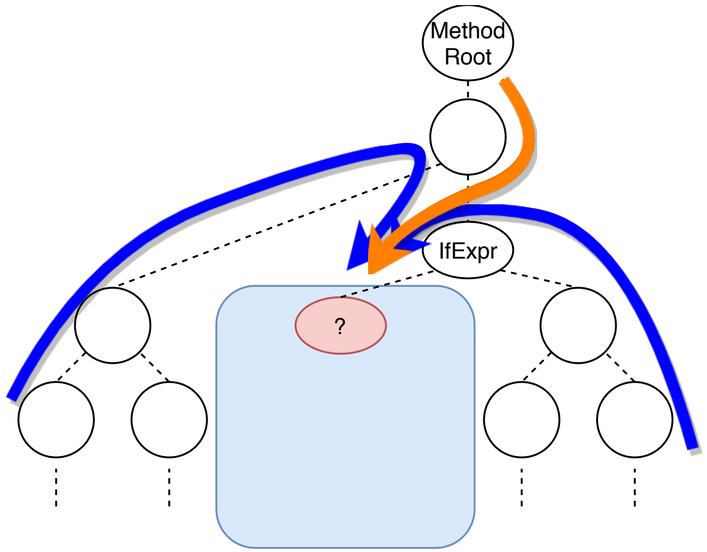
Code captioning in C#:

```
void Main() {
    string text = File.ReadAllText(@"T:\File1.txt");
    int num = 0;

    text = (Regex).Replace(text, "map", delegate (Match m) {
        return "map" + num++;
    }));
    File.WriteAllText(@"T:\File1.txt", text);
}
```

① ② ③

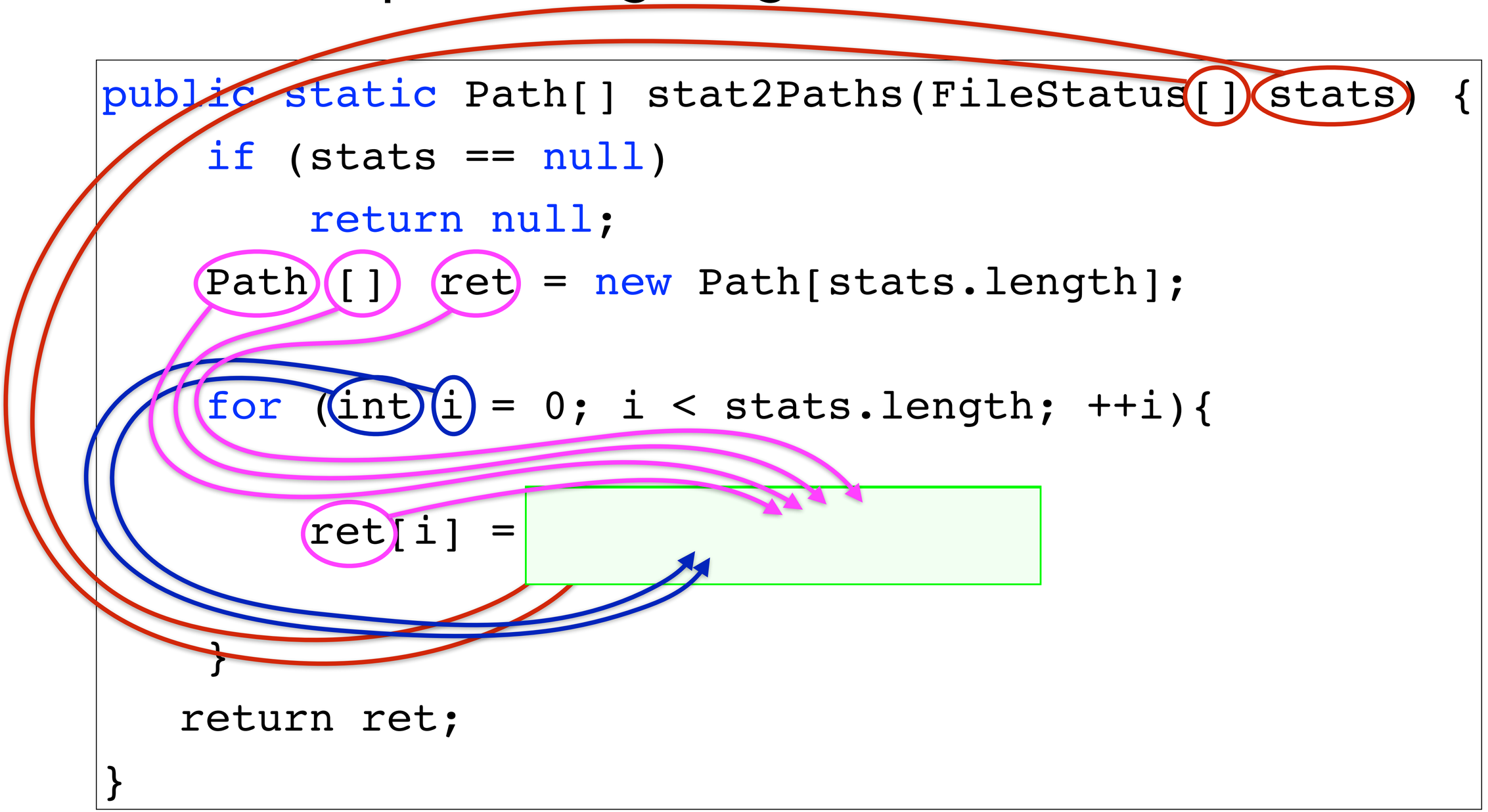
replace① a string② in a text③ file③



["code2seq", ICLR'2019]

