Learning (Un)Natural Run-Time Behavior from Source Code

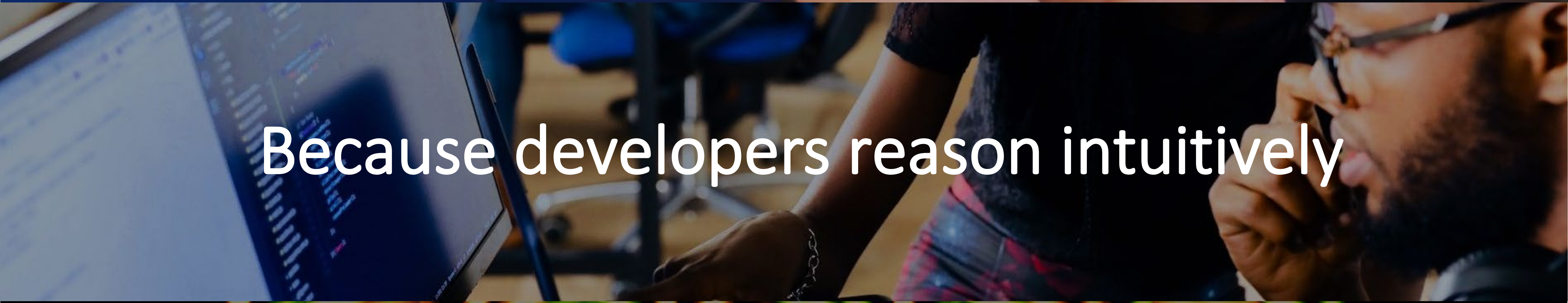
(or: What to Expect when you’re Executing)

Vincent Hellendoorn

CyLab Partner’s Conference

CMU, 2020

Security needs intuitive reasoning



Because developers reason intuitively

(and deep learning is the way to do it)

A tale of two snippets

```
public void setChild(int index, Node n) {
  this.children.add(index, n);
  n.parent=this;
}
```

```
public void setChild(int index, Node n) {
   if (index < 0)
     throw new IllegalArgumentException();
  if (this.children == null)
     this.children = new ArrayList<Node>();
  if (index > this.children.size())
     throw new IndexOutOfBoundsException();
  this.children.add(index, n);
  if (n.parent != null) {
     n.parent.removeChild(n);
     n.parent = this;
  }
  else
    n.parent = this;
}
```

What do we see?

Creating a Relationship

Node ⇒Tree?

```java
public void setChild(int index, Node n) {
  this.children.add(index, n);
  n.parent = this;
}
```

Must be ordered: List?

Singular, so 1-ary

What do we see?

```
index >= 0
```

```
public void setChild(int index, Node n) {
  this.children.add(index, n);
  n.parent = this;
}
```

