Understanding and Generating Source Code … with Deep Learning

Marc Brockschmidt - MSR Cambridge

@mmjb86

Microsoft

Understanding and Generating Source Code … with Deep Learning

Marc Brockschmidt - MSR Cambridge

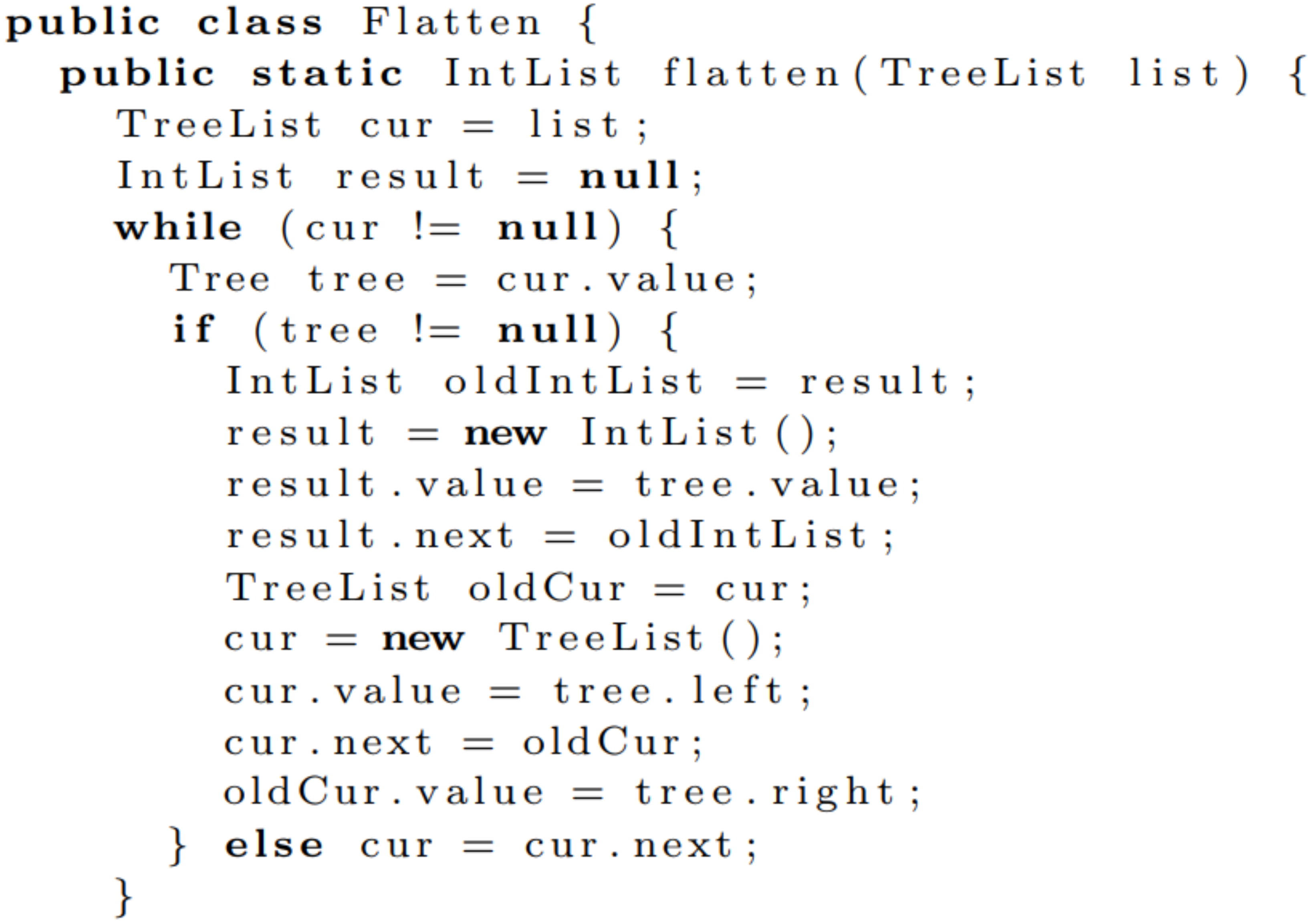
@mmjb86

& MSR collaborators
& MSR Interns
& VS IntelliCode Team

Microsoft

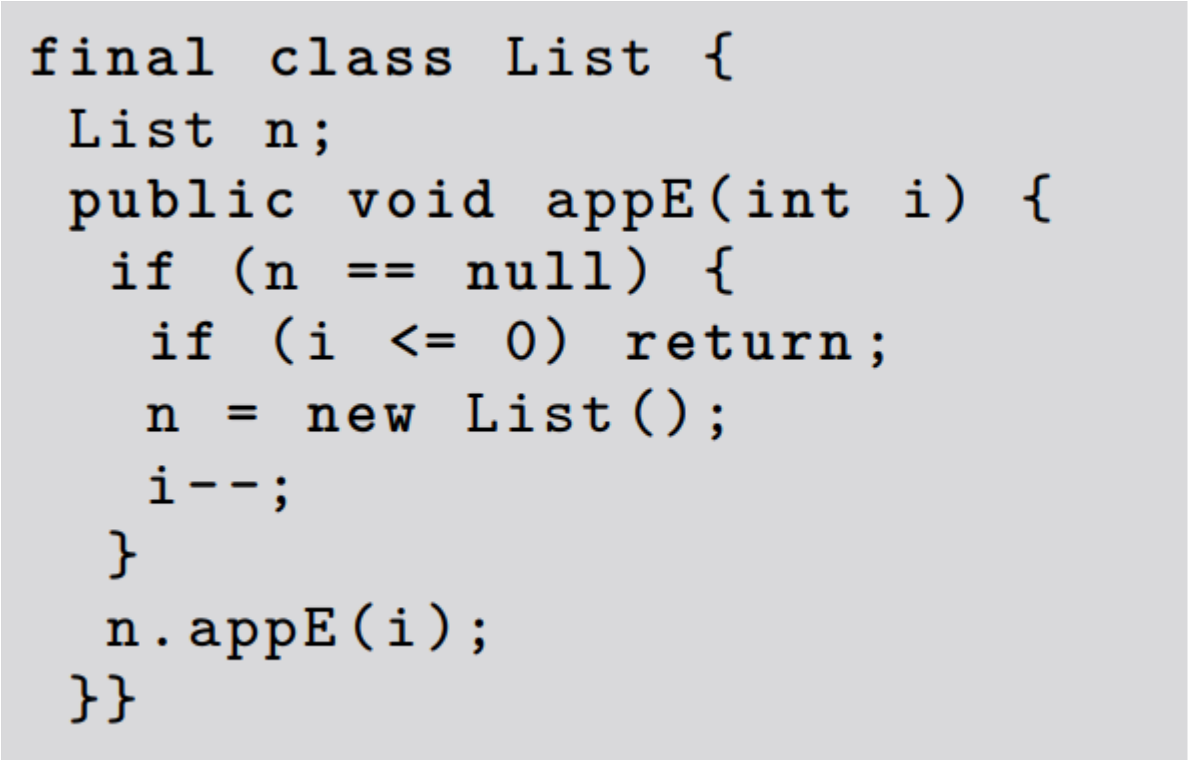
Personal History: Termination Proving

Personal History: Termination Proving



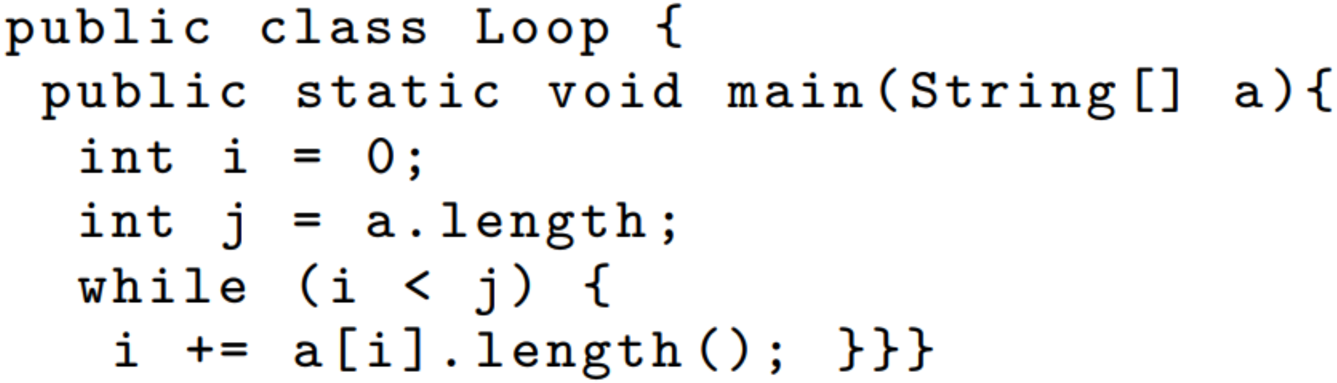
```
public class Flatten {
  public static IntList flatten(TreeList list) {
    TreeList cur = list;
    IntList result = null;
    while (cur != null) {
      Tree tree = cur.value;
      if (tree != null) {
        IntList oldIntList = result;
        result = new IntList();
        result.value = tree.value;
        result.next = oldIntList;
        TreeList oldCur = cur;
        cur = new TreeList();
        cur.value = tree.left;
        cur.next = oldCur;
        oldCur.value = tree.right;
      } else cur = cur.next;
    }
```

Personal History: Termination Proving



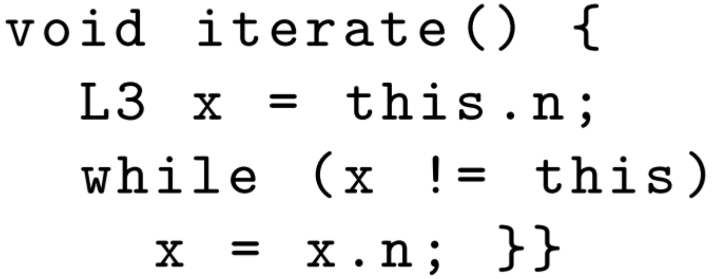
```
final class List {
 List n;
 public void appE(int i) {
  if (n == null) {
   if (i <= 0) return;
   n = new List();
   i--;
  }
  n.appE(i);
 }}
```

Personal History: Termination Proving



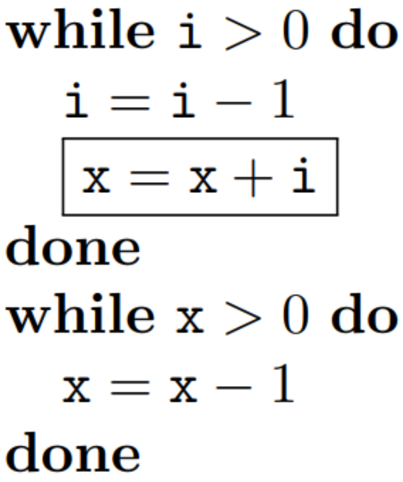
```
public class Loop {
 public static void main(String[] a){
  int i = 0;
  int j = a.length;
  while (i < j) {
   i += a[i].length(); }}}
```

Personal History: Termination Proving



```
void iterate() {
  L3 x = this.n;
  while (x != this)
    x = x.n; }}
```

Personal History: Termination Proving

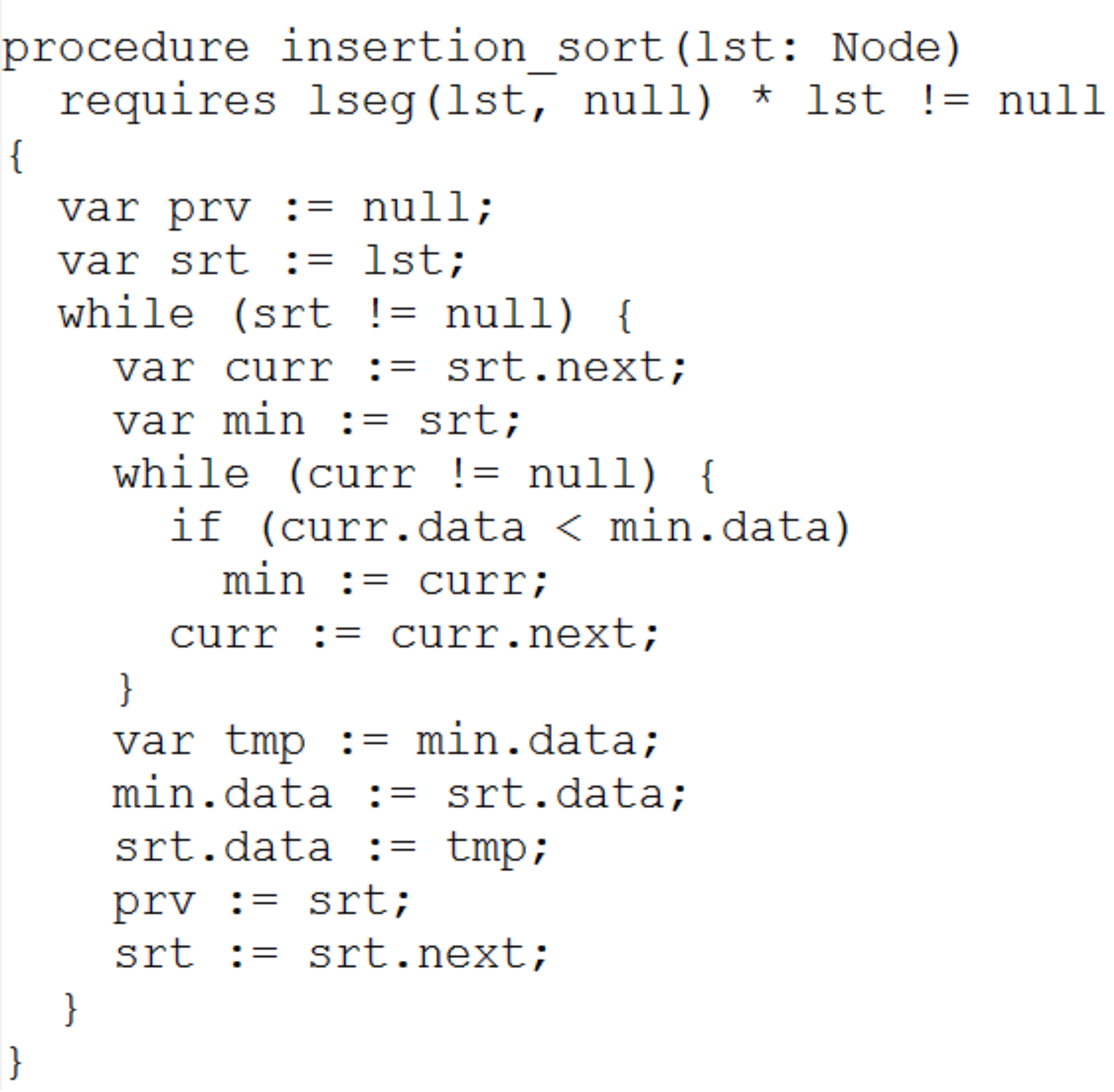


```
while $i > 0$ do
    $i = i - 1$
    $\boxed{x = x + i}$
done
while $x > 0$ do
    $x = x - 1$
done
```

Personal History: Termination Proving

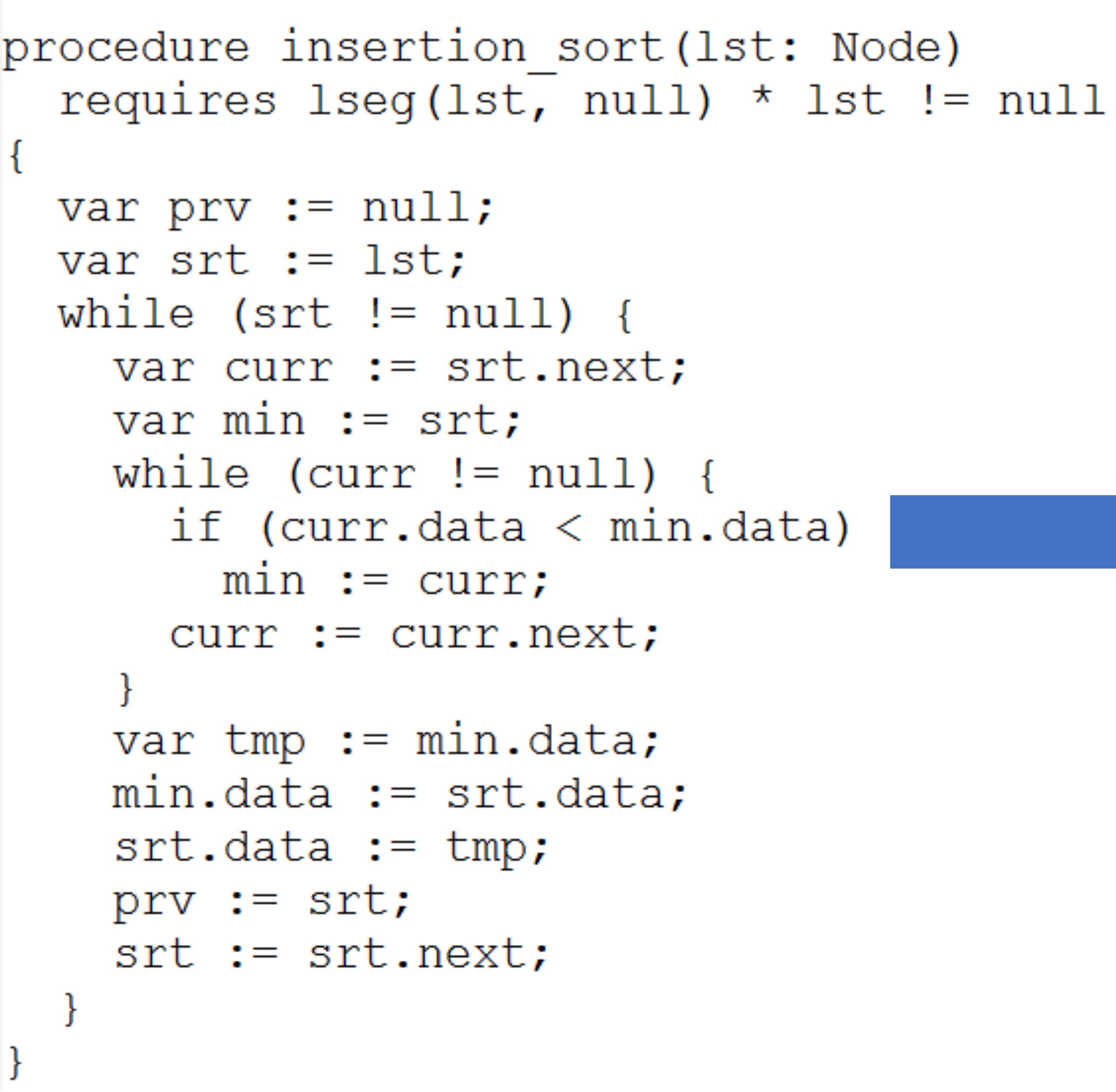
```
System.out.println(“Hello World!”)
```