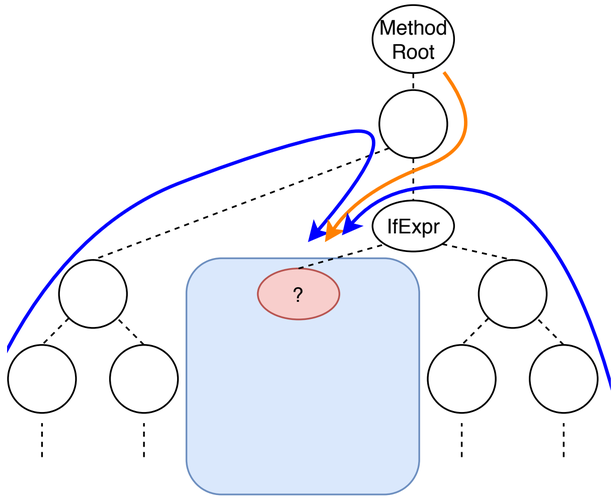
SLM, this work

AST Paths capture long-range interactions

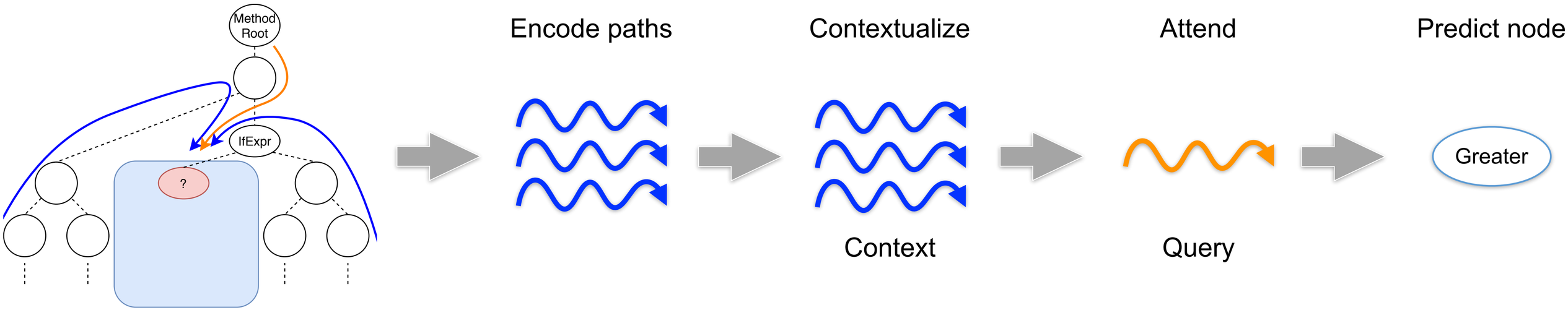


Model

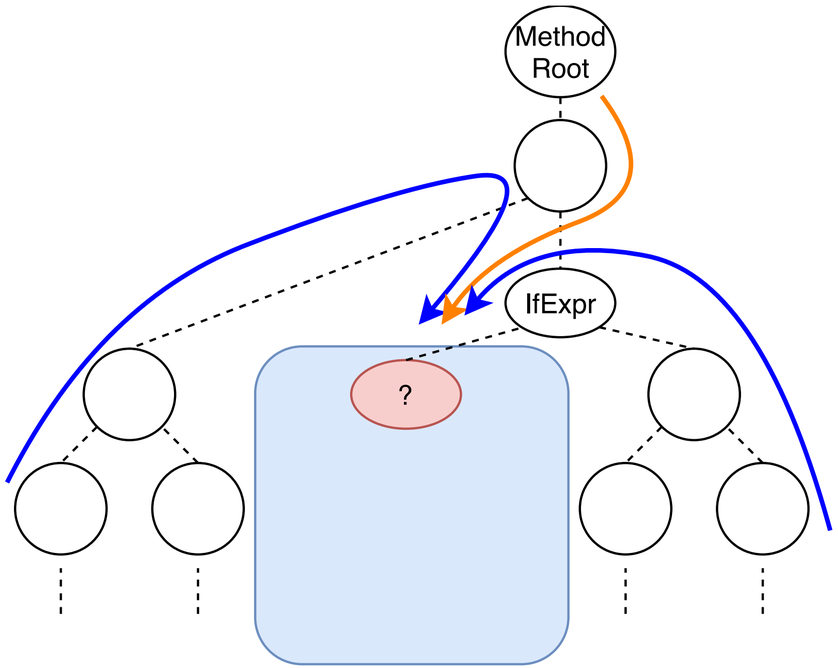
- Any sequential encoder to encode each arbitrary-length **path** into a fixed-length vector separately
  (e.g., LSTM, transformer encoder)
- Any contextualizer to let all **paths** interact
  (e.g., transformer encoder)
- Attend to the **contextualized paths** using the **root path** as the query