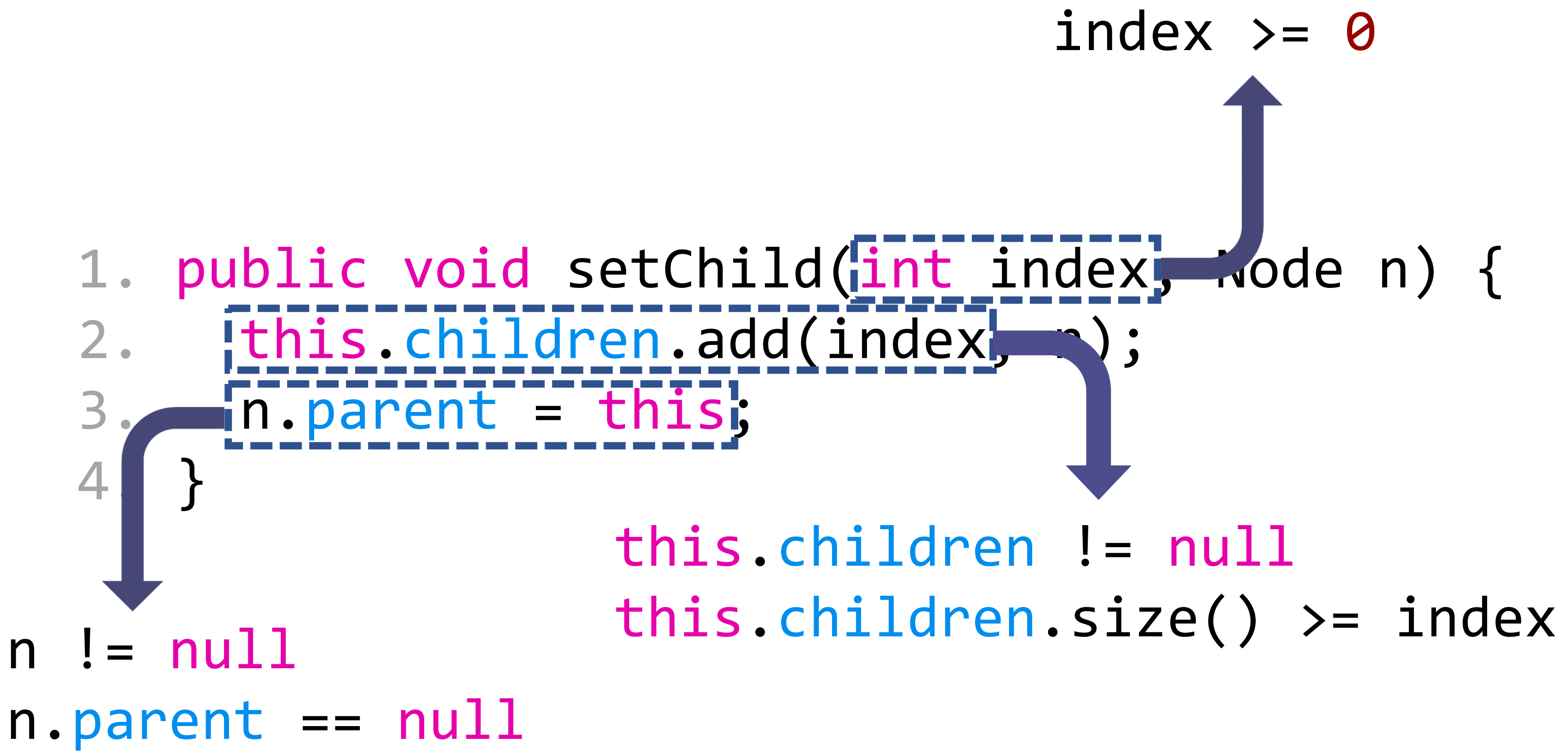
What do we see?

```
index >= 0
```

```java
public void setChild(int index, Node n) {
  this.children.add(index, n);
  n.parent = this;
}
```

```java
this.children != null
this.children.size() >= index
```

What do we see?



```
public void setChild(int index, Node n) {
  this.children.add(index, n);
  n.parent = this;
}
```

```
index >= 0
```

```
this.children != null
this.children.size() >= index
```

```
n != null
n.parent == null
```

```java
1. public void setChild(int index, Node n) {
     if (index < 0)
       throw new IllegalArgumentException();
     if (this.children == null)
       this.children = new ArrayList<Node>();
     if (index > this.children.size())
       throw new IndexOutOfBoundsException();
2.   this.children.add(index, n);
     if (n.parent != null) {
        n.parent.removeChild(n);
        n.parent = this;
     }
     else
3.     n.parent = this;
4. }
```

Developers reason intuitively

```
1. public void setChild(int index, Node n) {
     if (index < 0)
       throw new IllegalArgumentException();
     if (this.children == null)
       this.children = new ArrayList<Node>();
     if (index > this.children.size())
       throw new IndexOutOfBoundsException();
2.   this.children.add(index, n);
     if (n.parent != null) {
        n.parent.removeChild(n);
        n.parent = this;
     }
     else
3.     n.parent = this;
4. }
```

Invariants

```java
public void setChild(int index, Node n) {
  this.children.add(index, n);
    n.parent = this;
}
```

Invariants

Universal constraint

```java
public void setChild(int index, Node n) {
  if (index < 0)
    throw new IllegalArgumentException();

  this.children.add(index, n);

    n.parent = this;
}
```

Invariants

Programmatic constraints

```
1. public void setChild(int index, Node n) {

     if (this.children == null)
       this.children = new ArrayList<Node>();

2.   this.children.add(index, n);

3.     n.parent = this;
4. }
```