Learning proofs from data

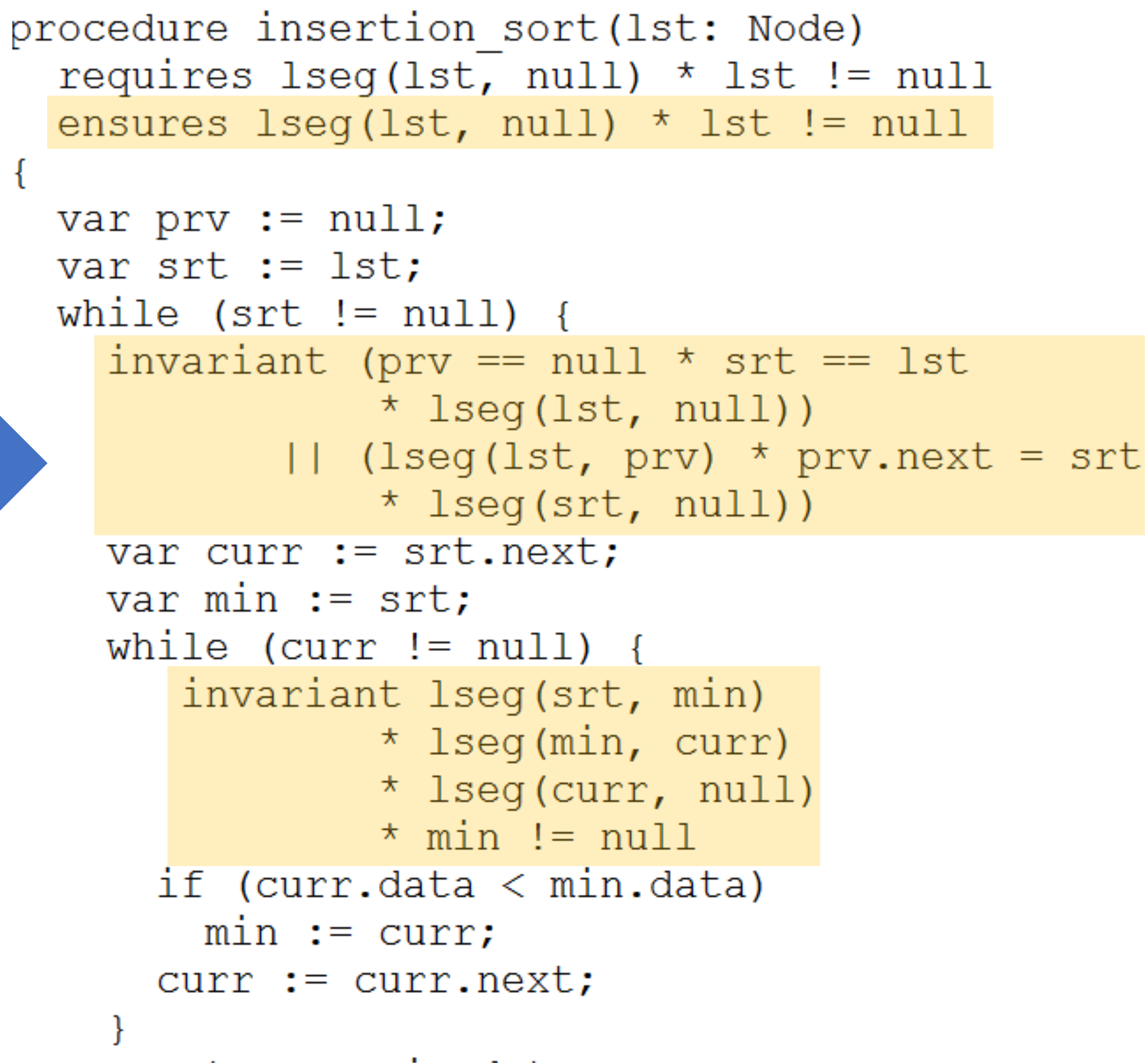


```
procedure insertion_sort(lst: Node)
  requires lseg(lst, null) * lst != null
{
  var prv := null;
  var srt := lst;
  while (srt != null) {
    var curr := srt.next;
    var min := srt;
    while (curr != null) {
      if (curr.data < min.data)
        min := curr;
      curr := curr.next;
    }
    var tmp := min.data;
    min.data := srt.data;
    srt.data := tmp;
    prv := srt;
    srt := srt.next;
  }
}
```

Learning proofs from data

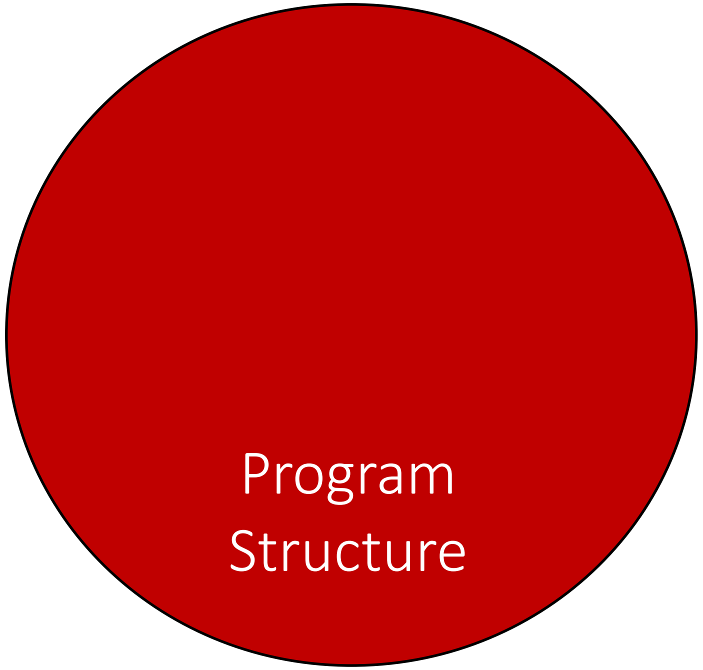


```
procedure insertion_sort(lst: Node)
  requires lseg(lst, null) * lst != null
{
  var prv := null;
  var srt := lst;
  while (srt != null) {
    var curr := srt.next;
    var min := srt;
    while (curr != null) {
      if (curr.data < min.data)
        min := curr;
      curr := curr.next;
    }
    var tmp := min.data;
    min.data := srt.data;
    srt.data := tmp;
    prv := srt;
    srt := srt.next;
  }
}
```



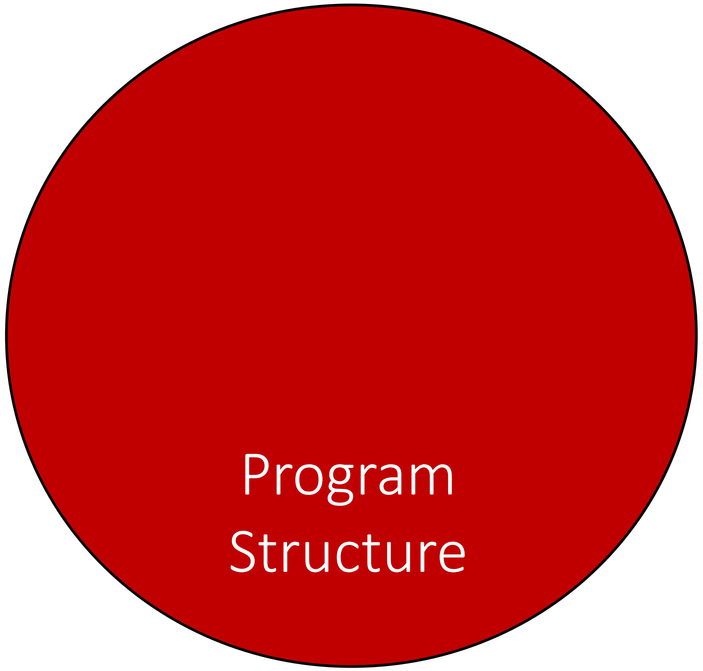
```
procedure insertion_sort(lst: Node)
  requires lseg(lst, null) * lst != null
  ensures lseg(lst, null) * lst != null
{
  var prv := null;
  var srt := lst;
  while (srt != null) {
    invariant (prv == null * srt == lst
                * lseg(lst, null))
           || (lseg(lst, prv) * prv.next = srt
                * lseg(srt, null))
    var curr := srt.next;
    var min := srt;
    while (curr != null) {
      invariant lseg(srt, min)
                * lseg(min, curr)
                * lseg(curr, null)
                * min != null
      if (curr.data < min.data)
        min := curr;
      curr := curr.next;
    }
```

Team Overview



Program Structure

Team Overview

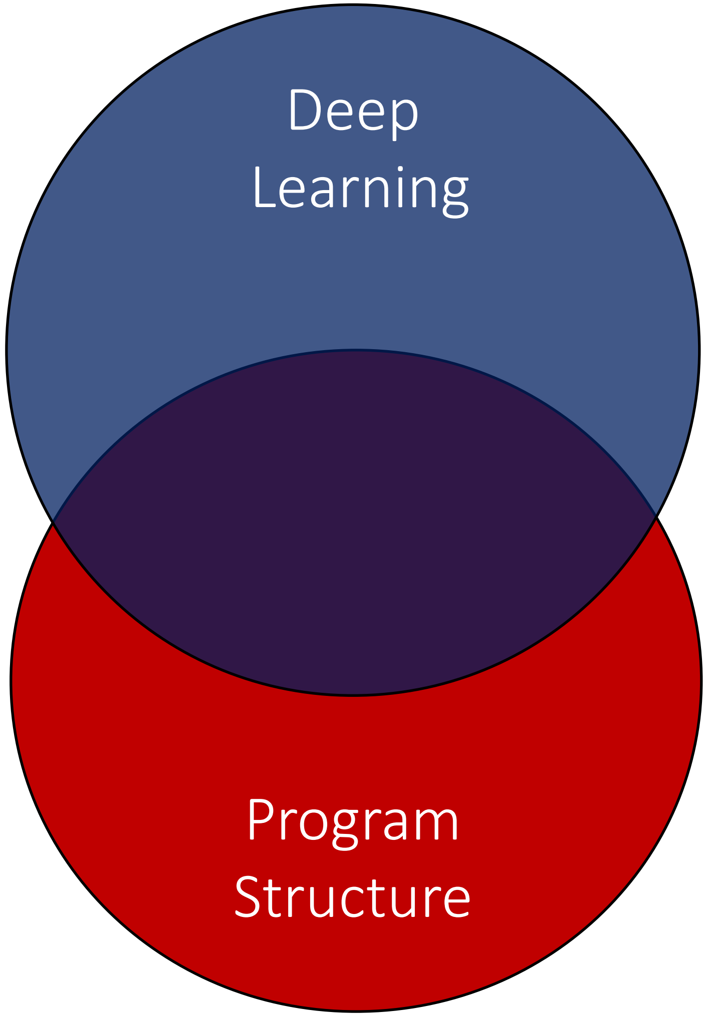


Program Structure

- ✓ Interpretable
- ✓ Generalisation verifiable

- Manual effort
- Limited to specialists

Team Overview



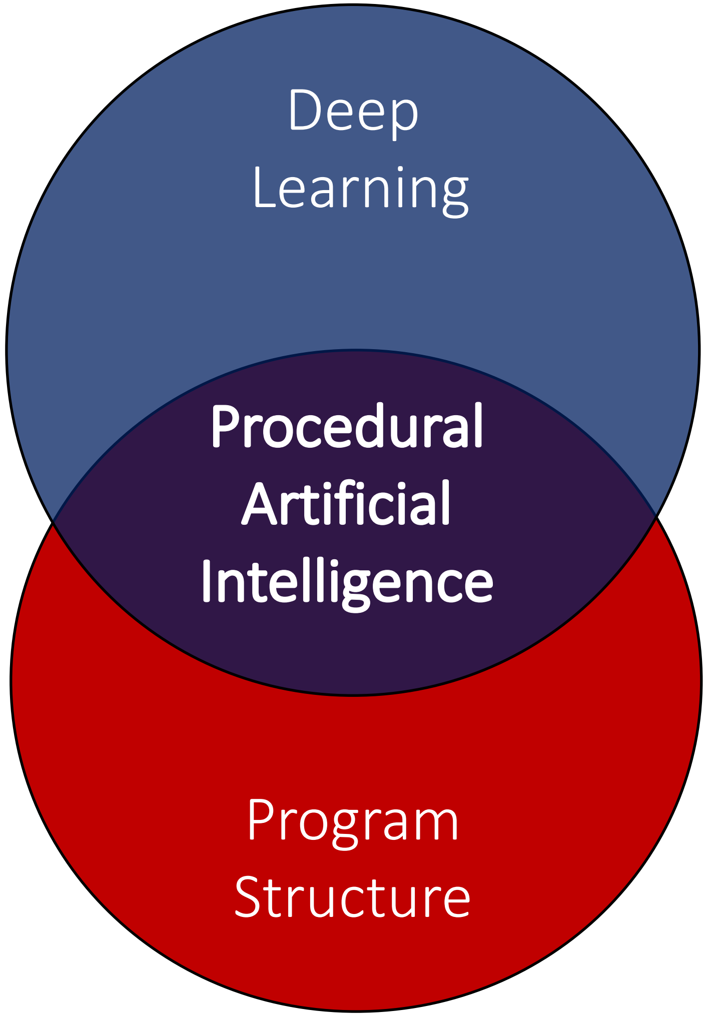
- ✓ Understands images/language/speech
- ✓ Finds patterns in noisy data

- - Requires many samples
- - Handling structured data is hard

- ✓ Interpretable
- ✓ Generalisation verifiable

- - Manual effort
- - Limited to specialists

Team Overview



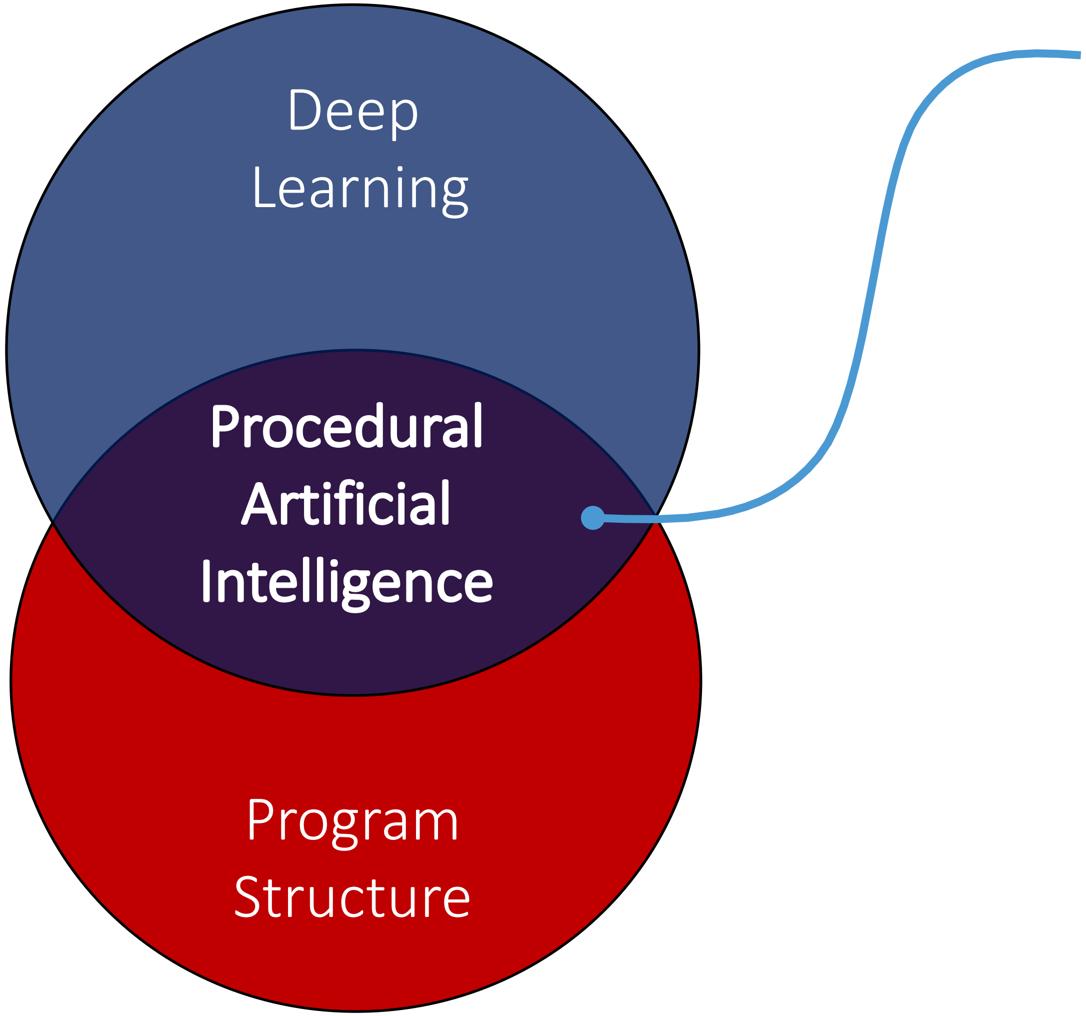
- ✓ Understands images/language/speech
- ✓ Finds patterns in noisy data

- Requires many samples
- Handling structured data is hard

- ✓ Interpretable
- ✓ Generalisation verifiable

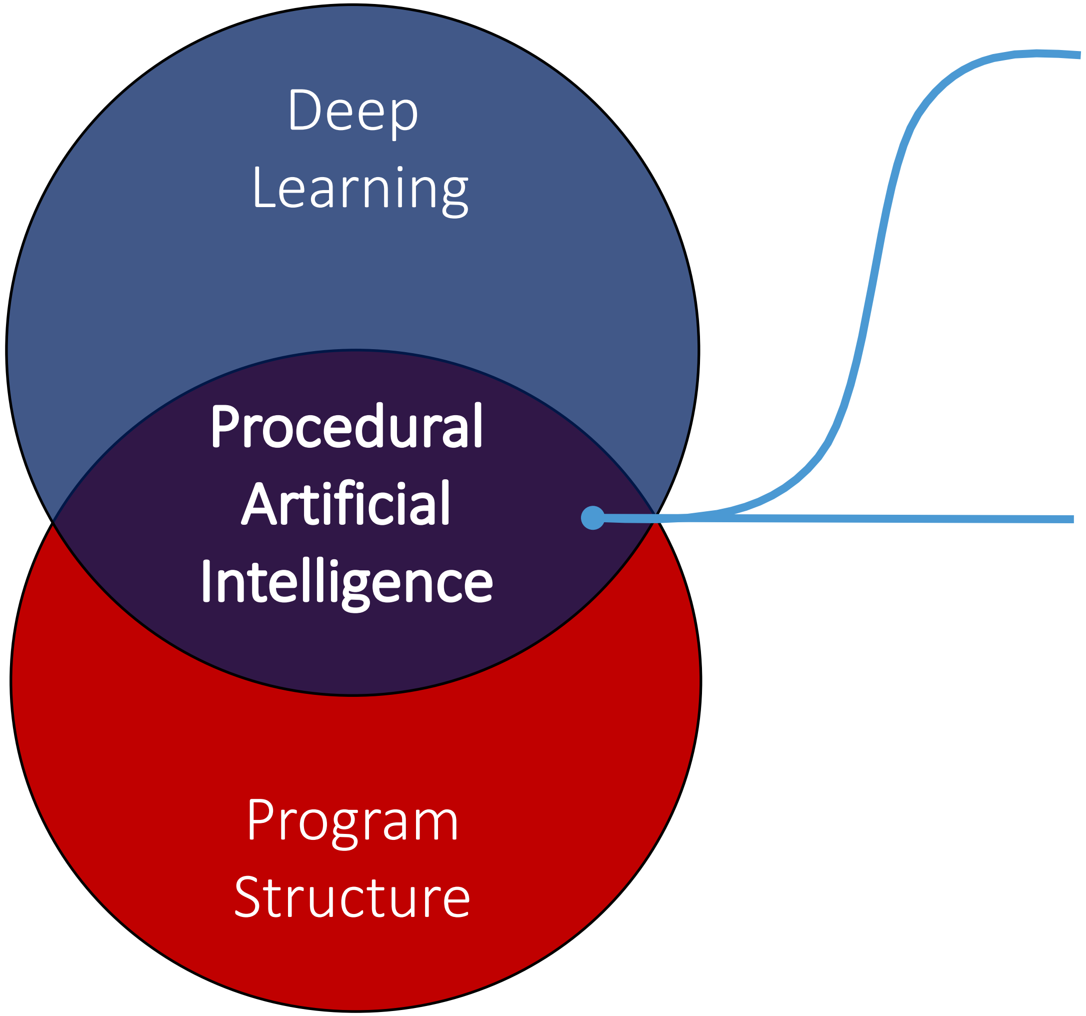
- Manual effort
- Limited to specialists