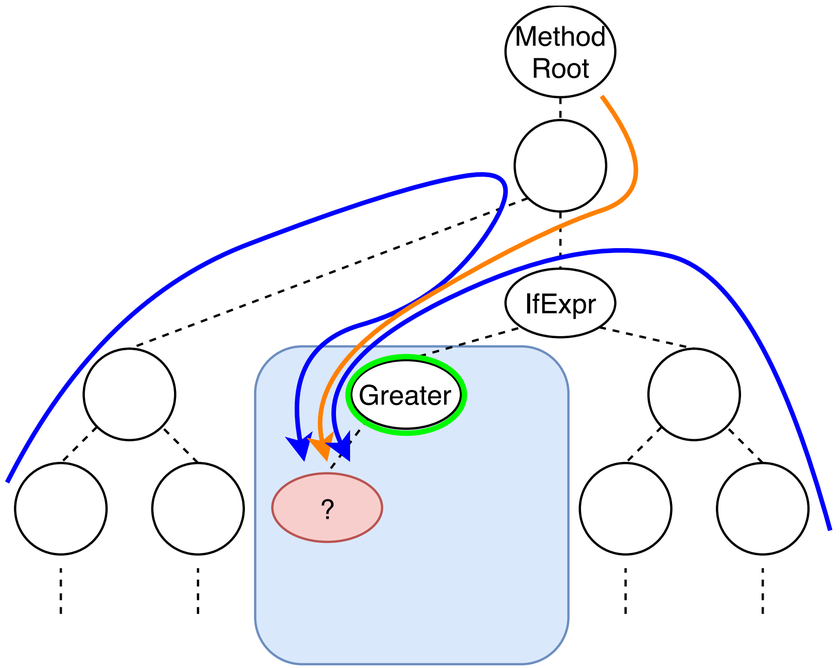
Model


Method
Root
IfExpr
?
Encode paths
Contextualize
Attend
Predict node
Greater
Context
Query