Invariants

Logical constraints

```
public void setChild(int index, Node n) {
    if (index > this.children.size())
      throw new IndexOutOfBoundsException();
    this.children.add(index, n);
      n.parent = this;
}
```

Invariants

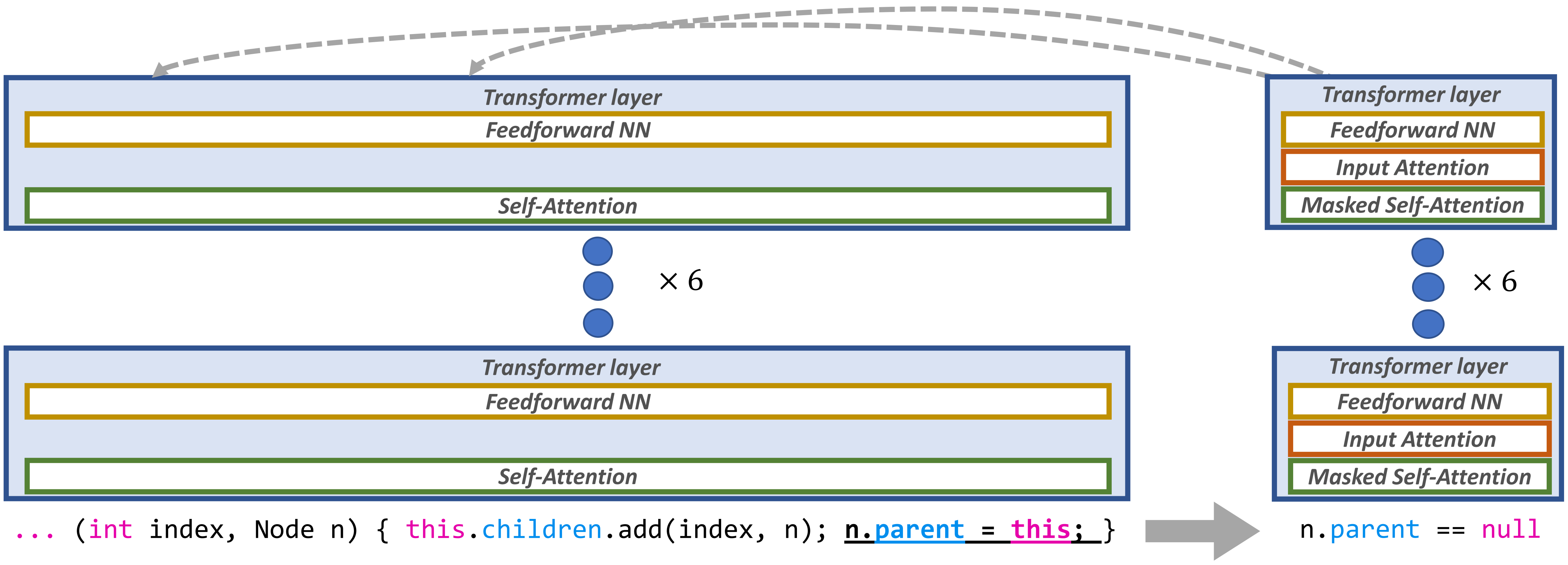
```
1. public void setChild(int index, Node n) {




2.   this.children.add(index, n);
     if (n.parent != null) {
        n.parent.removeChild(n);
        n.parent = this;
     }
     else
3.     n.parent = this;
4. }
```