Team Overview



Understanding Programs

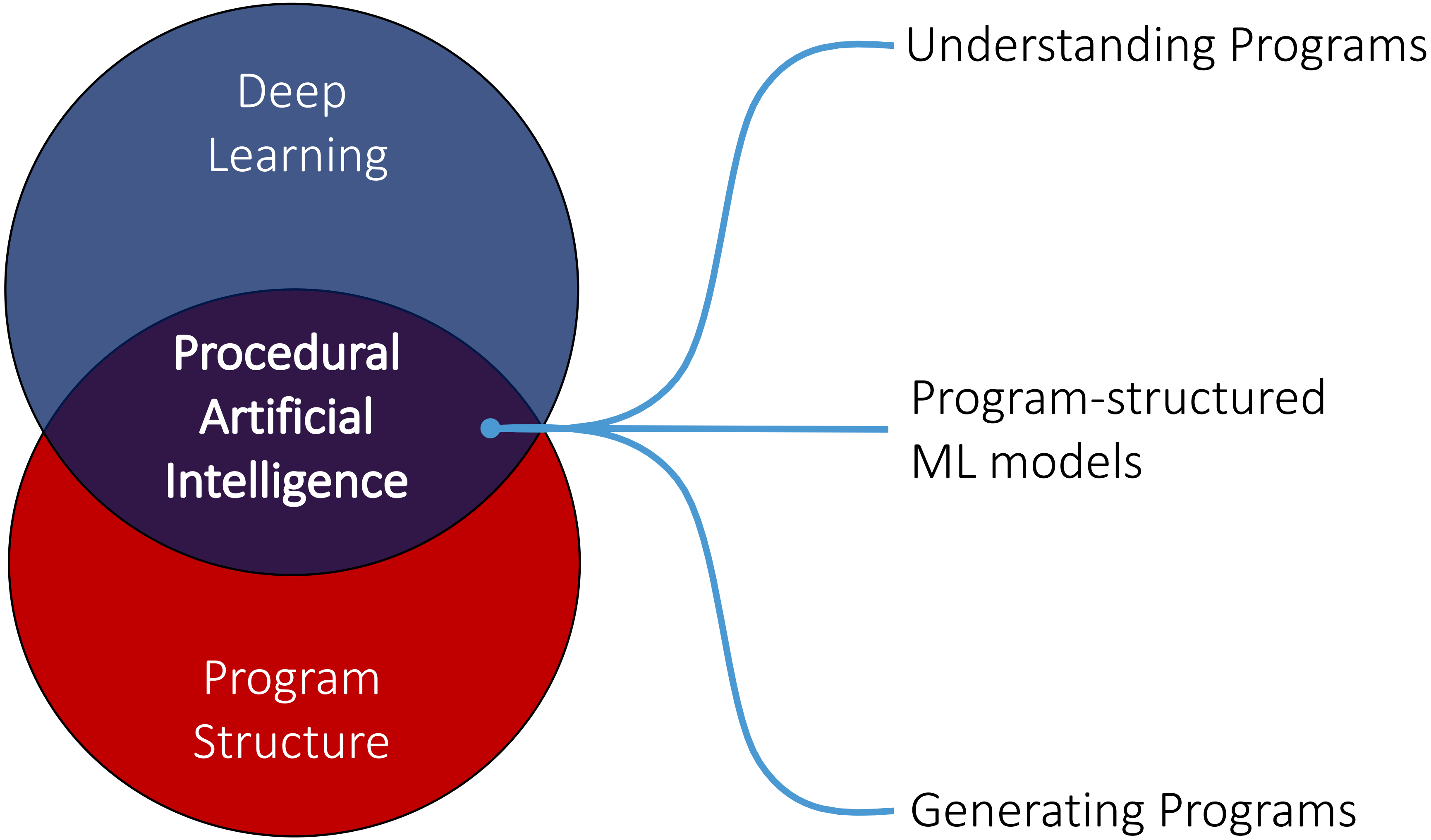
Team Overview



Understanding Programs

Program-structured
ML models

Team Overview



Deep Learning

Procedural Artificial Intelligence

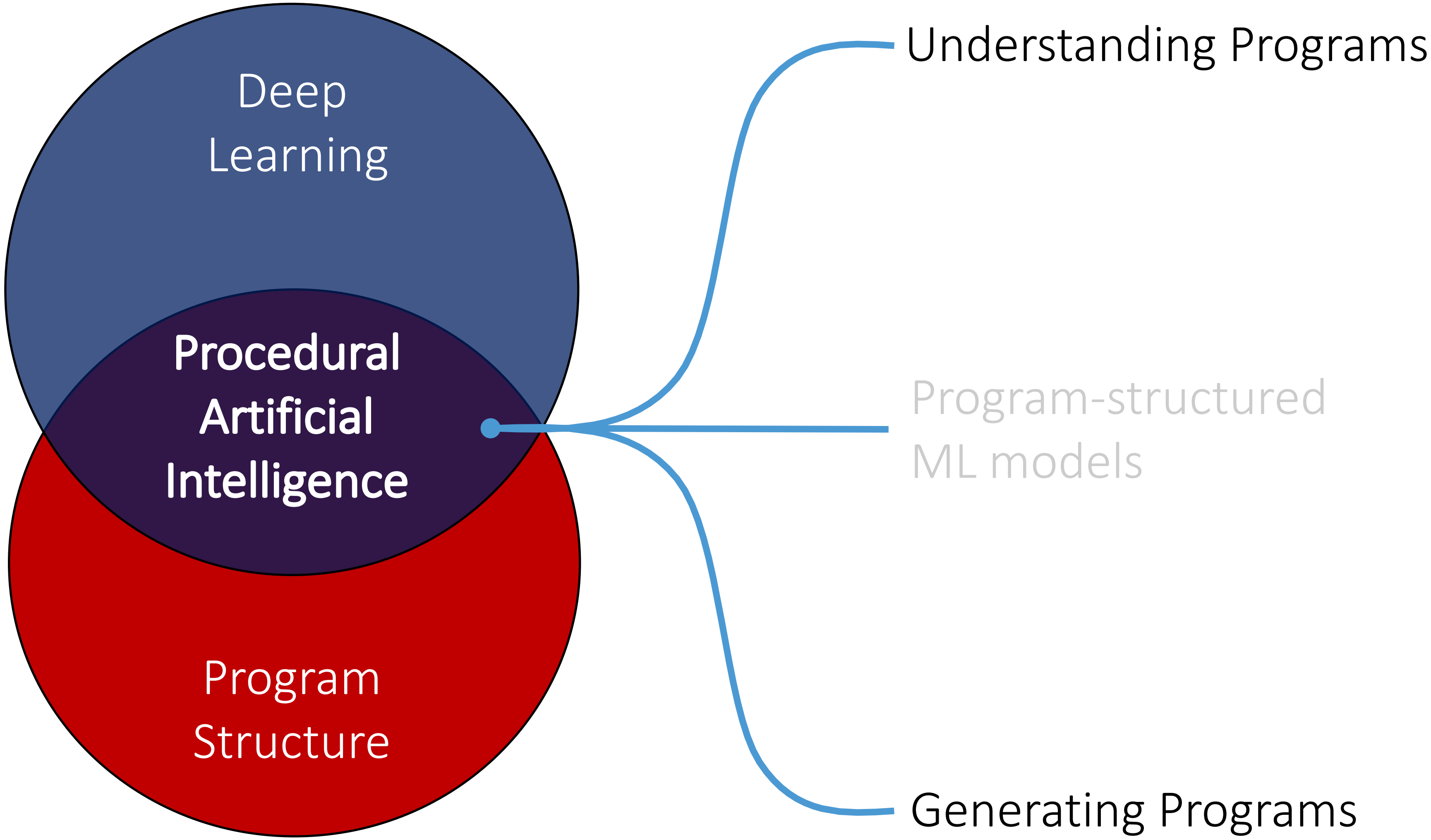
Program Structure

Understanding Programs

Program-structured ML models

Generating Programs

Team Overview



Deep Learning

Procedural Artificial Intelligence

Program Structure

Understanding Programs

Program-structured ML models

Generating Programs

The Big Picture

Big Code: Potential

Big Code: Potential

Code Patterns

Big Code: Potential

Code Patterns

Natural Language

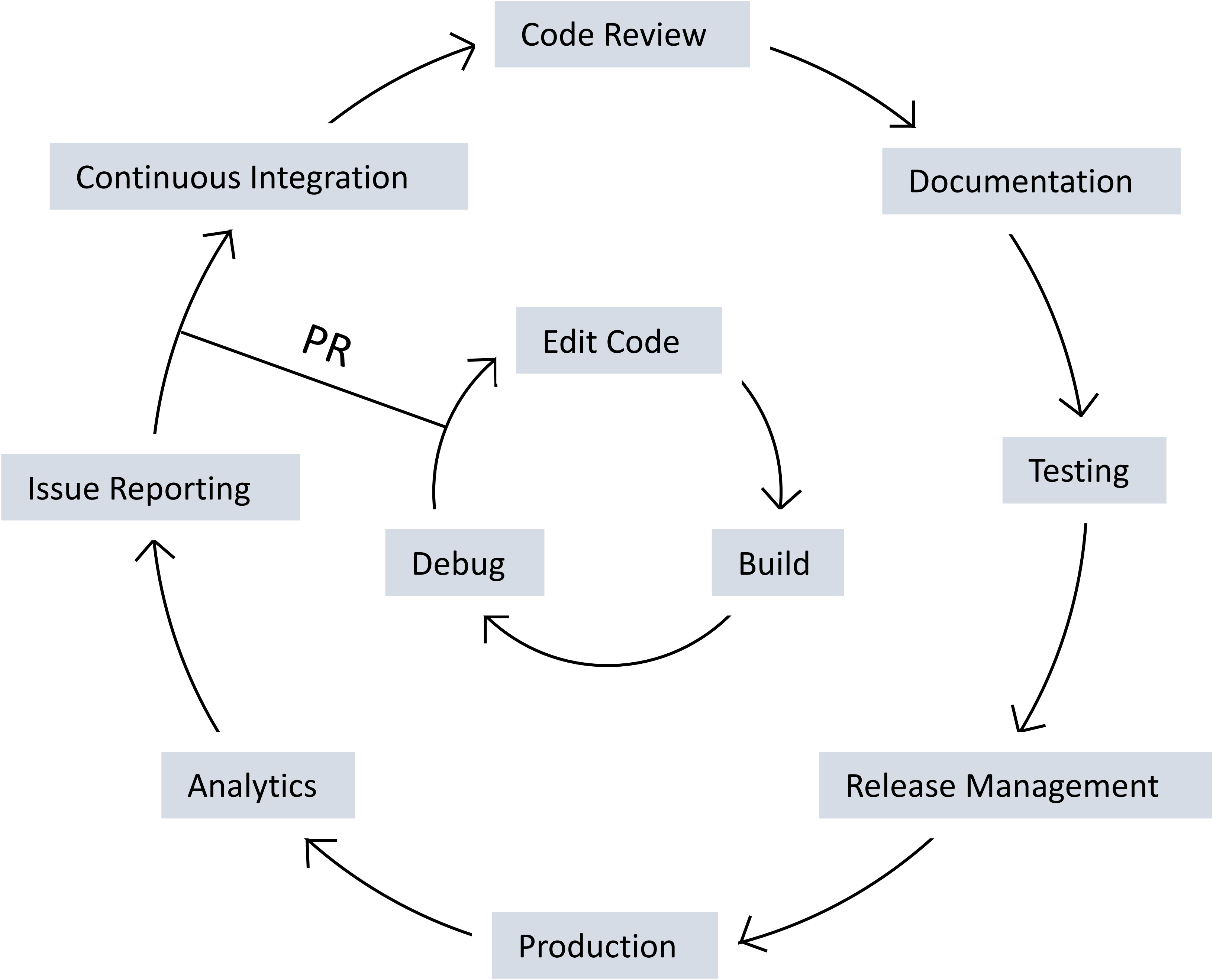
Big Code: Potential

Code Patterns

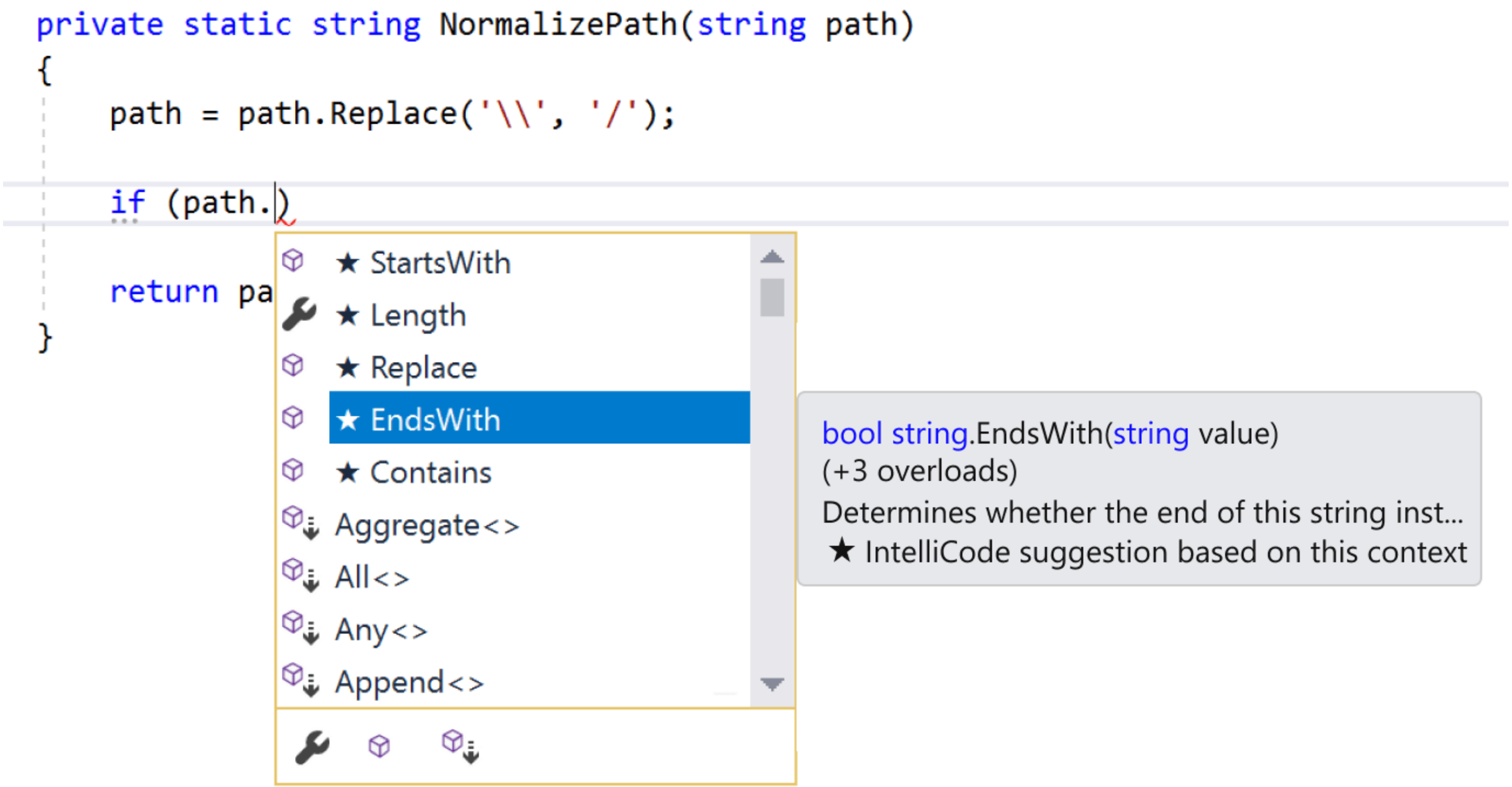
Natural Language

Development Histories

Microsoft
GitHub


Code Review
Documentation
Testing
Release Management
Production
Analytics
Issue Reporting
Continuous Integration
PR
Edit Code
Build
Debug

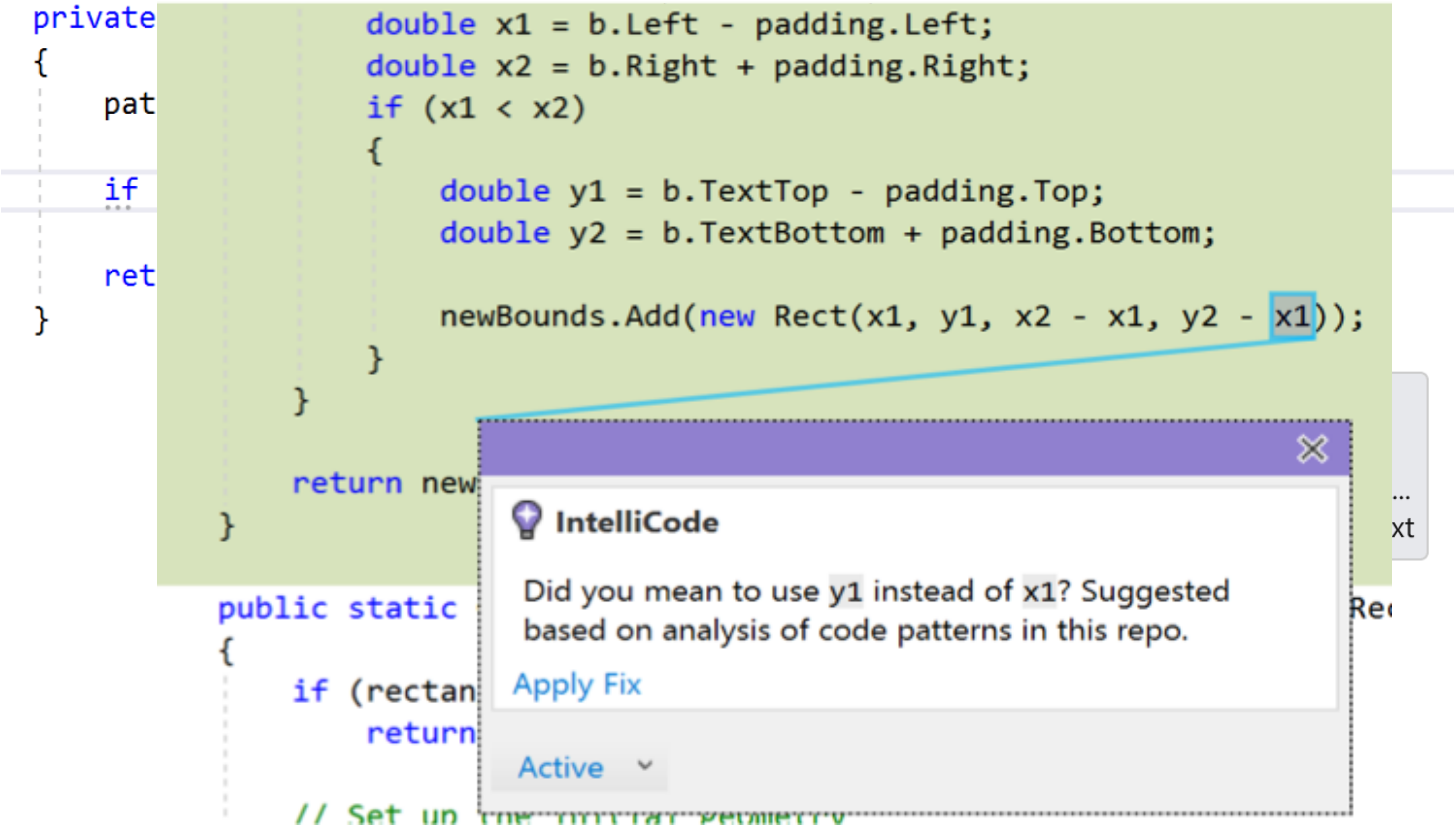
Visual Studio IntelliCode



```
private static string NormalizePath(string path)
{
    path = path.Replace('\\', '/');

    if (path.)

    return pa
}
```

bool string.EndsWith(string value)
(+3 overloads)
Determines whether the end of this string inst...
★ IntelliCode suggestion based on this context