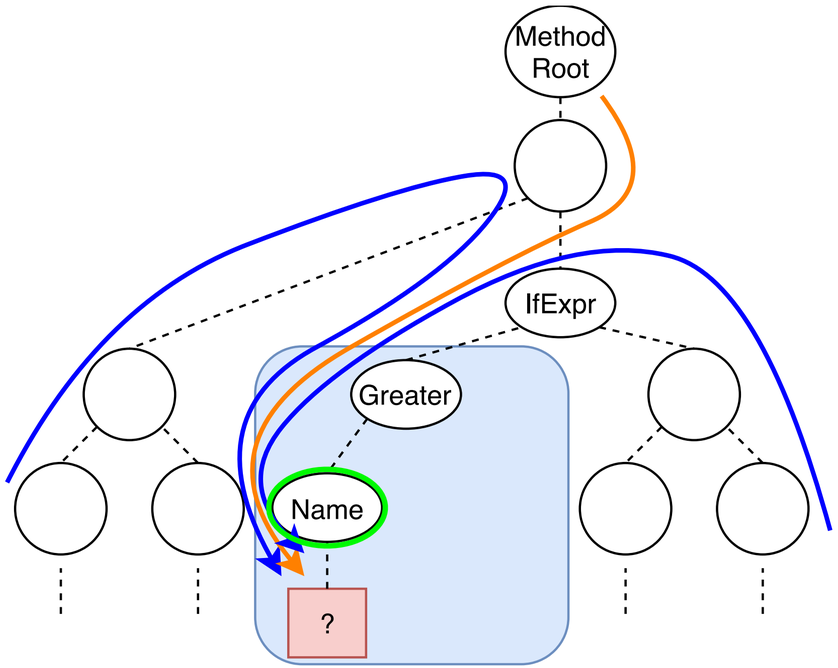
Generate the Tree of: `x > 1`



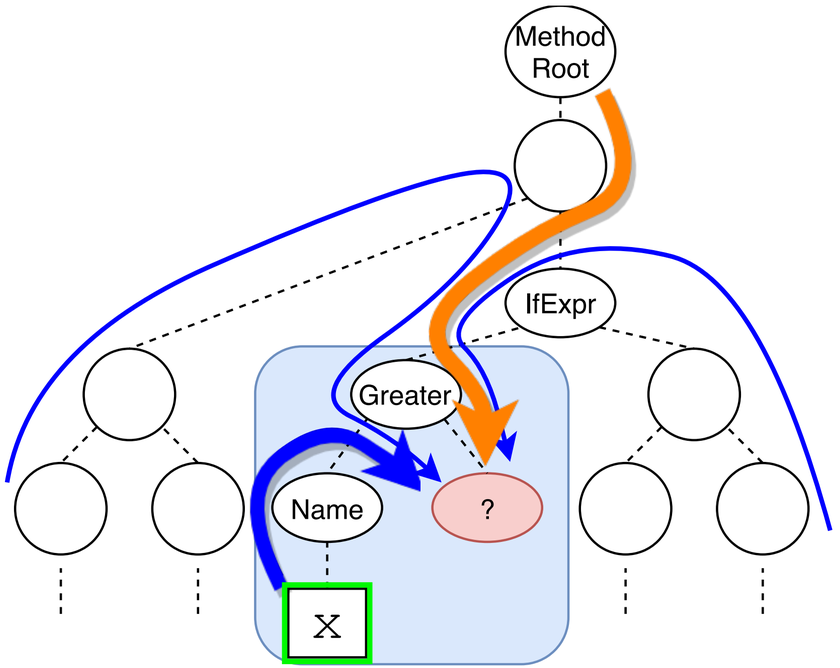
Generate the Tree of: `x > 1`



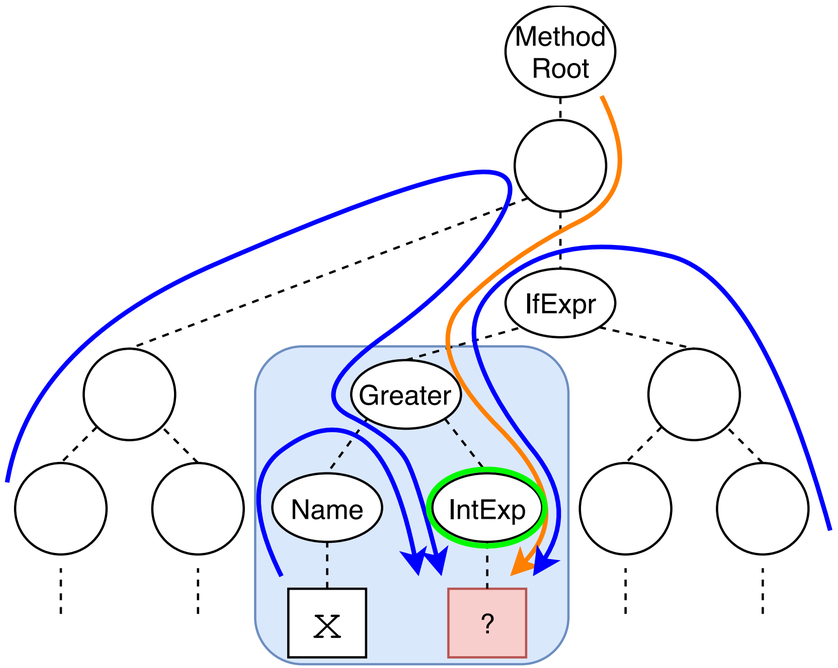
Generate the Tree of: `x > 1`



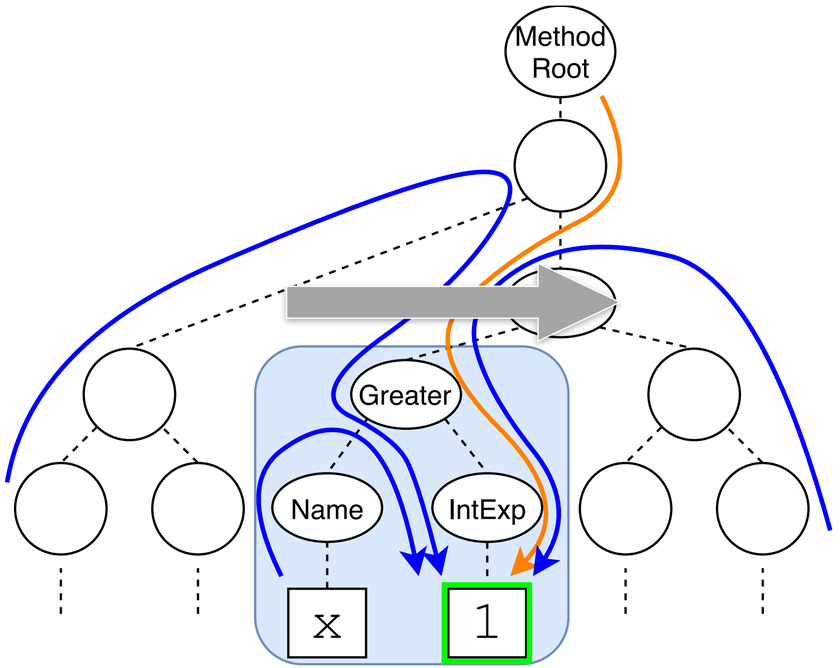
Generate the Tree of: `x > 1`



Generate the Tree of: `x > 1`

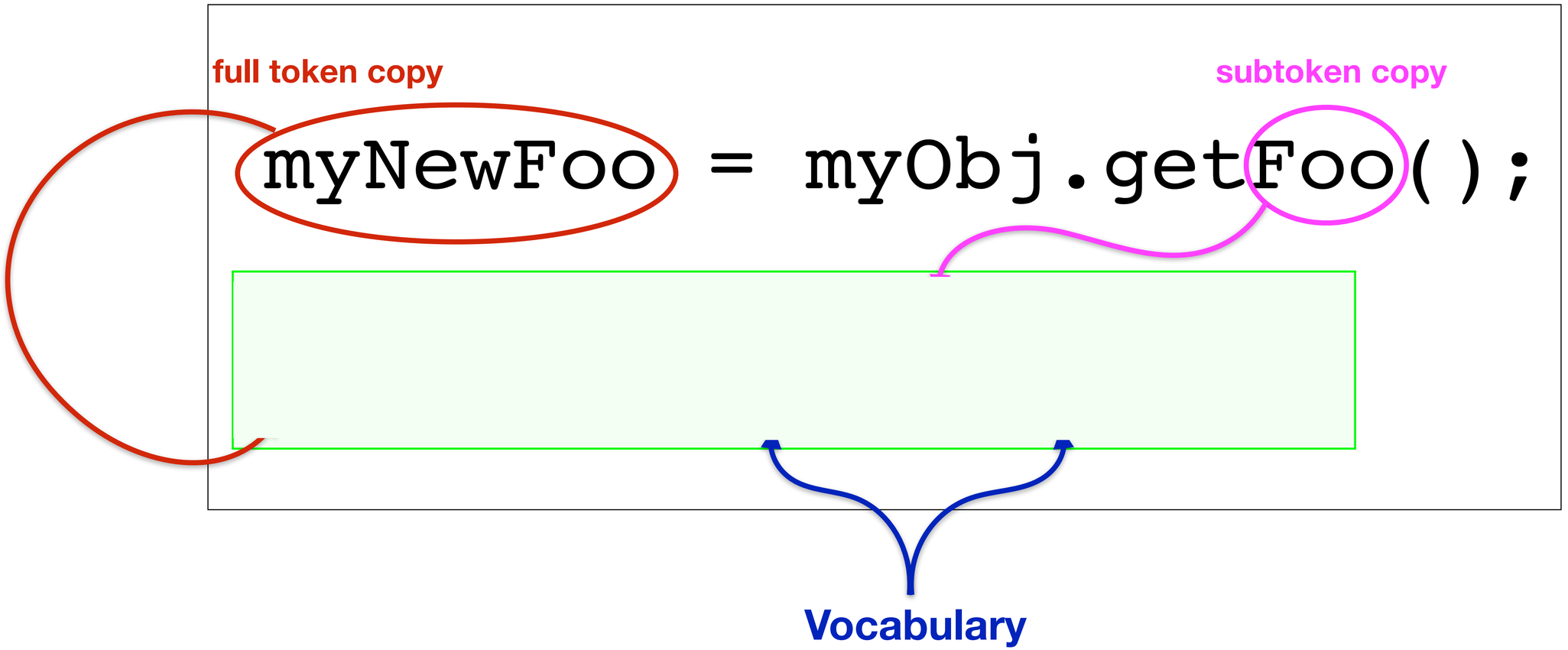


Generate the Tree of: `x > 1`




```
x > 1
```

Copy Mechanism



Example - Java

```
public static Path[] stat2Paths(FileStatus[] stats) {
    if (stats == null)
        return null;
    Path[] ret = new Path[stats.length];
    for (int i = 0; i < stats.length; ++i){
        ret[i] =
    }
    return ret;
}
```

Generated: (Java)

`stats[i].getPath()`	(25.2%)
`new Path(stats[i])`	(3.3%)
`new Path(stats[i], charset)`	(2.5%)