Complex constraints

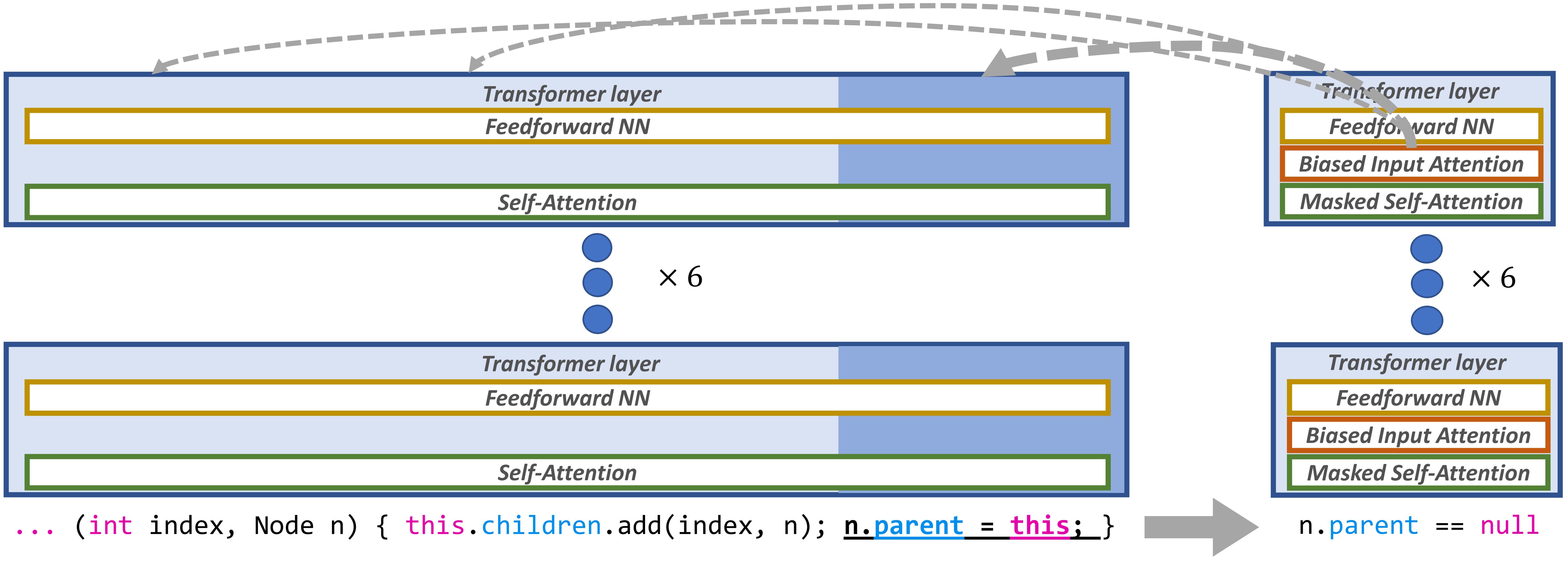
```
1. public void setChild(int index, Node n) {



2.    this.children.add(index, n);
      if (n.parent != null) {
         n.parent.removeChild(n);
         n.parent = this;
      }
      else
3.       n.parent = this;
4. }
```