Visual Studio IntelliCode



```
private
{
    pat

    if

    ret
}
```

```
            double x1 = b.Left - padding.Left;
            double x2 = b.Right + padding.Right;
            if (x1 < x2)
            {
                double y1 = b.TextTop - padding.Top;
                double y2 = b.TextBottom + padding.Bottom;

                newBounds.Add(new Rect(x1, y1, x2 - x1, y2 - x1));
            }
        }

        return new
    }

    public static
    {
        if (rectan
            return

        // Set up
```

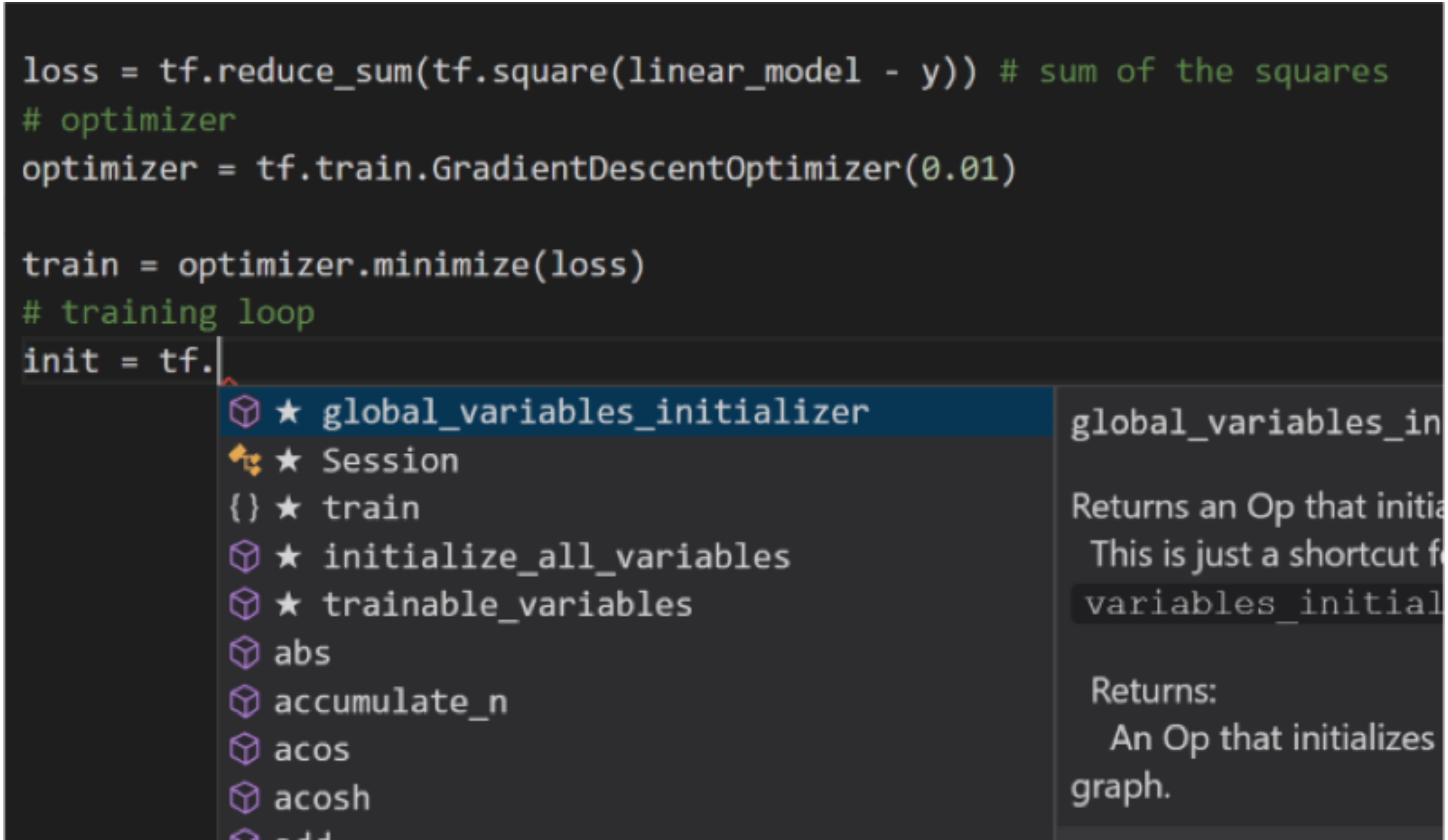
IntelliCode

Did you mean to use y1 instead of x1? Suggested based on analysis of code patterns in this repo.

Apply Fix

Active

Visual Studio IntelliCode



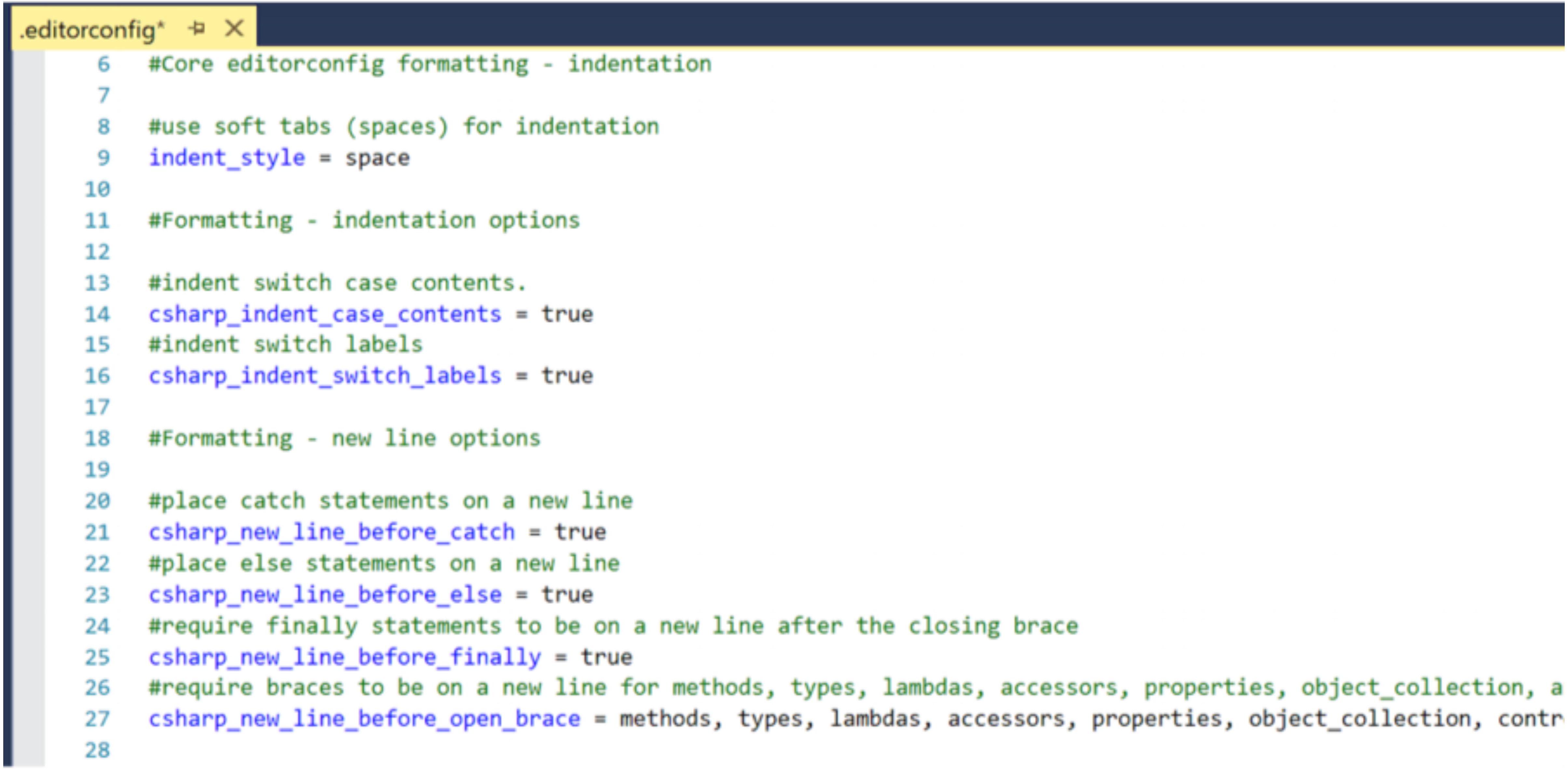
```
loss = tf.reduce_sum(tf.square(linear_model - y)) # sum of the squares
# optimizer
optimizer = tf.train.GradientDescentOptimizer(0.01)

train = optimizer.minimize(loss)
# training loop
init = tf.
```

global_variables_in

Returns an Op that initia
This is just a shortcut f
variables_initial

Returns:
An Op that initializes
graph.

Visual Studio IntelliCode

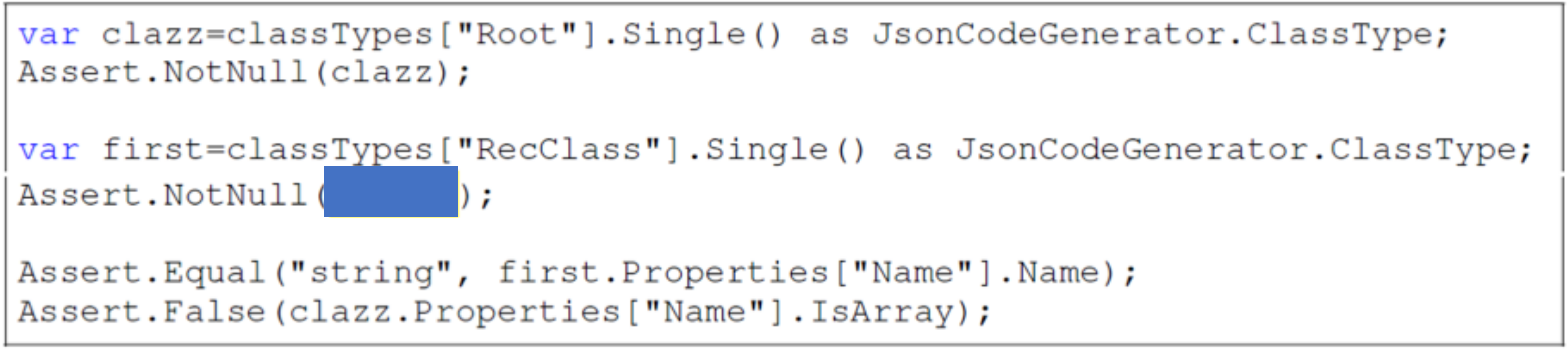


```
#Core editorconfig formatting - indentation

#use soft tabs (spaces) for indentation
indent_style = space

#Formatting - indentation options

#indent switch case contents.
csharp_indent_case_contents = true
#indent switch labels
csharp_indent_switch_labels = true

#Formatting - new line options

#place catch statements on a new line
csharp_new_line_before_catch = true
#place else statements on a new line
csharp_new_line_before_else = true
#require finally statements to be on a new line after the closing brace
csharp_new_line_before_finally = true
#require braces to be on a new line for methods, types, lambdas, accessors, properties, object_collection, a
csharp_new_line_before_open_brace = methods, types, lambdas, accessors, properties, object_collection, contr

```

Understanding Programs

Task: Detecting Variable Misuse

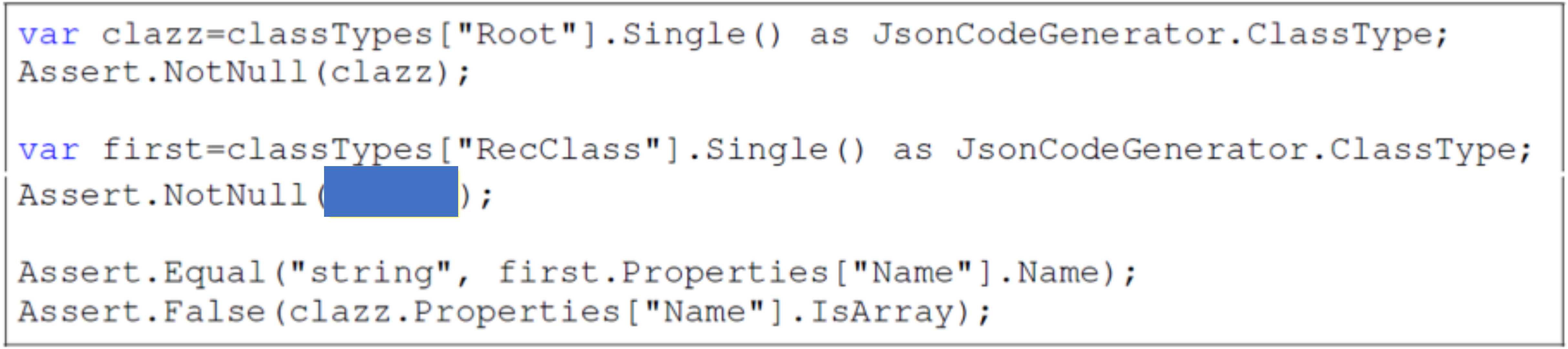
Given location in program code, identify which variable should be used:



```
var clazz=classTypes["Root"].Single() as JsonCodeGenerator.ClassType;
Assert.NotNull(clazz);

var first=classTypes["RecClass"].Single() as JsonCodeGenerator.ClassType;
Assert.NotNull(        );

Assert.Equal("string", first.Properties["Name"].Name);
Assert.False(clazz.Properties["Name"].IsArray);
```

Task: Detecting Variable Misuse

Given location in program code, identify which variable should be used:

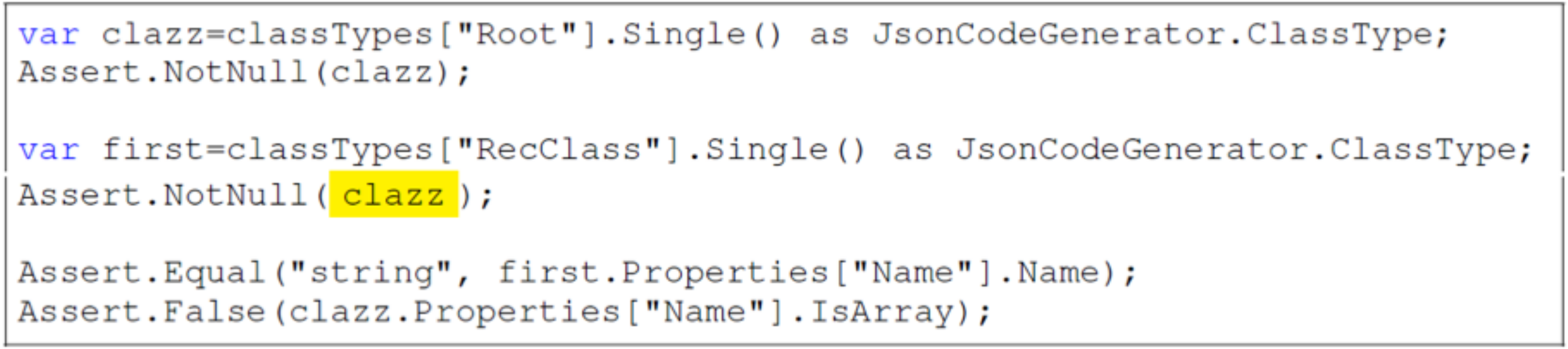


```
var clazz=classTypes["Root"].Single() as JsonCodeGenerator.ClassType;
Assert.NotNull(clazz);

var first=classTypes["RecClass"].Single() as JsonCodeGenerator.ClassType;
Assert.NotNull([redacted]);

Assert.Equal("string", first.Properties["Name"].Name);
Assert.False(clazz.Properties["Name"].IsArray);
```

Possible type-correct options: `clazz, first`

Task: Detecting Variable Misuse

Given location in program code, identify which variable should be used:



```
var clazz=classTypes["Root"].Single() as JsonCodeGenerator.ClassType;
Assert.NotNull(clazz);

var first=classTypes["RecClass"].Single() as JsonCodeGenerator.ClassType;
Assert.NotNull( clazz );

Assert.Equal("string", first.Properties["Name"].Name);
Assert.False(clazz.Properties["Name"].IsArray);
```

Possible type-correct options: `clazz, first`

⚠ Not easy to catch with static analysis tools.