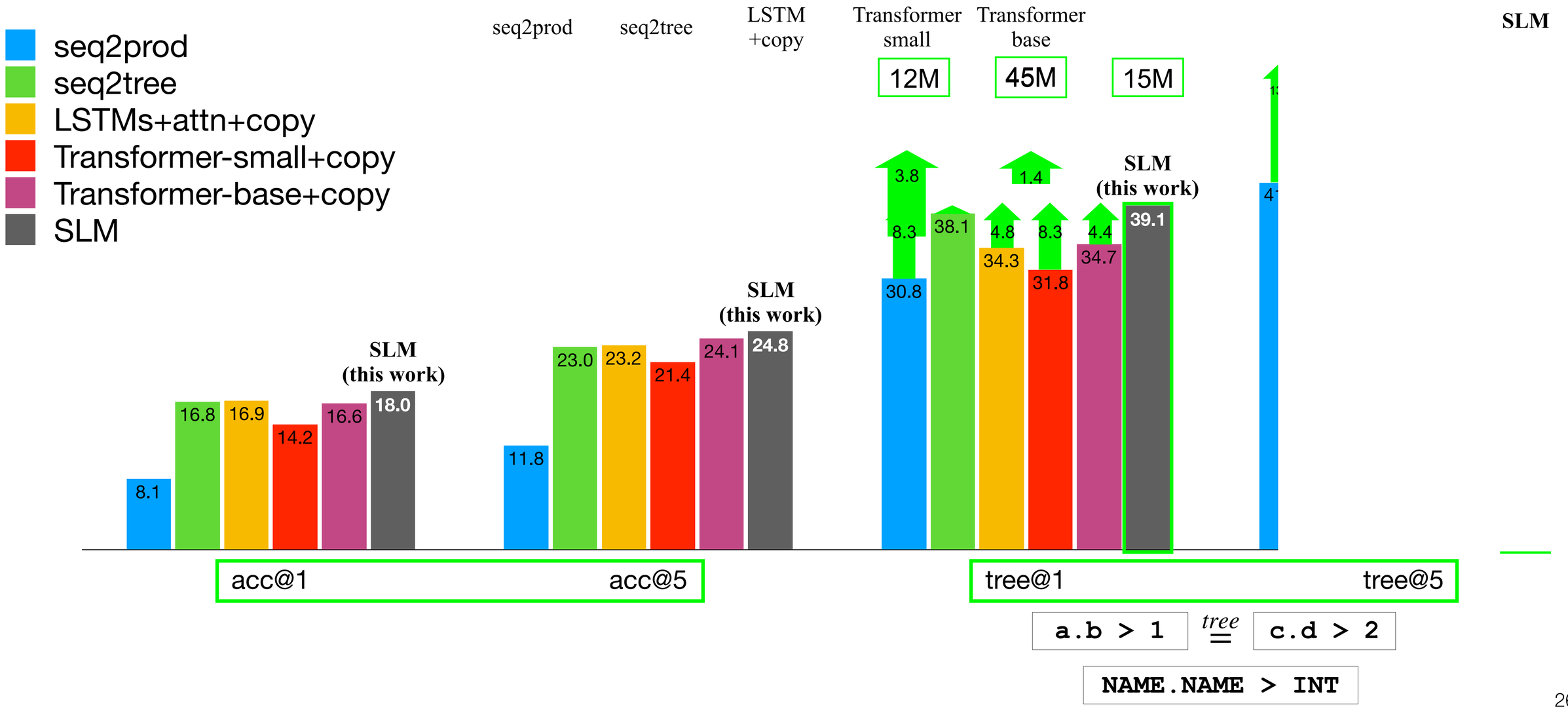
Example - C#

```
public static string Camelize(this string input)
{
    var word = input.Pascalize();
    return word.Length > 0 ?                    .ToLower()
        + word.Substring(1) : word;
}
```

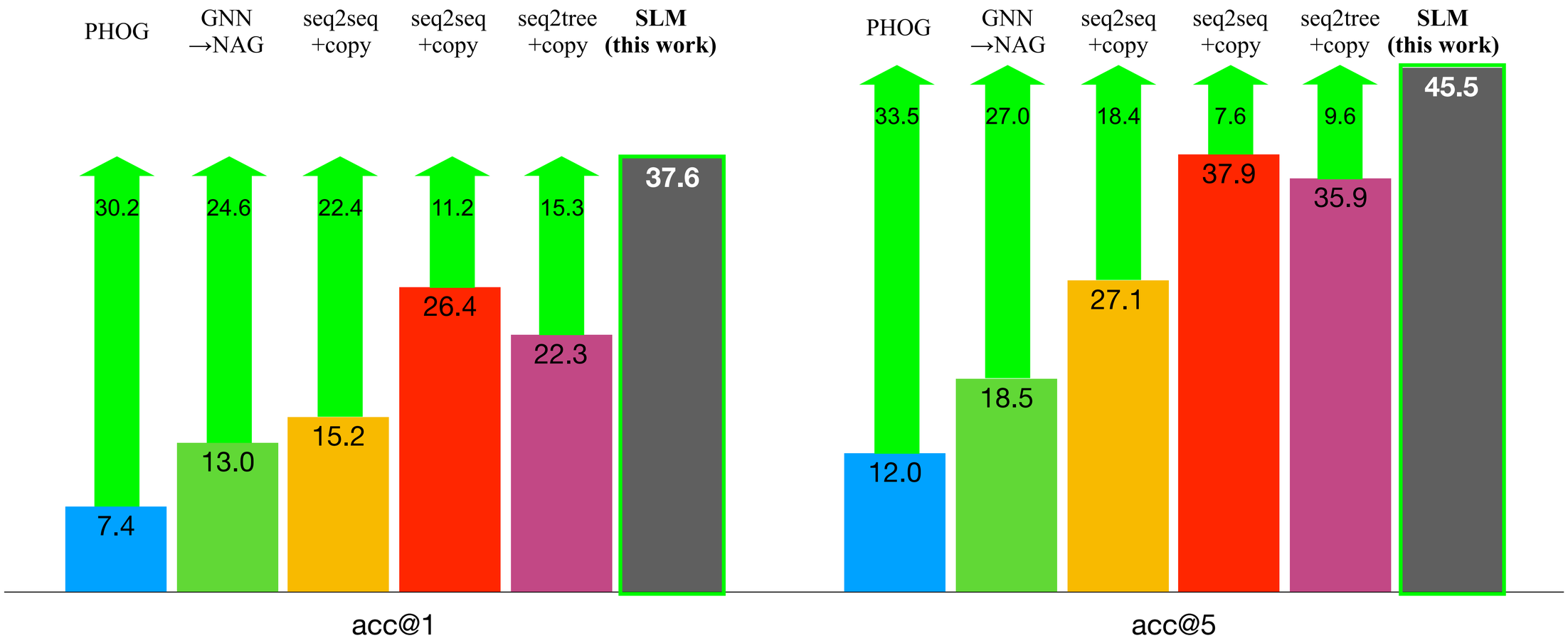
Generated: (C#)