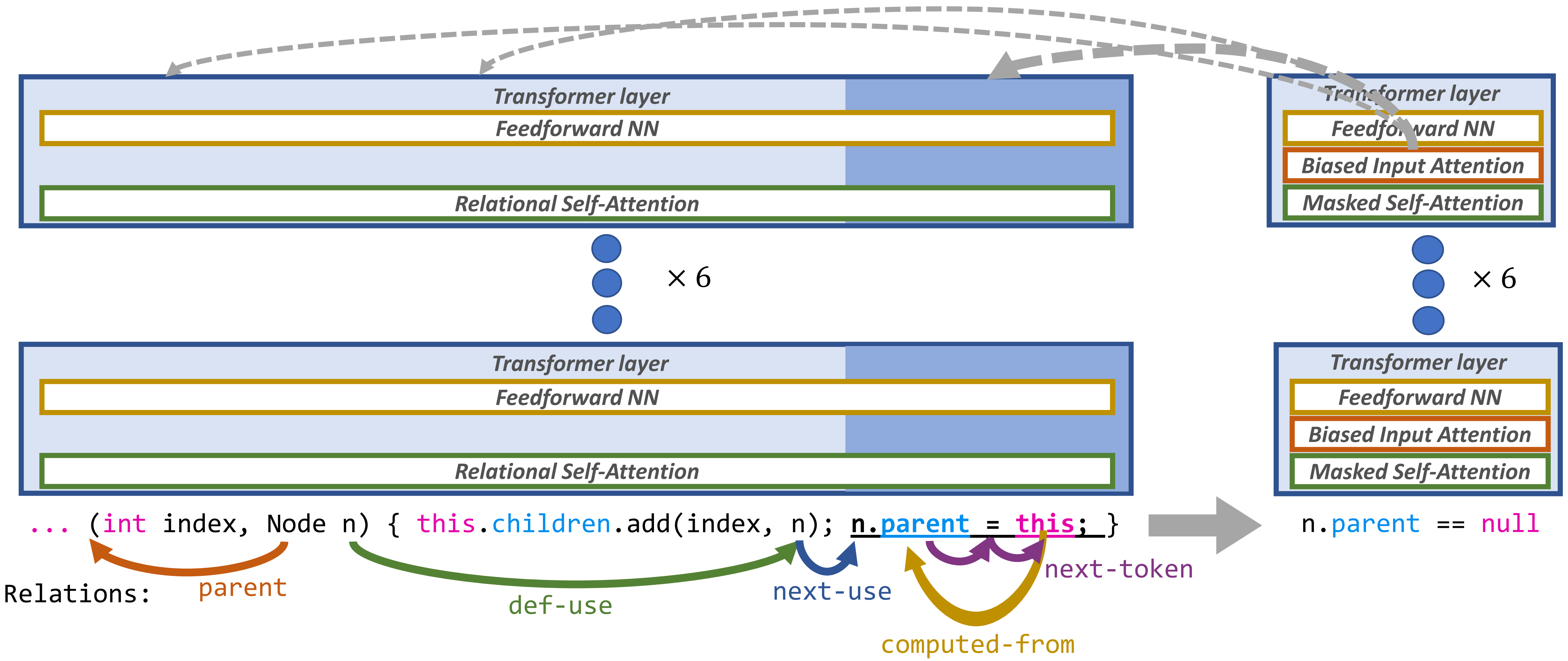
Complex constraints

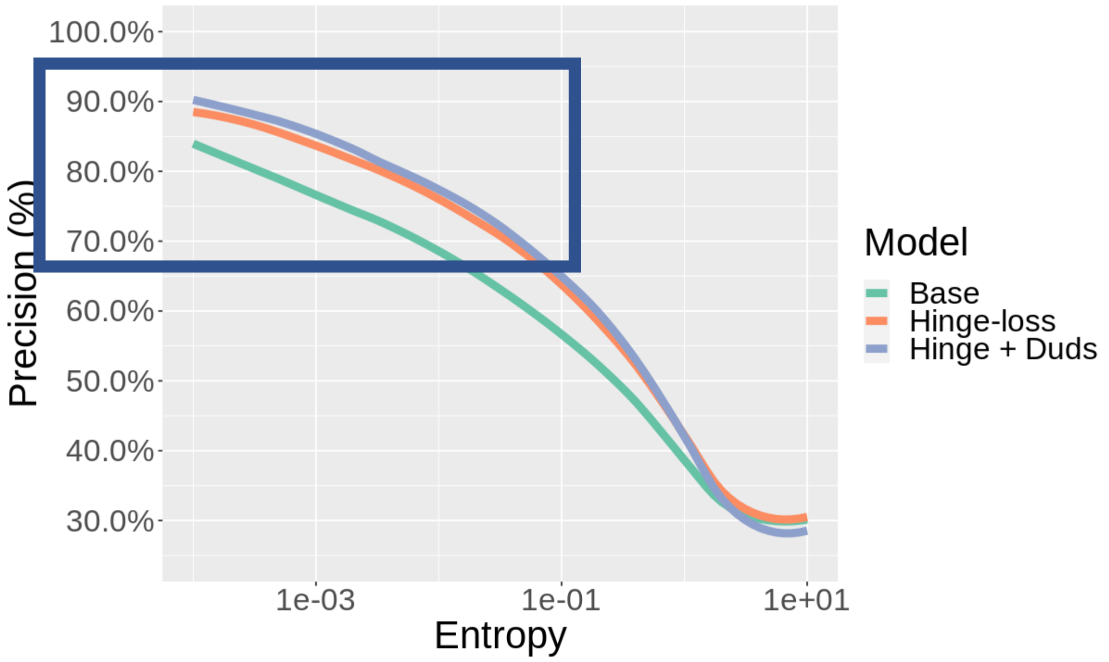
```
n.parent == null
```



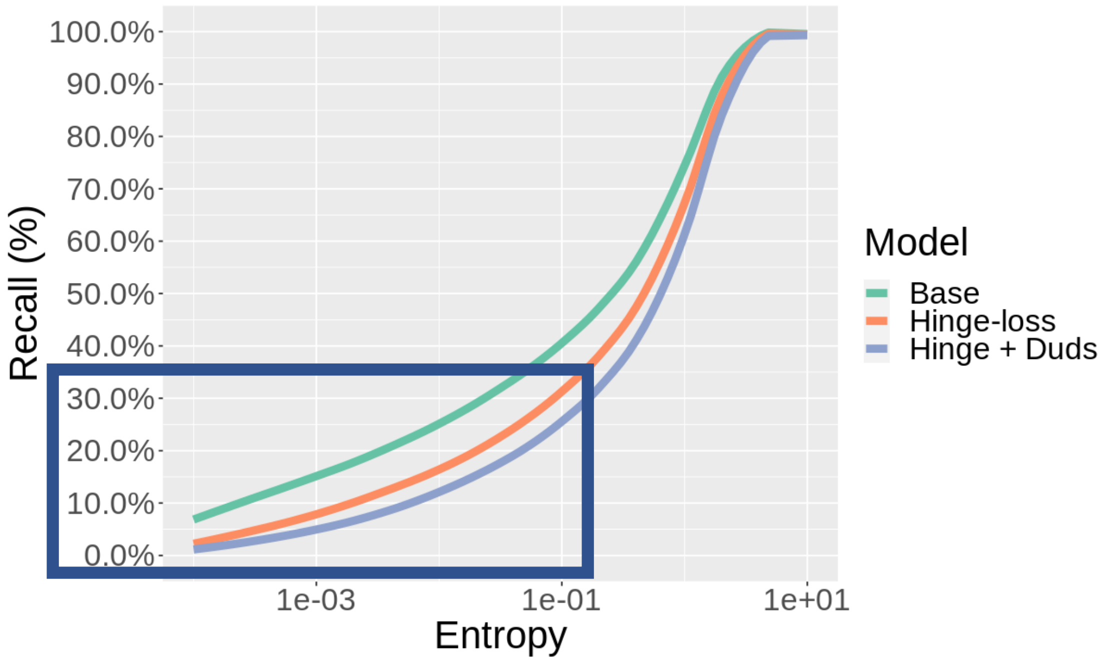
Transformer layer
Feedforward NN
Self-Attention
× 6
Transformer layer
Feedforward NN
Self-Attention
Transformer layer
Feedforward NN
Input Attention
Masked Self-Attention
× 6
Transformer layer
Feedforward NN
Input Attention
Masked Self-Attention
... (int index, Node n) { this.children.add(index, n); n.parent = this; }
n.parent == null


Transformer layer
Feedforward NN
Self-Attention
× 6
Transformer layer
Feedforward NN
Self-Attention
... (int index, Node n) { this.children.add(index, n); n.parent = this; }
Transformer layer
Feedforward NN
Biased Input Attention
Masked Self-Attention
× 6
Transformer layer
Feedforward NN
Biased Input Attention
Masked Self-Attention
n.parent == null


Transformer layer
Feedforward NN
Relational Self-Attention
× 6
Transformer layer
Feedforward NN
Relational Self-Attention
Transformer layer
Feedforward NN
Biased Input Attention
Masked Self-Attention
× 6
Transformer layer
Feedforward NN
Biased Input Attention
Masked Self-Attention
... (int index, Node n) { this.children.add(index, n); n.parent = this; }
n.parent == null
Relations:
parent
def-use
next-use
next-token
computed-from



Precision (%)
100.0%
90.0%
80.0%
70.0%
60.0%
50.0%
40.0%
30.0%
1e-03
1e-01
1e+01
Entropy
Model
Base
Hinge-loss
Hinge + Duds



Recall (%)
100.0%
90.0%
80.0%
70.0%
60.0%
50.0%
40.0%
30.0%
20.0%
10.0%
0.0%
1e-03
1e-01
1e+01
Entropy
Model
Base
Hinge-loss
Hinge + Duds

Bug Detection

```
public Task startTask(String taskId,
        boolean beginConversation) {
  ...
  if (beginConversation) {
    Conversation conversation = conversationInstance.get();
    if (conversation.isTransient()) {
      conversation.begin();
  ...
```

Code sourced from github here.