Task: Suggesting Good Variable Names

```
int SumEven(int[] arr, int lim) {
  int       = 0;
  for (int i = 0; i < lim; i++)
    if (arr[i] % 2 == 0)
           += arr[i];

  return      ;
}
```

Sum Of Even

Analysing Code: PL View

Analysing Code: PL View

Approach 1: Proving Software Correct

- Needs Specifications
- Limited Domains
- Limited Size

Approach 2: Finding Software Bugs

- Manual Error Pattern Definitions
- Hard to Configure

Analysing Code: ML View

Analysing Code: ML View

Approach 1.1: Sequence or tree of words

Analysing Code: ML View

Approach 1.1: Sequence or tree of words (re-using NLP ideas)

Programs are different from natural language:

- Semantics for keywords already known
- Many words (APIs, local methods) only used seldomly
- Long-distance dependencies common

Approach 2: Graphs

- Nodes labelled by semantic information
- Edges for semantic relationships

Analysing Code: ML View

Approach 1.1: Sequence or tree of words (re-using NLP ideas)

Programs are different from natural language:

- Semantics for keywords already known
- Many words (APIs, local methods) only used seldomly
- Long-distance dependencies common

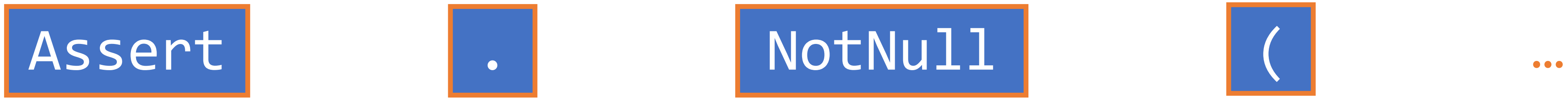
Approach 2: Graphs

- Nodes labelled by semantic information
- Edges for semantic relationships

} From Static Analysis

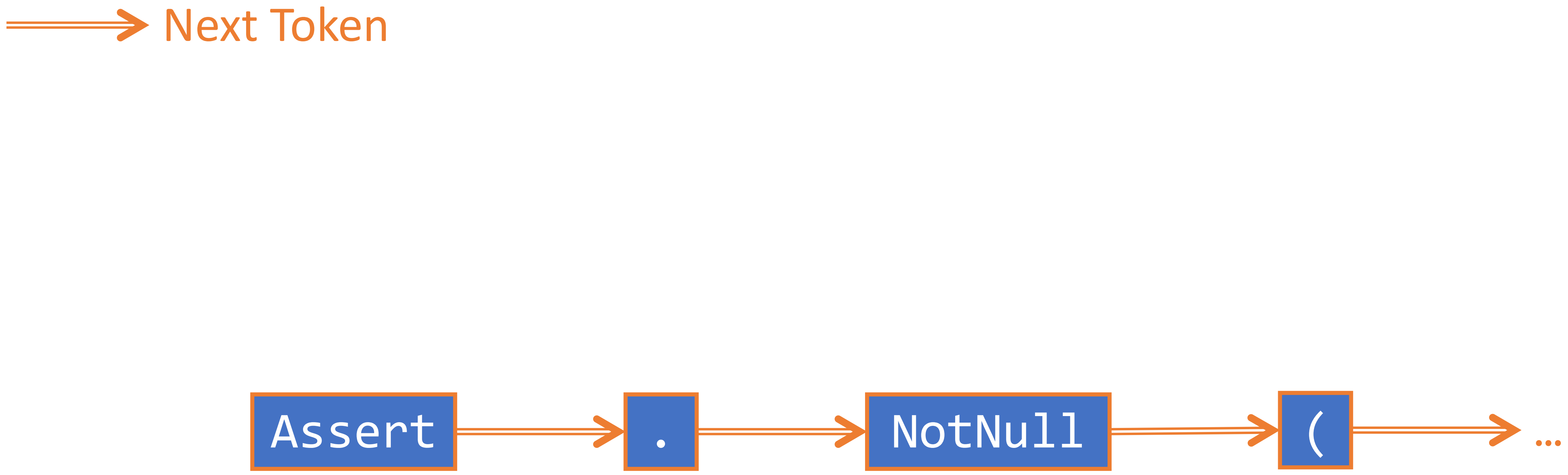
Programs as Graphs: Syntax

```
Assert.NotNull(clazz);
```



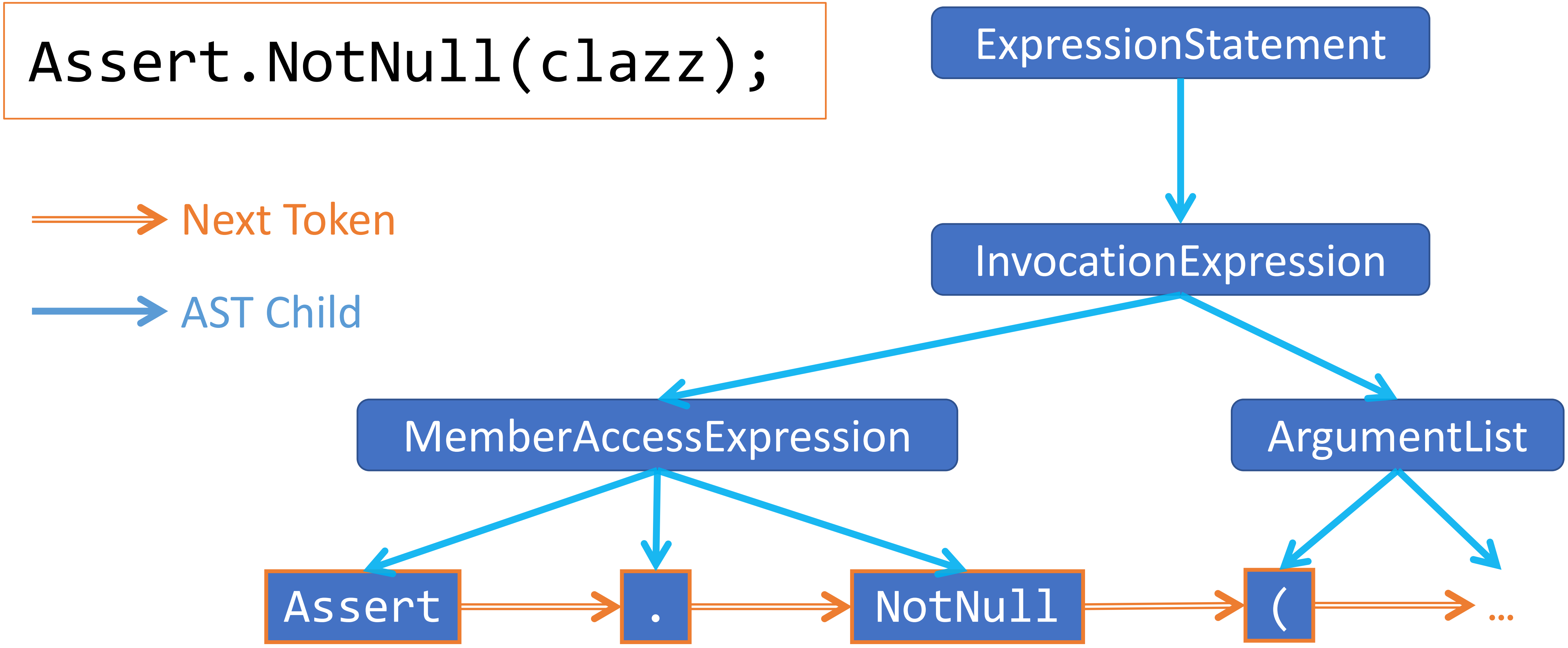
Programs as Graphs: Syntax

```
Assert.NotNull(clazz);
```



Programs as Graphs: Syntax

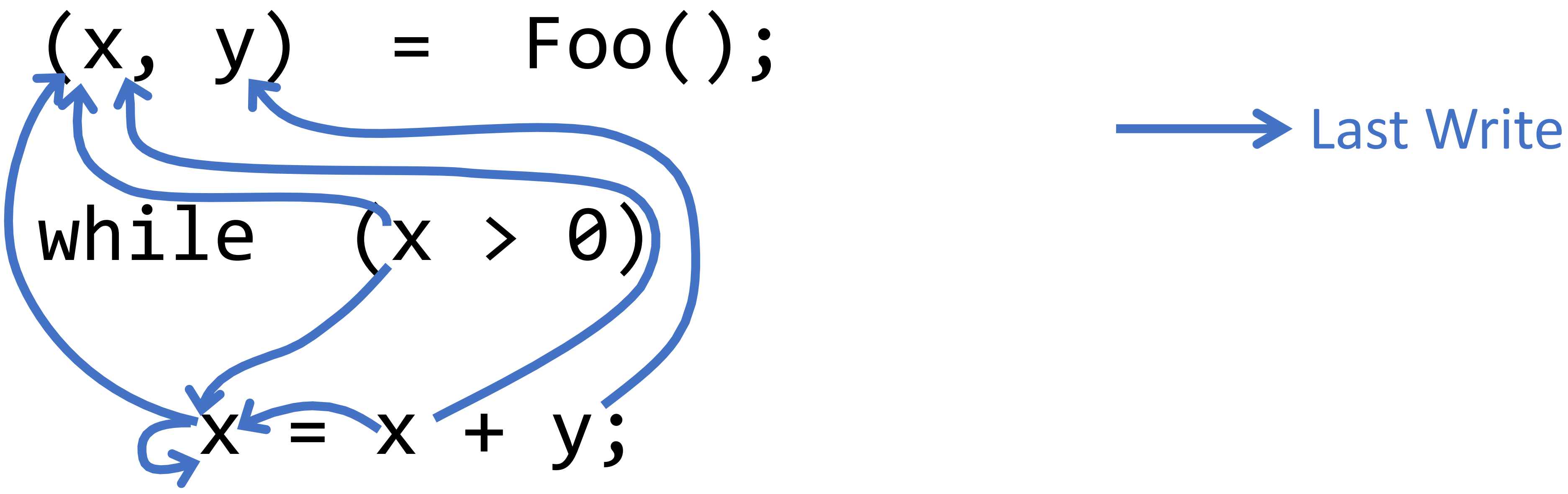
```
Assert.NotNull(clazz);
```



Programs as Graphs: Data Flow

```
(x, y)  =  Foo();

while  (x > 0)

    x = x + y;
```

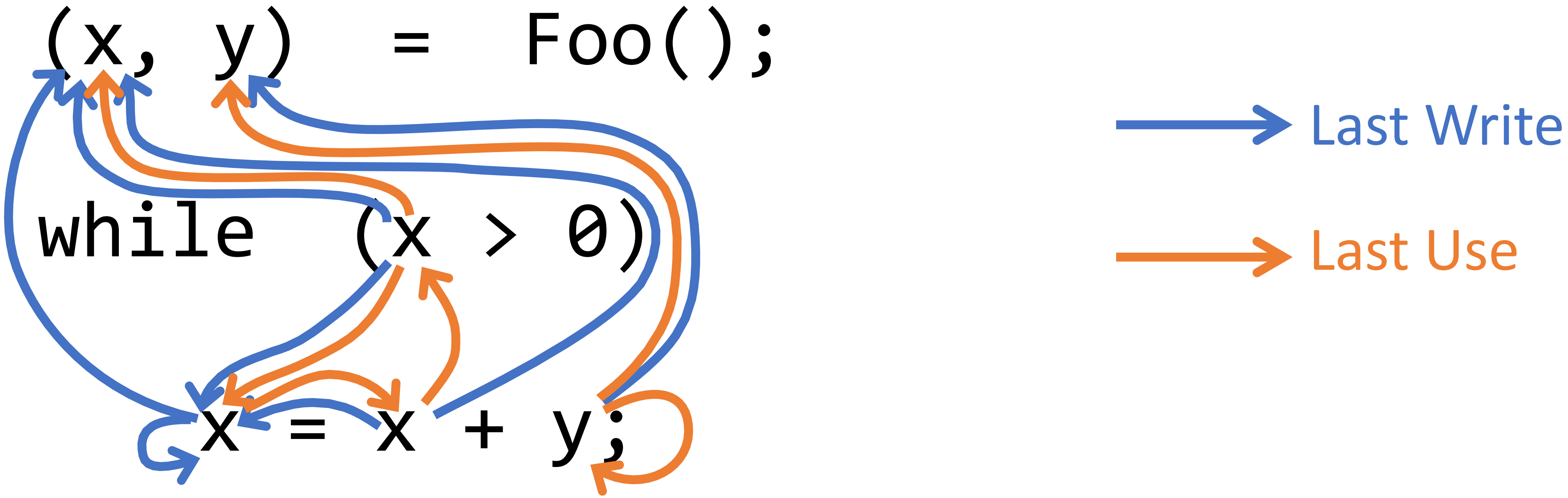
Programs as Graphs: Data Flow



```
(x, y) = Foo();

while (x > 0)

    x = x + y;
```

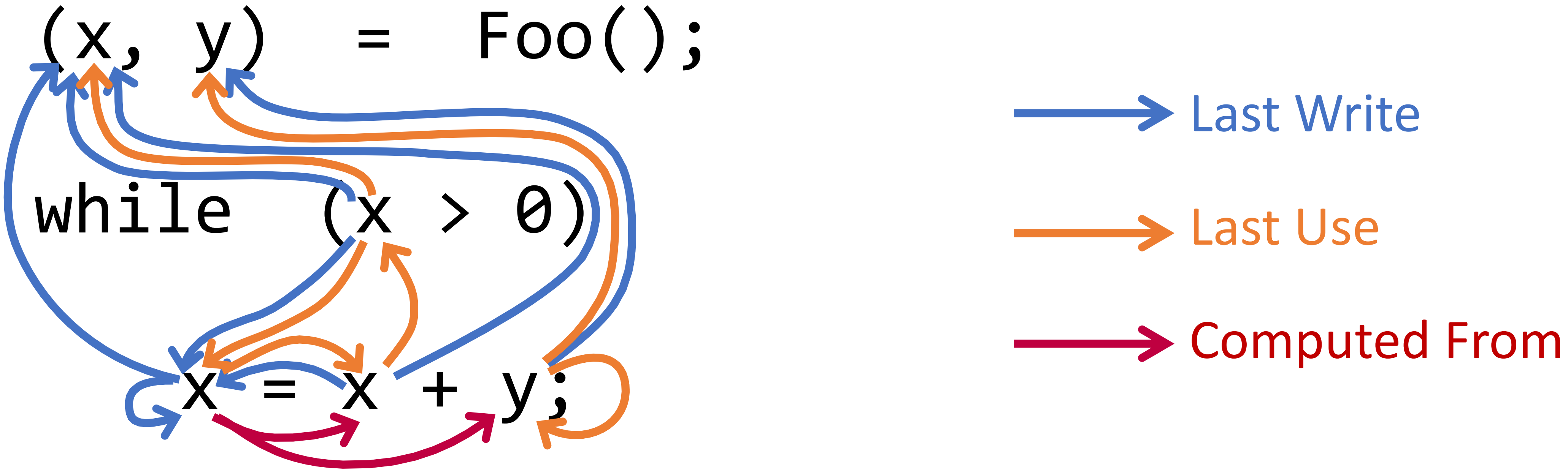
→ Last Write

Programs as Graphs: Data Flow



```
(x, y)  =  Foo();

while  (x > 0)

     x = x + y;
```

Programs as Graphs: Data Flow



```
(x, y) = Foo();
while (x > 0)
    x = x + y;
```