`word.Substring(0, 1)`	(14.1%)
`word.trim()`	(8.2%)
`word.Substring(1)`	(5.8%)

Java Results (trained on 1.3M examples)



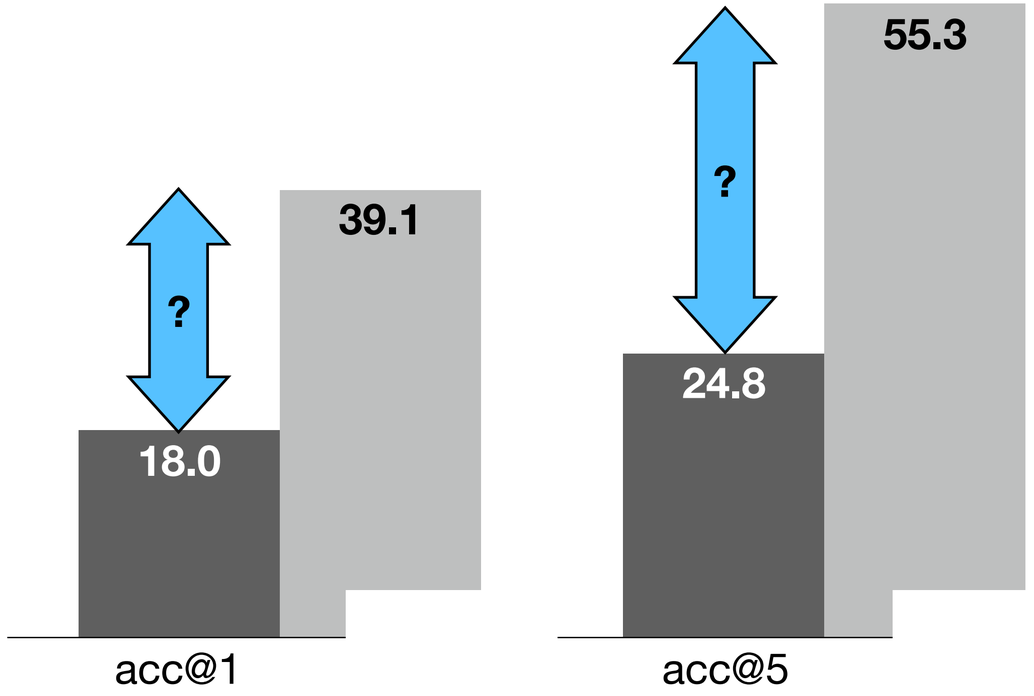
C# Results


PHOG
GNN →NAG
seq2seq +copy
seq2seq +copy
seq2tree +copy
SLM (this work)
30.2
24.6
22.4
11.2
15.3
37.6
7.4
13.0
15.2
26.4
22.3
acc@1
PHOG
GNN →NAG
seq2seq +copy
seq2seq +copy
seq2tree +copy
SLM (this work)
33.5
27.0
18.4
7.6
9.6
45.5
12.0
18.5
27.1
37.9
35.9
acc@5

Error Analysis

What kind of mistakes are responsible for the gap between **acc@k** and **tree@k** ?

SLM (this work):

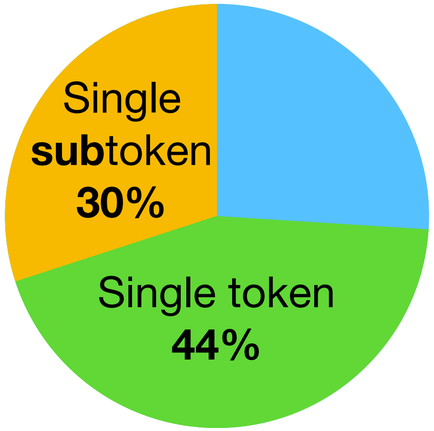


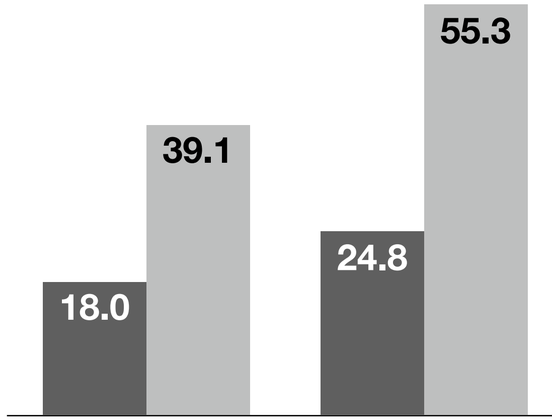
Error Analysis

What kind of mistakes are responsible for the gap between **acc@k** and **tree@k** ?