Bug Detection

```
private void remove(SModelUID uid, String id) {
  String key = uid + "#" + id;

  myMap.remove(key);
  myUIDToKeys.get(uid).remove(key);
  ...
```

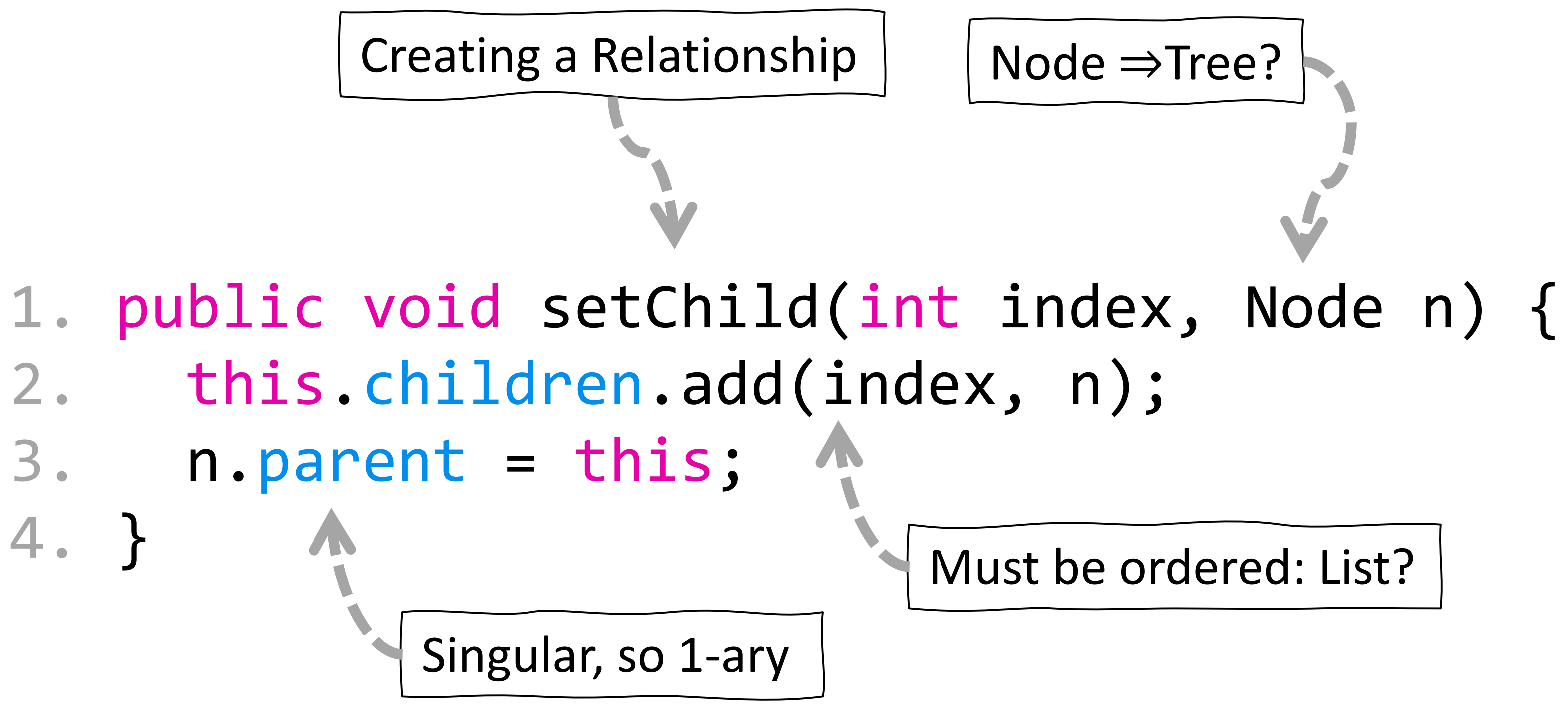
Code sourced from github here.

Bug Detection

```
private void remove(SModelUID uid, String id) {
  String key = uid + "#" + id;
  if (myMap.containsKey(key)) {
    myMap.remove(key);
    myUIDToKeys.get(uid).remove(key);
  ...
```

Code sourced from github here.

What do we see?



```java
public void setChild(int index, Node n) {
  this.children.add(index, n);
  n.parent = this;
}
```

vhellendoorn@cmu.edu