→ Last Write

→ Last Use

→ Computed From

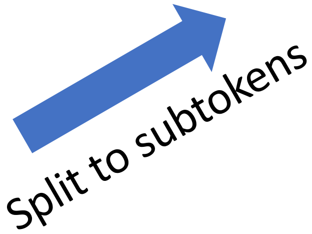
Programs as Graphs: Node Representation

Label: `outFilePrefix`

Type: `string`

Programs as Graphs: Node Representation

`out, file, prefix`

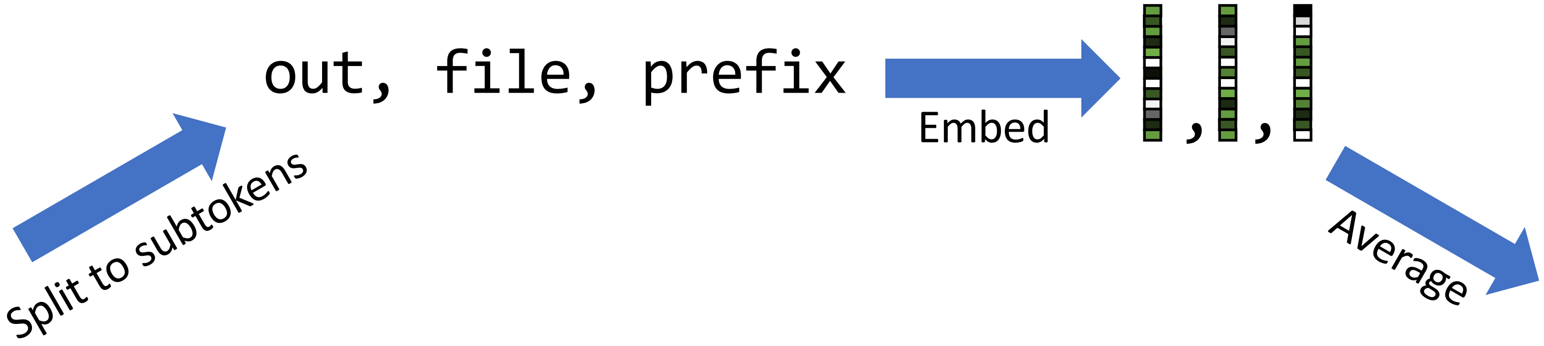


Split to subtokens

Label: `outFilePrefix`

Type: `string`

Programs as Graphs: Node Representation



Split to subtokens

`out, file, prefix`

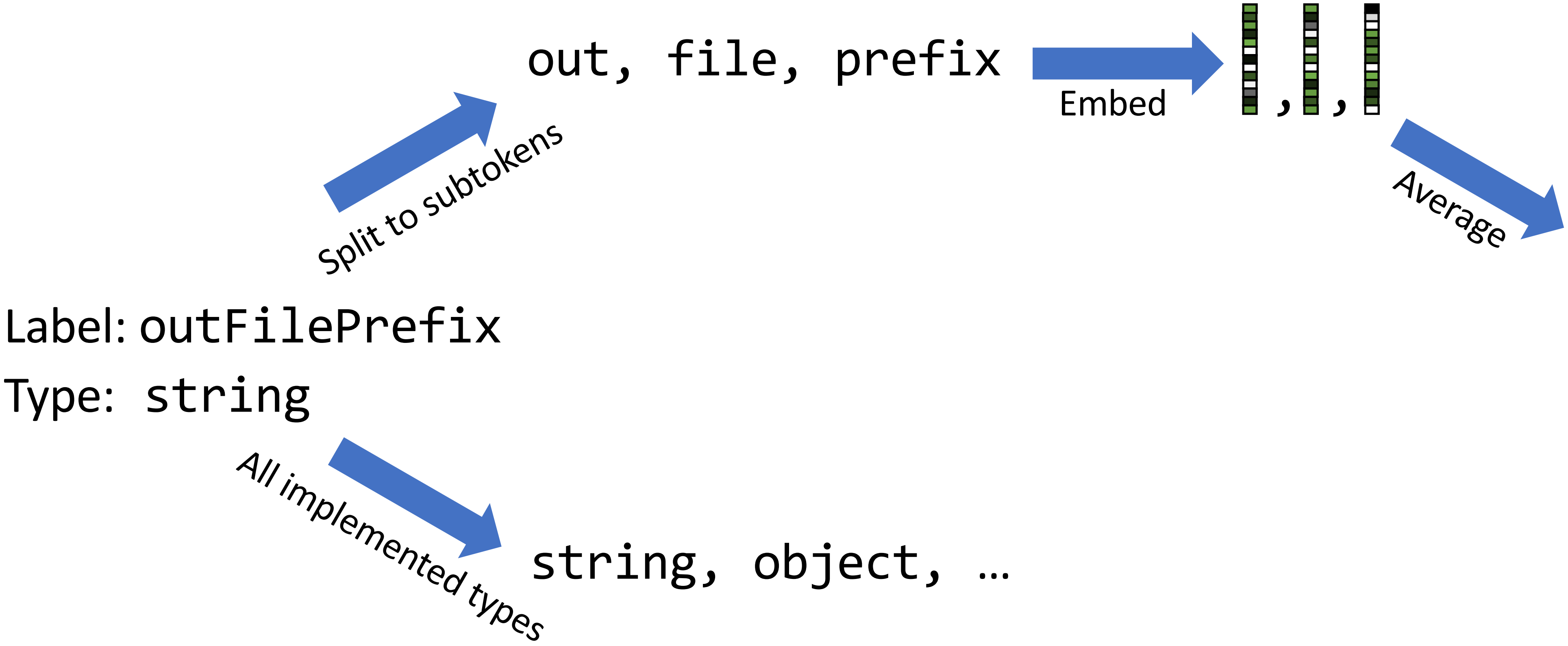
Embed

Average

Label: `outFilePrefix`

Type: `string`

Programs as Graphs: Node Representation



Label: `outFilePrefix`

Type: `string`

Split to subtokens

`out, file, prefix`

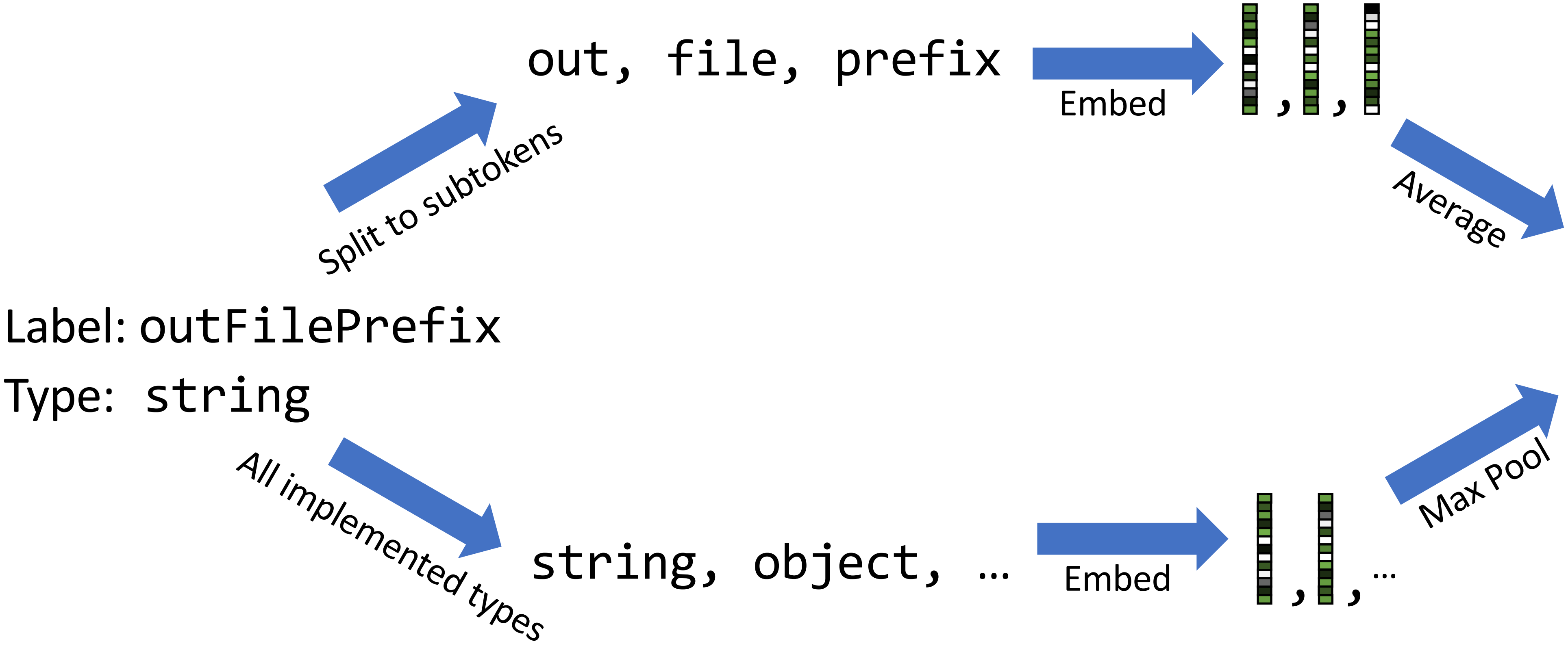
Embed

Average

All implemented types

`string, object, …`

Programs as Graphs: Node Representation



Label: `outFilePrefix`

Type: `string`

Split to subtokens

`out, file, prefix`

Embed

Average

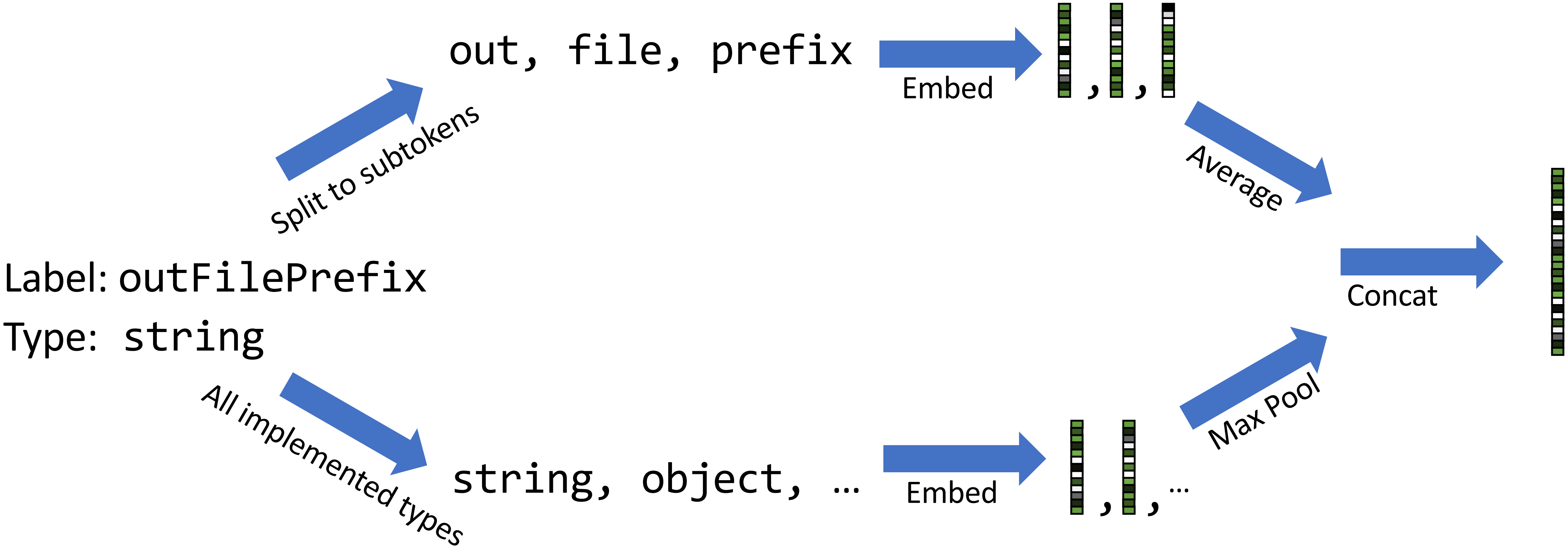
All implemented types

`string, object, …`

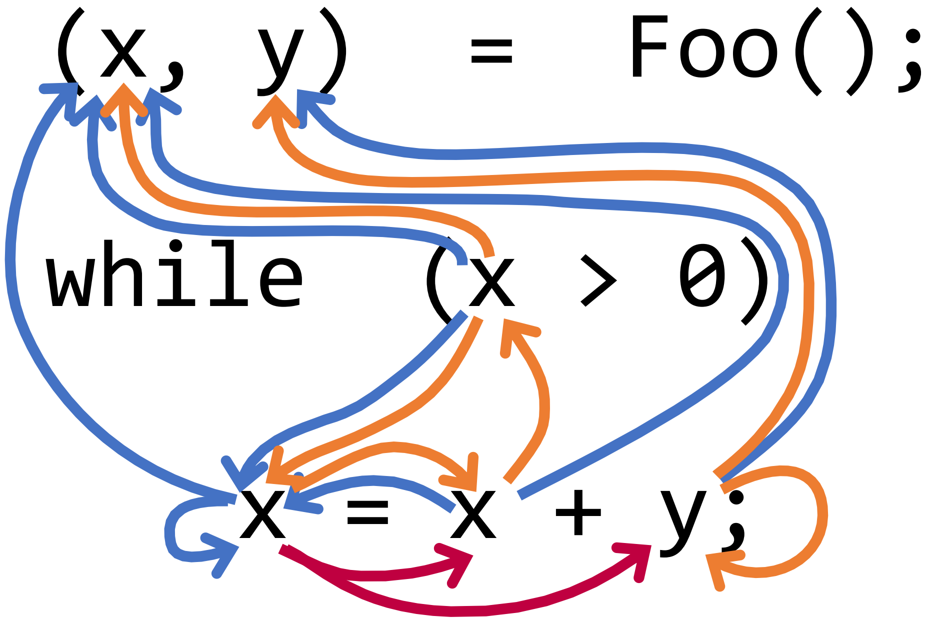
Embed

Max Pool

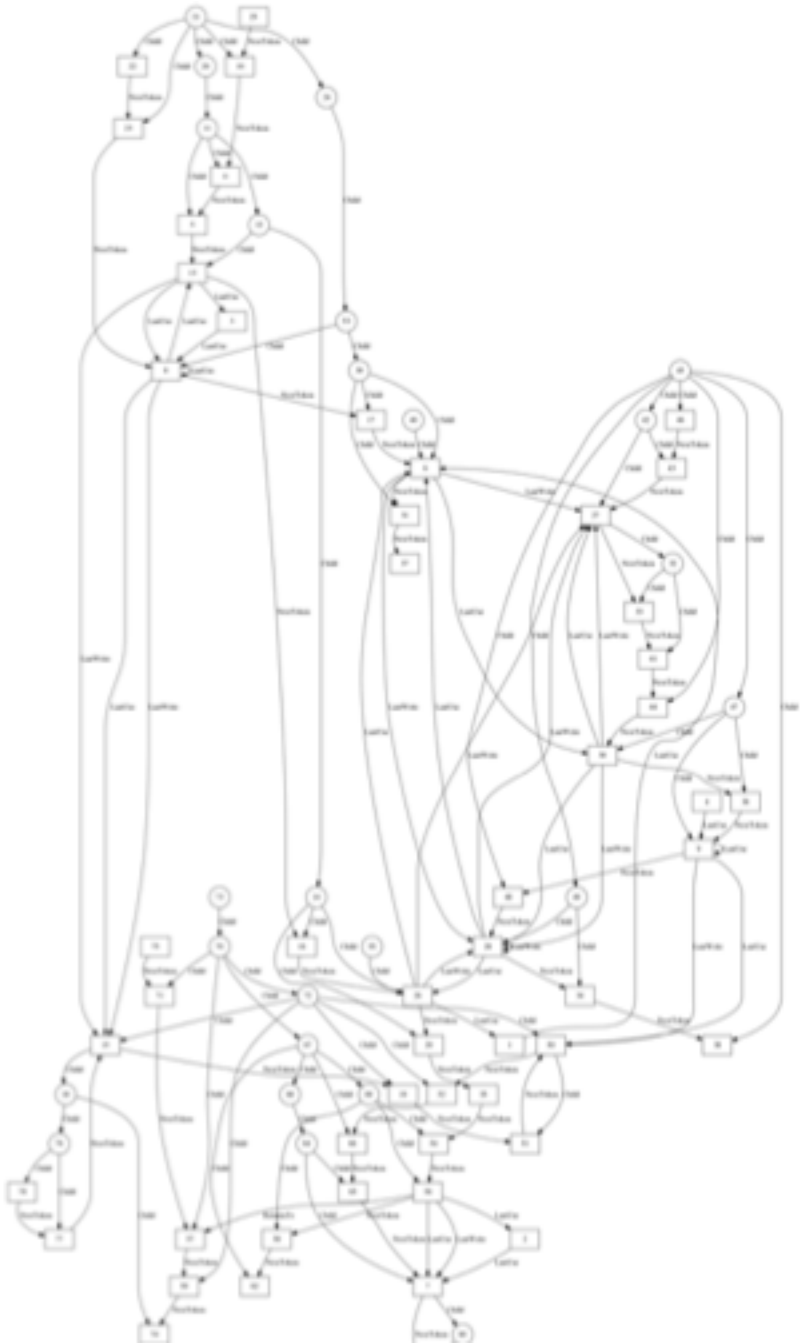
Programs as Graphs: Node Representation



Programs as Graphs



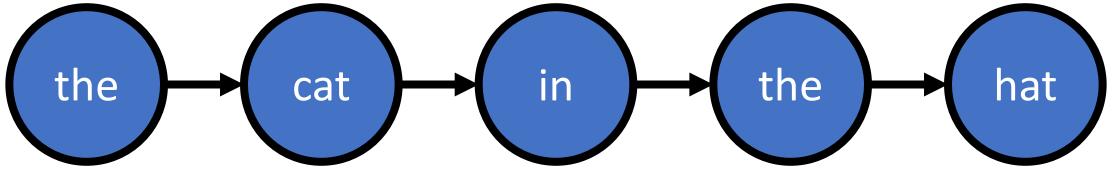
```
(x, y) = Foo();
while (x > 0)
x = x + y;
```



In practice: ~3000 nodes/graph, ~10000 edges/graph

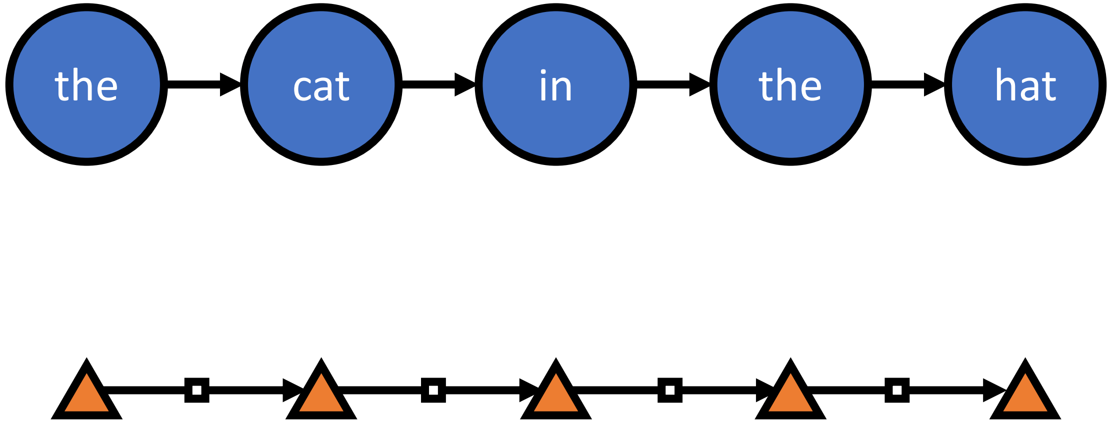
Graph Neural Networks: Extending RNNs

Graph Neural Networks: Extending RNNs



Chain structured data
(e.g. text)

Graph Neural Networks: Extending RNNs

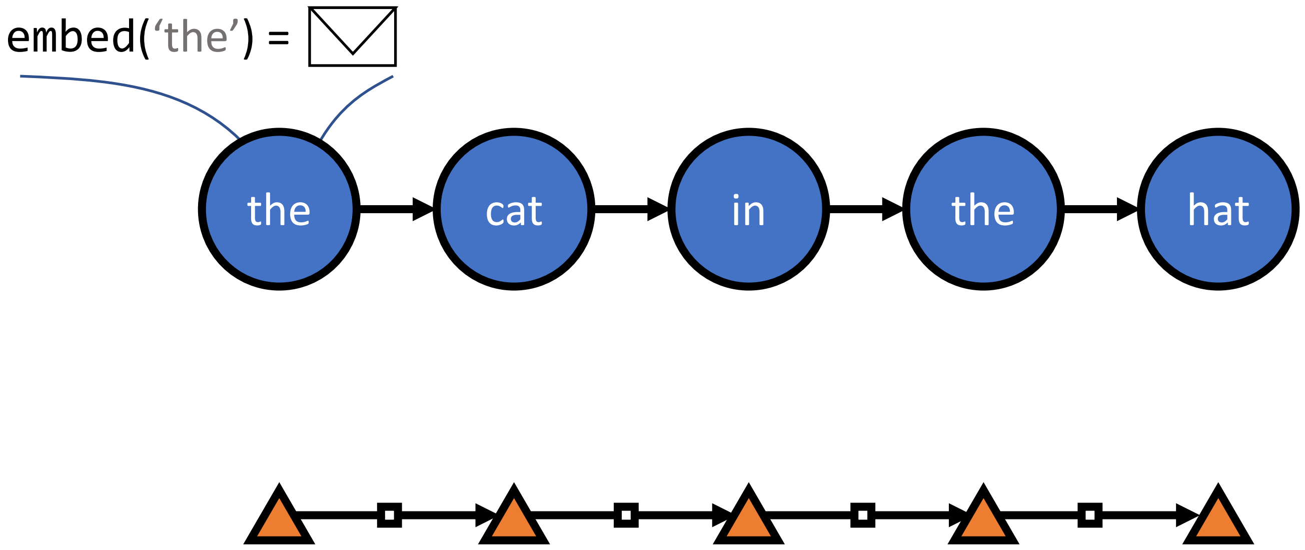


Chain structured data

(e.g. text)

▲ Recurrent unit

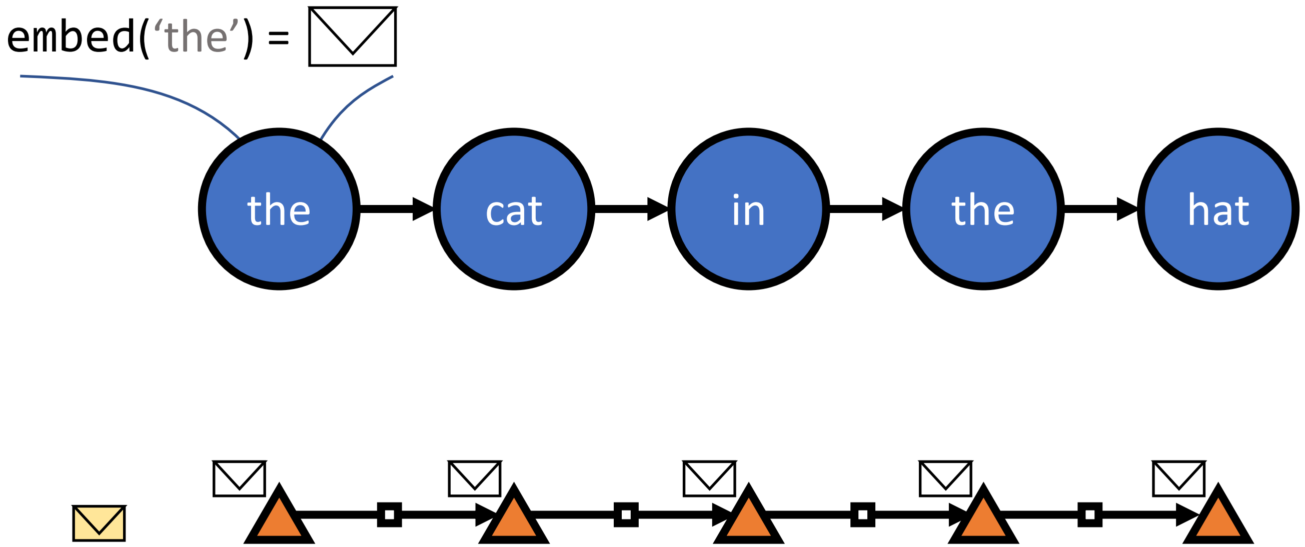
Graph Neural Networks: Extending RNNs



Chain structured data
(e.g. text)