74%: Single-token mismatch

30%: Single-**sub**token mismatch





Error Analysis

```
public float getProgress() {
    this.readLock.lock();
    try {
        if (this.currentAttempt != null) {
            return this.currentAttempt.getProgress();
        }
        return 0;
    } finally {
        this.readLock.unlock();
    }
}
```

Generated:		Exact-match	Tree-match	Compiles
`this.currentAttempt.getCount()`	(31.3%)	✘	✔	✘
`-1`	(30.6%)	✘	✘	✔
`this.currentAttempt.get()`	(1.5%)	✘	✔	✘
`this.currentAttempt.getTime()`	(1.2%)	✘	✔	✘
`this.currentAttempt.getProgress()`	(0.9%)	✔	✔	✔

Error Analysis

```
public float getProgress() {
    this.readLock.lock();
    try {
        if (this.currentAttempt != null) {
            return this.currentAttempt.getProgress();
        }
        return 0;
    } finally {
        this.readLock.unlock();
    }
}
```

Generated:		Exact-match	Tree-match	Compiles
`this.currentAttempt.getCount()`	(31.3%)	✘	✔	✘
`-1`	(30.6%)	✘	✘	✔
`this.currentAttempt.get()`	(1.5%)	✘	✔	✘
`this.currentAttempt.getTime()`	(1.2%)	✘	✔	✘
`this.currentAttempt.getProgress()`	(0.9%)	✔	✔	✔

Any Code Gen

STRUCTURAL LANGUAGE MODELS OF CODE

Source: soon Paper

TO APPEAR IN ICML'2020

EXAMPLES: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

JAVA

```
1  public static Path[] stat2Paths(FileStatus[] stats) {
2      if (??) return null;
3      Path[] ret = new Path[??];
4      for (int i = 0; i < stats.length; ++i) {
5          ret[i] = ??
6      }
7      return ret;
8  }
```

Click here to predict

AST

Replace a code expression with "??". Then, hover over the "??" or press the green button.

http://AnyCodeGen.org

Any Code Gen

STRUCTURAL LANGUAGE MODELS OF CODE

Source: soon Paper

TO APPEAR IN ICML'2020

EXAMPLES: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

```java
public static Path[] stat2Paths(FileStatus[] stats) {
    if (??) return null;
    Path[] ret = new Path[??];
    for (int i = 0; i < stats.length; ++i) {
        ret[i] = stats[i].getPath();
    }
    return ret;
}
```

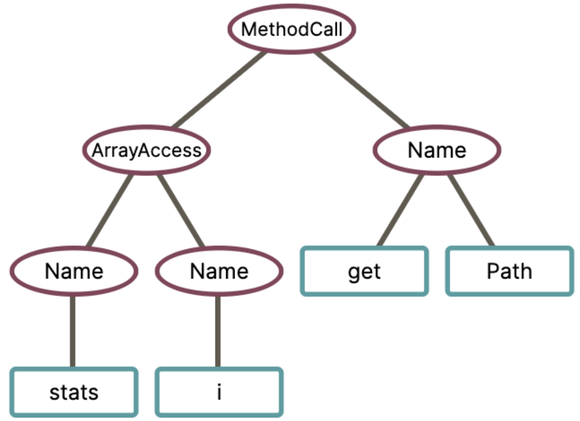
JAVA

- **stats[i].getPath()** 12.56%
- Path(stats[i]) 4.54%
- stat(stats[i], ret) 1.35%
- new Path(stats[i], charset) 1.15%
- statPath(stats[i]) 0.87%

Replace a code expression with "??". Then, hover over the "??" or press the green button.

AST

stats[i].getPath()



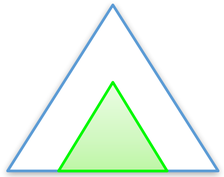
Tip: the tree is zoomable and movable.

http://AnyCodeGen.org

Structural Language Models of Code

Key points:

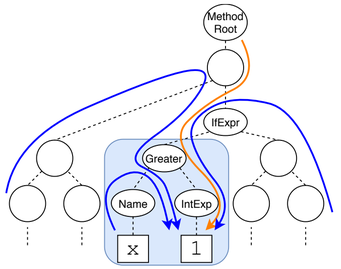
1. Predicting a missing **subtree** in a **tree**



2. A structural language model over ***trees***

$$Pr(\mathcal{A}) = \prod_{t=0}^{n} Pr\left(a_t \mid a_{<t}\right)$$

3. A partial AST as a set of paths



http://AnyCodeGen.org

urialon@cs.technion.ac.il