▲ Recurrent unit

Graph Neural Networks: Extending RNNs

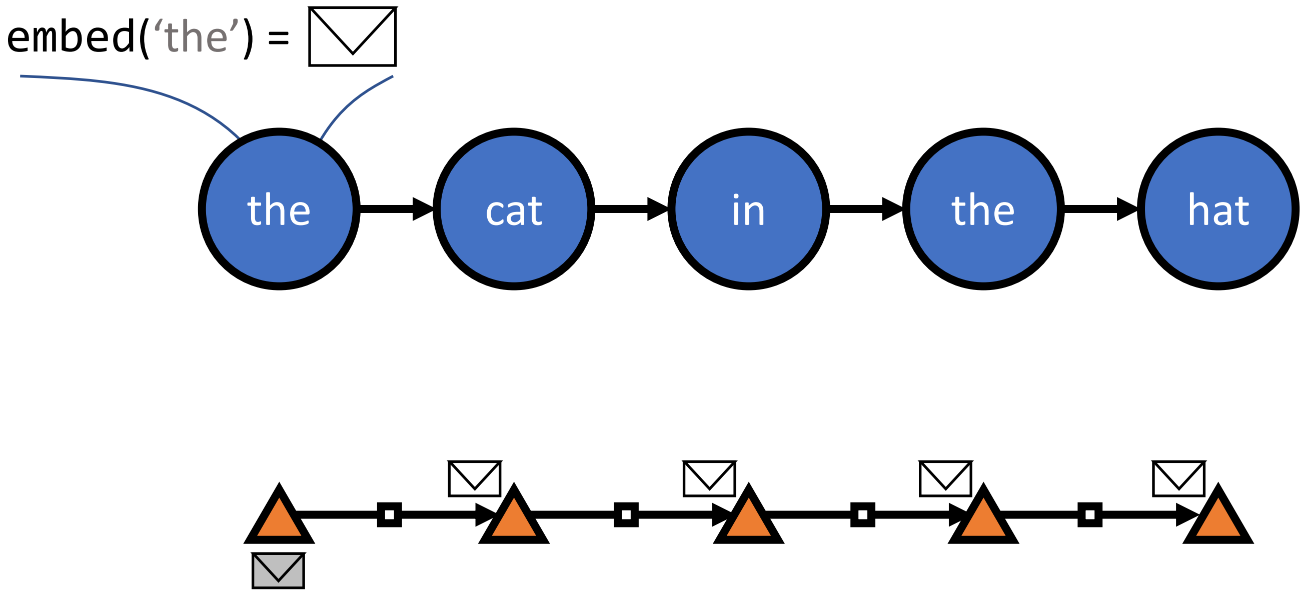


Chain structured data

(e.g. text)

▲ Recurrent unit

Graph Neural Networks: Extending RNNs

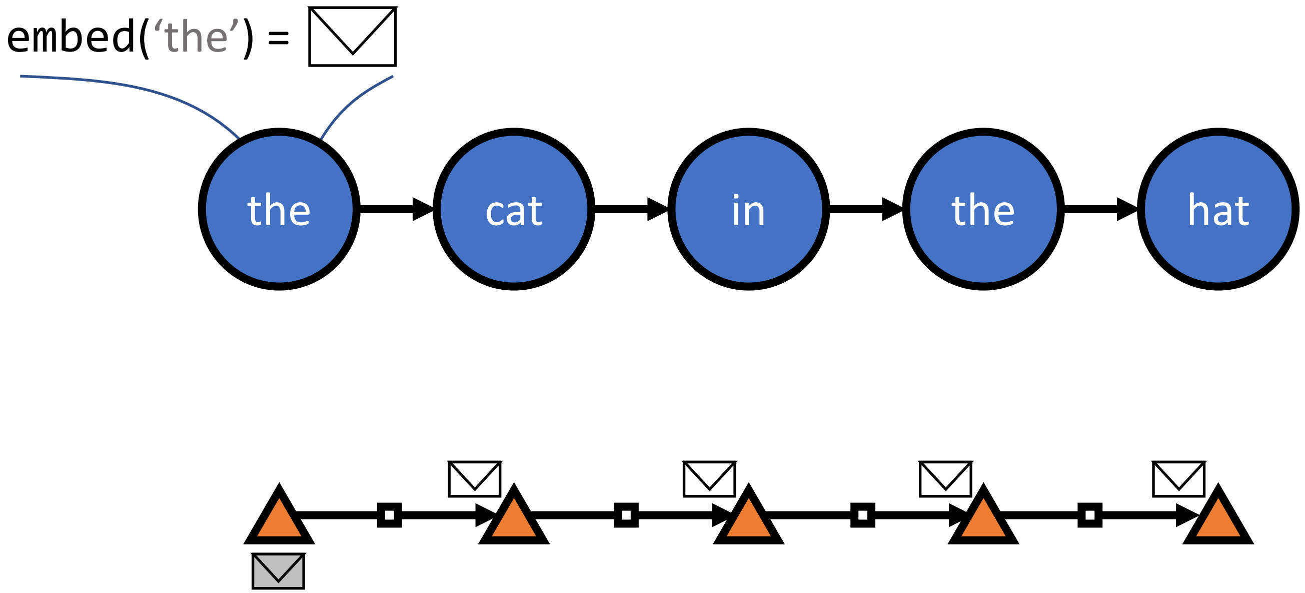


Chain structured data

(e.g. text)

▲ Recurrent unit

Graph Neural Networks: Extending RNNs

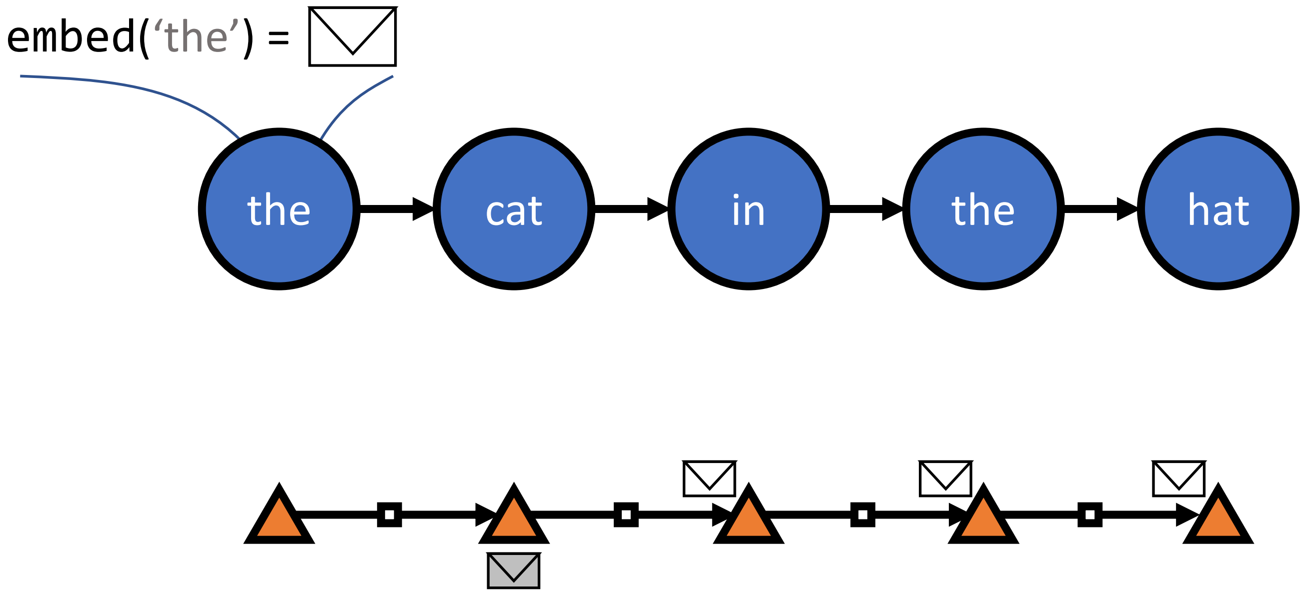


Chain structured data

(e.g. text)

▲ Recurrent unit

Graph Neural Networks: Extending RNNs

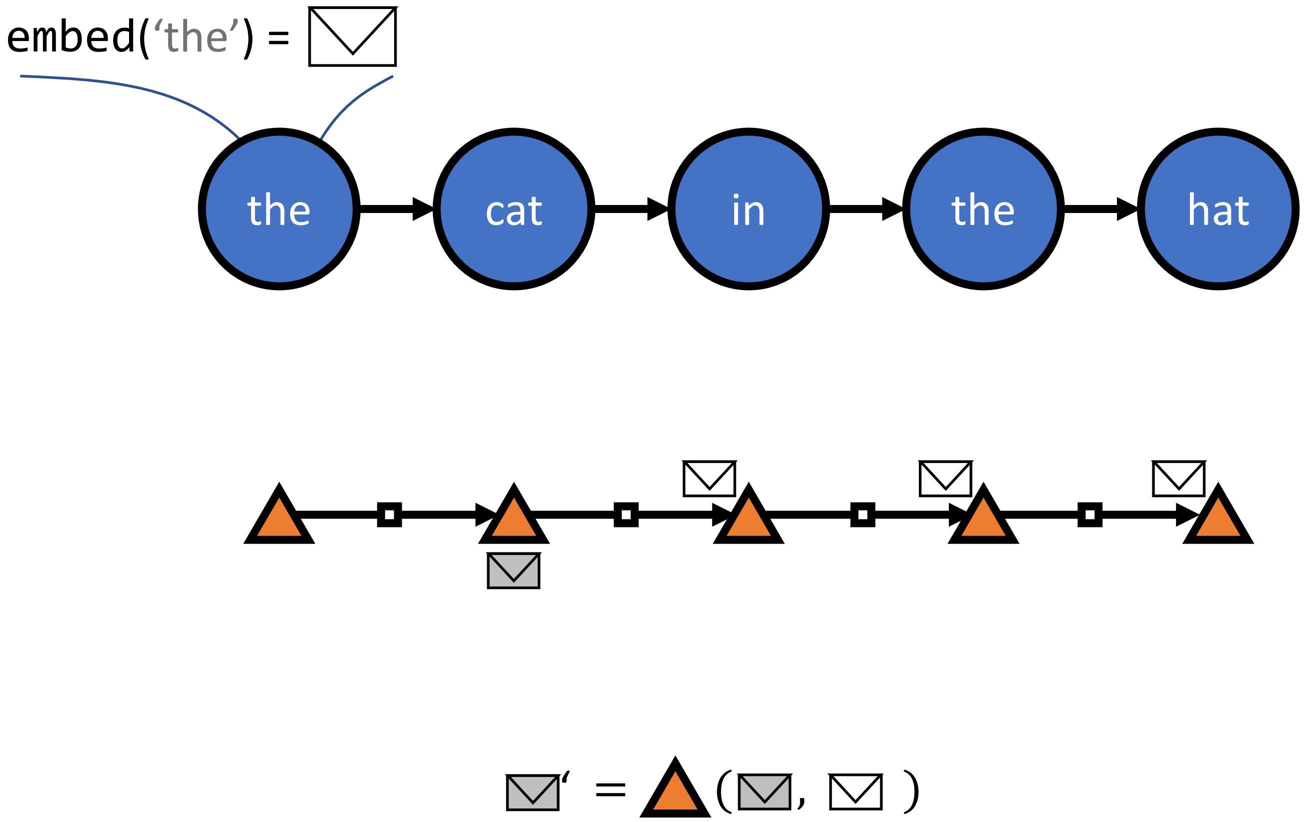


Chain structured data

(e.g. text)

▲ Recurrent unit

Graph Neural Networks: Extending RNNs

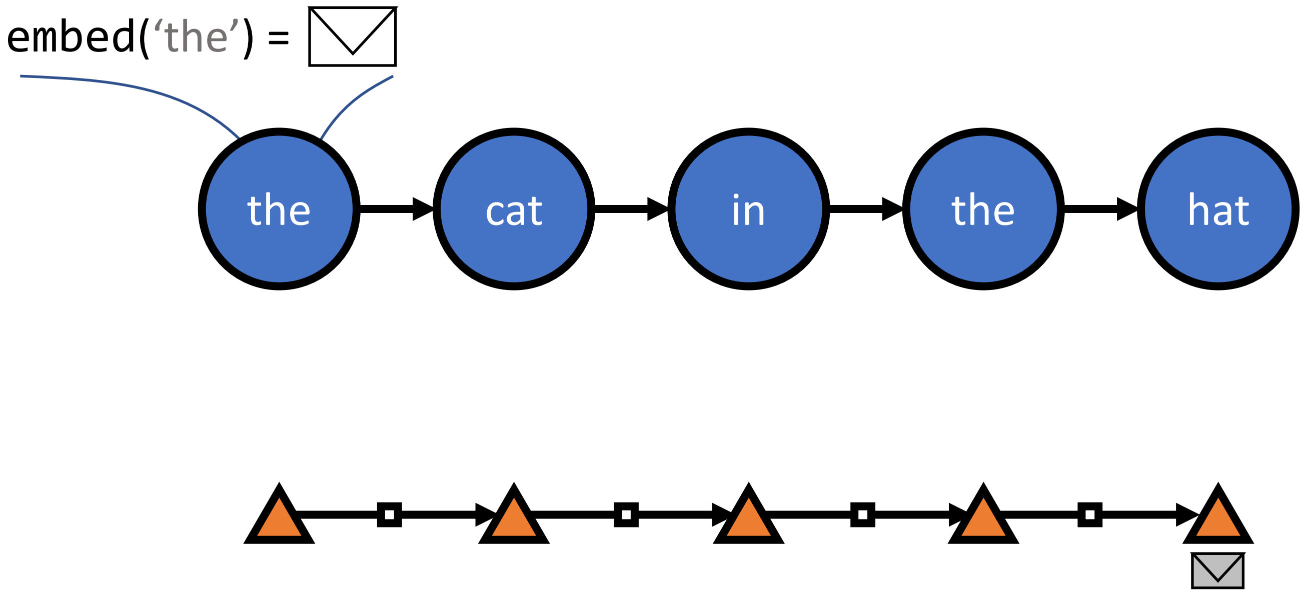


Chain structured data

(e.g. text)

▲ Recurrent unit

Graph Neural Networks: Extending RNNs

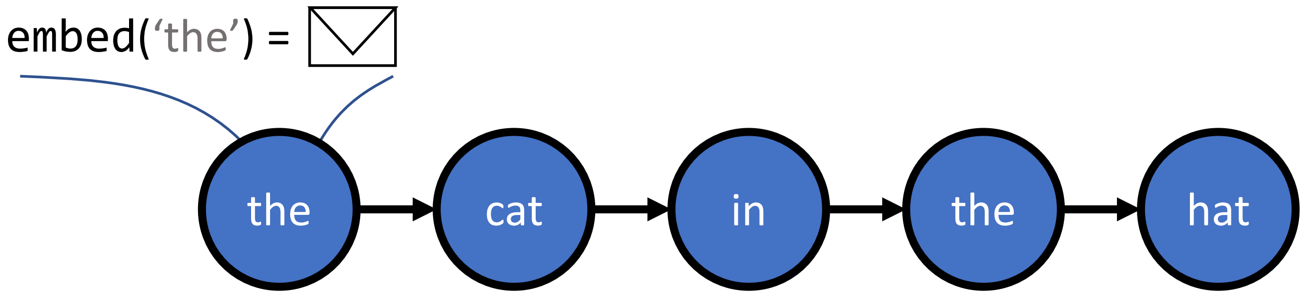


Chain structured data
(e.g. text)

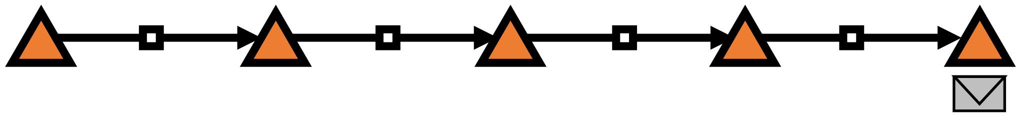
▲ Recurrent unit

✉' = ▲(✉, ✉)

Graph Neural Networks: Extending RNNs



Chain structured data
(e.g. text)

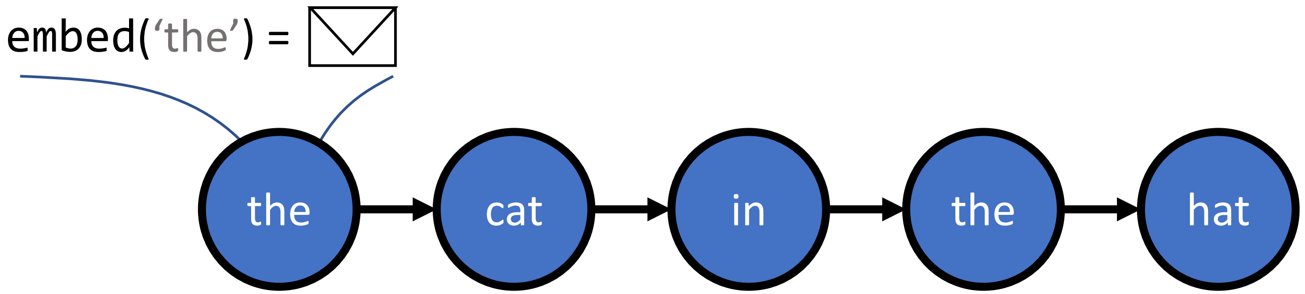


▲ Recurrent unit

```
let recurrent_unit state input = …     in
foldl recurrent_unit init_state seq
```

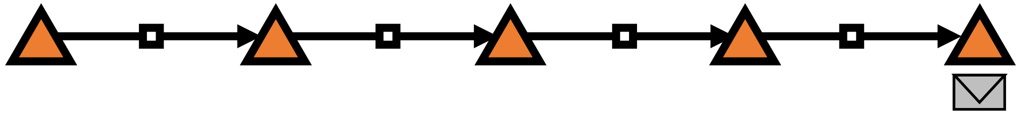
✉‘ = ▲(✉, ✉)

Graph Neural Networks: Extending RNNs



Chain structured data

(e.g. text)

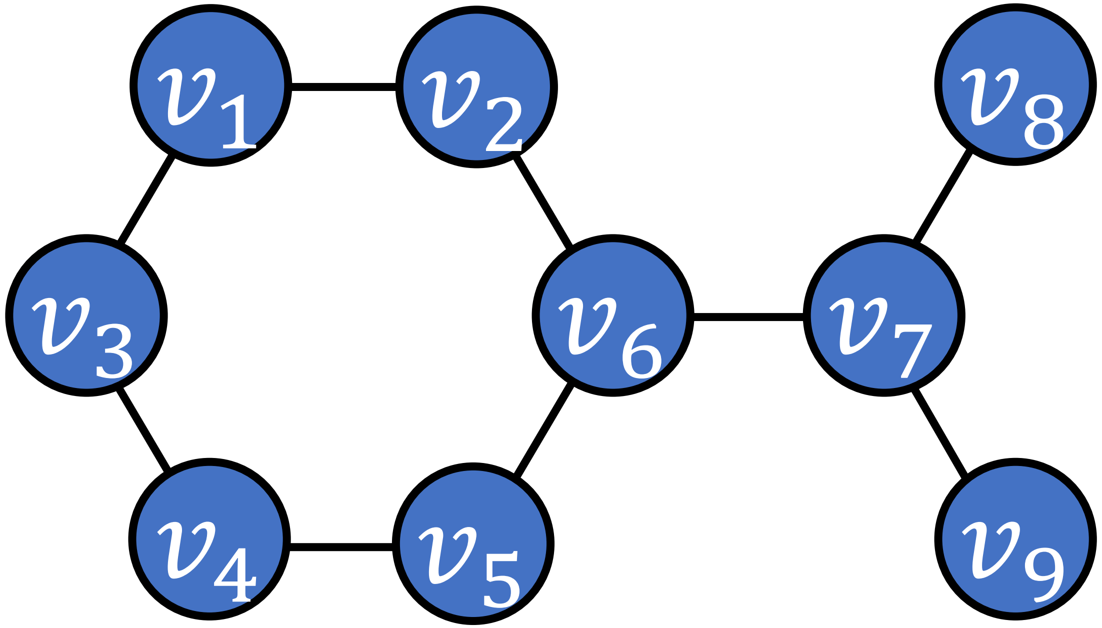


▲ Recurrent unit

✉' = ▲(✉, ✉)

```
let recurrent_unit state input = …     in
foldl recurrent_unit init_state seq

let      ▲      ✉      ✉      = …     in

foldl       ▲       ✉      [ ✉ , …, ✉ ]
```

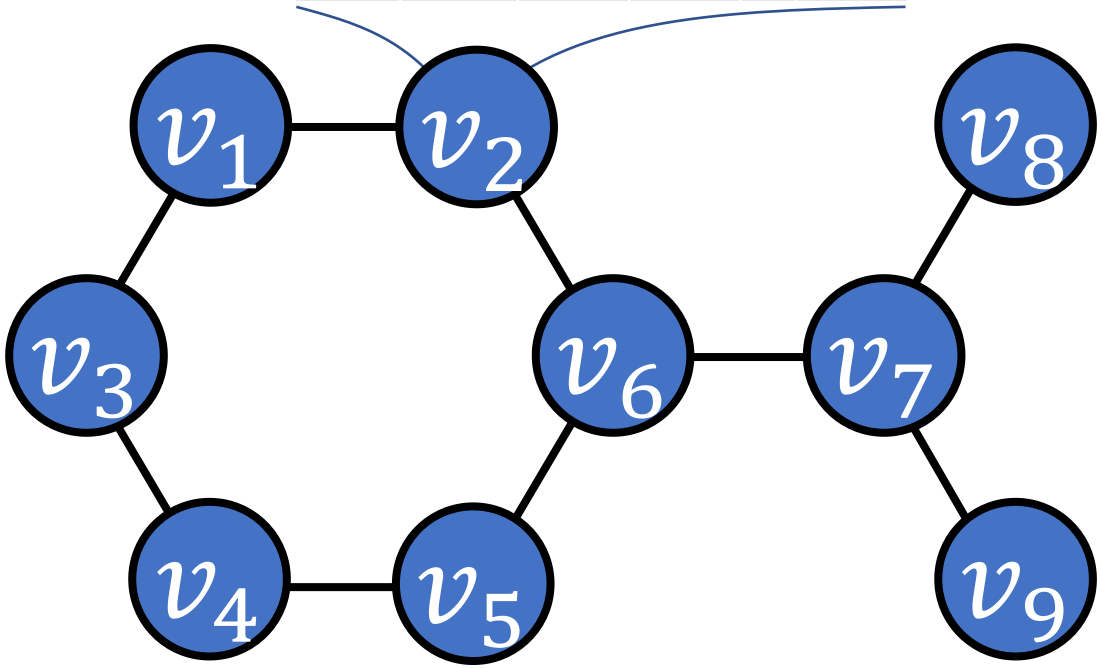
Graph Neural Networks: States



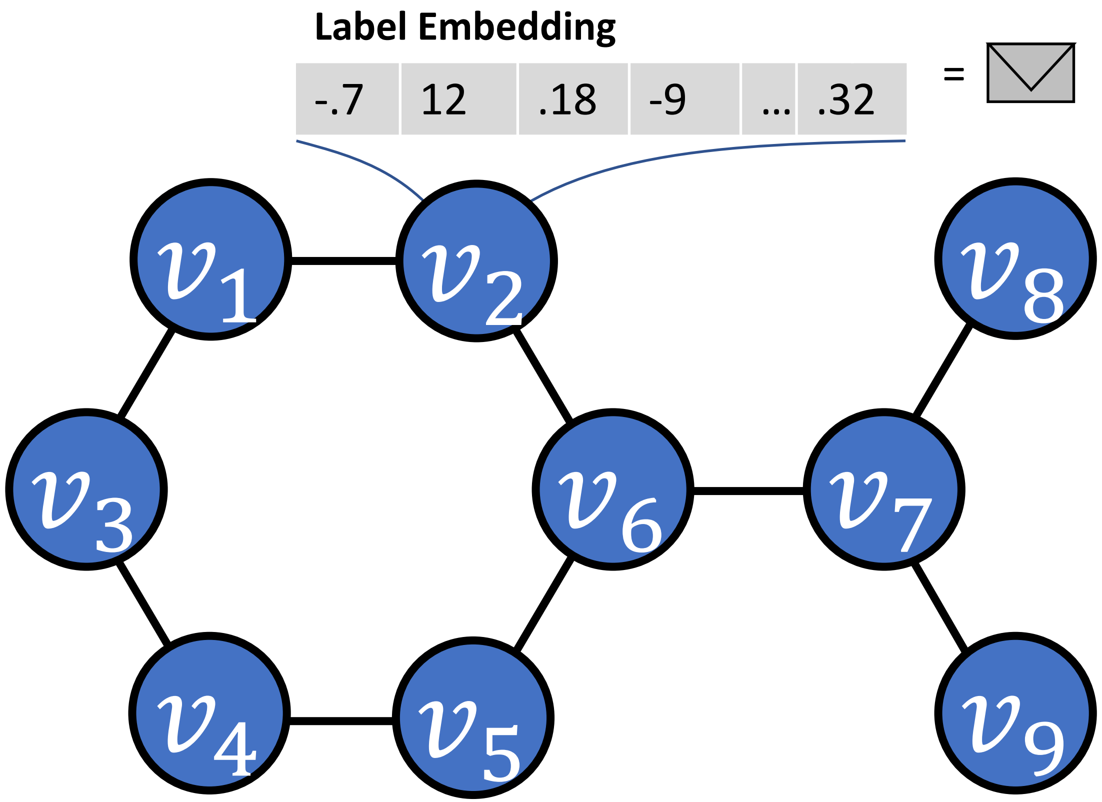
Graph Neural Networks: States

Label Embedding

-.7	12	.18	-9	…	.32



Graph Neural Networks: States

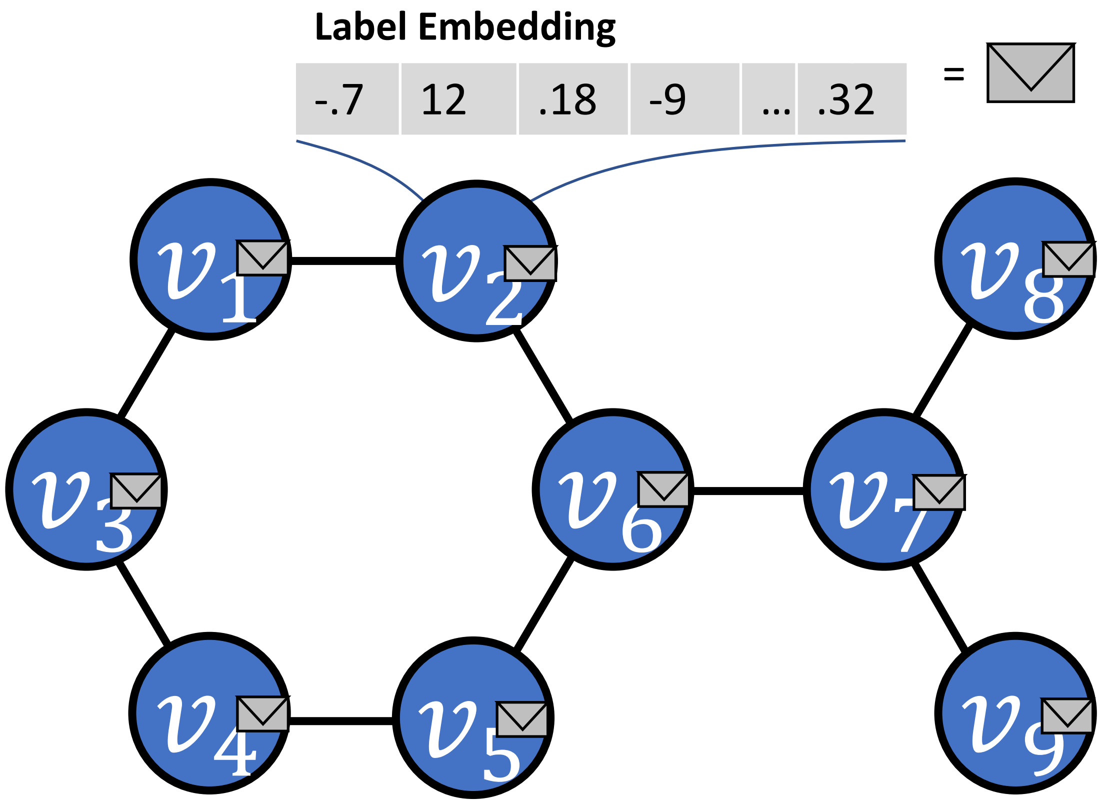


Label Embedding

-.7	12	.18	-9	...	.32

=

Graph Neural Networks: States

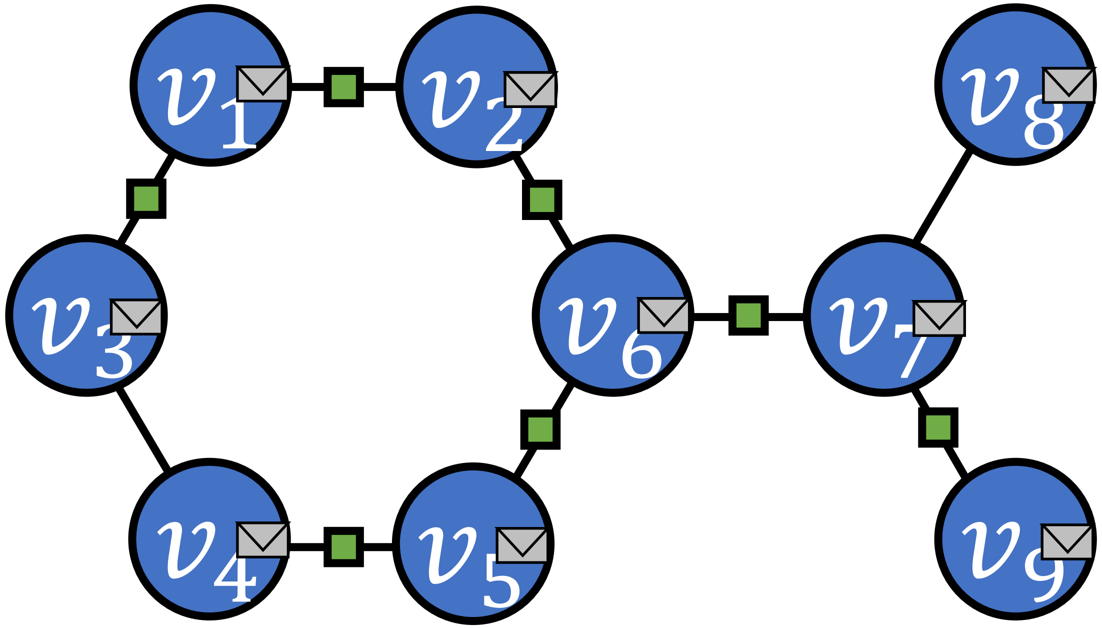


Label Embedding

-.7	12	.18	-9	...	.32

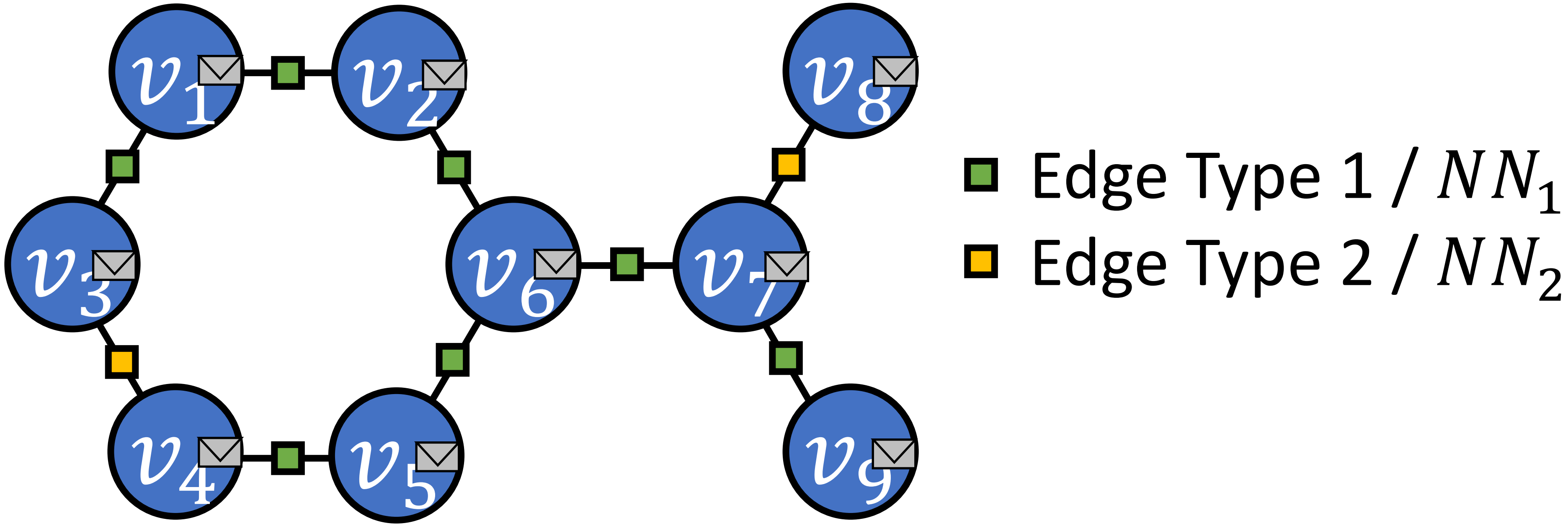
=

Graph Neural Networks: States



🟩 Edge Type 1 / NN_1

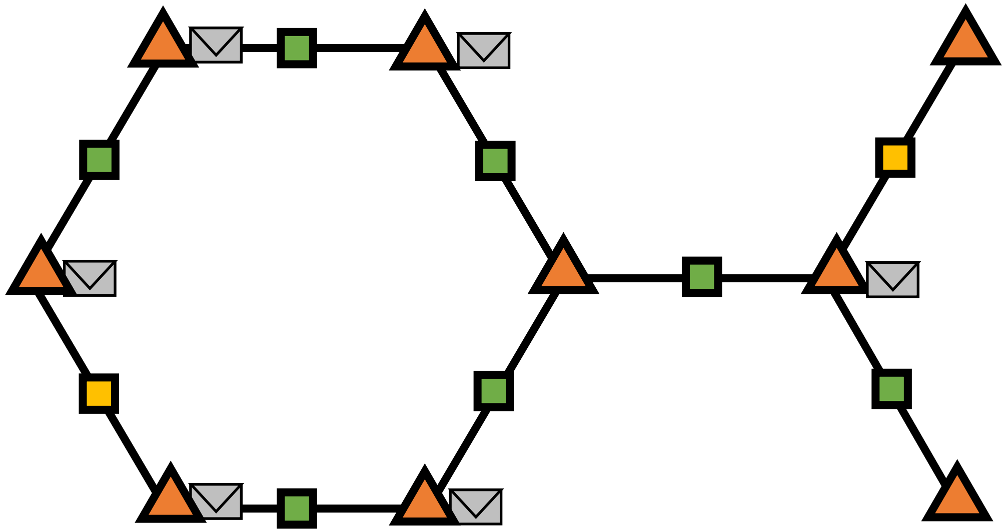
Graph Neural Networks: States



🟩 Edge Type 1 / NN_1

🟨 Edge Type 2 / NN_2

Graph Neural Networks: Propagation

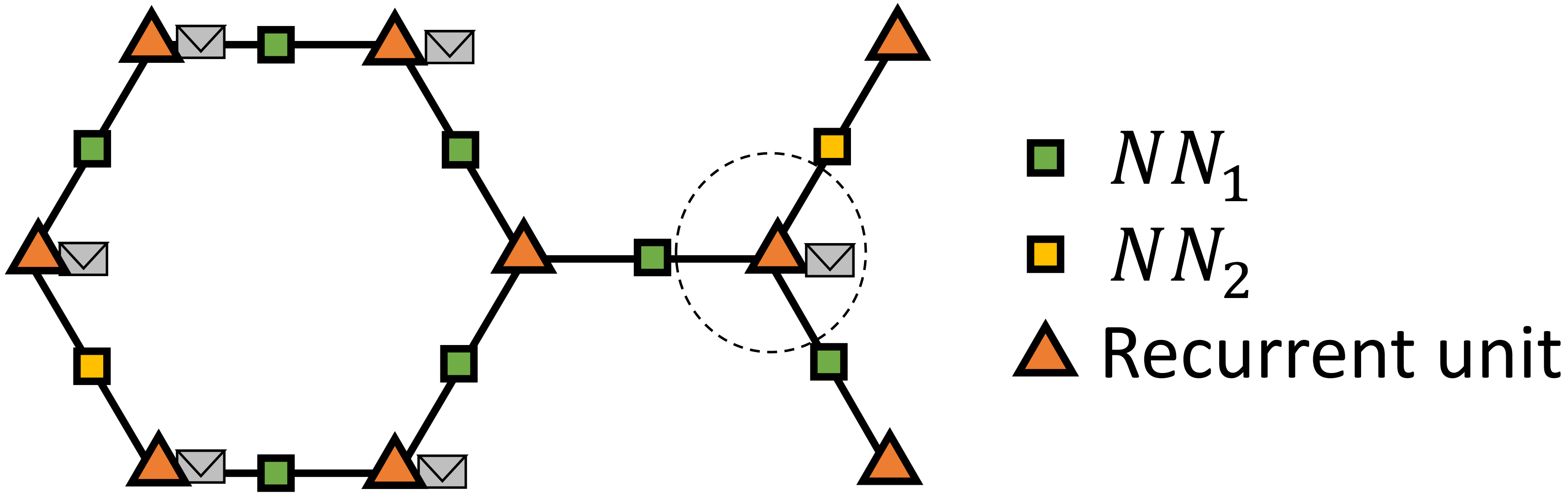


🟩 NN_1

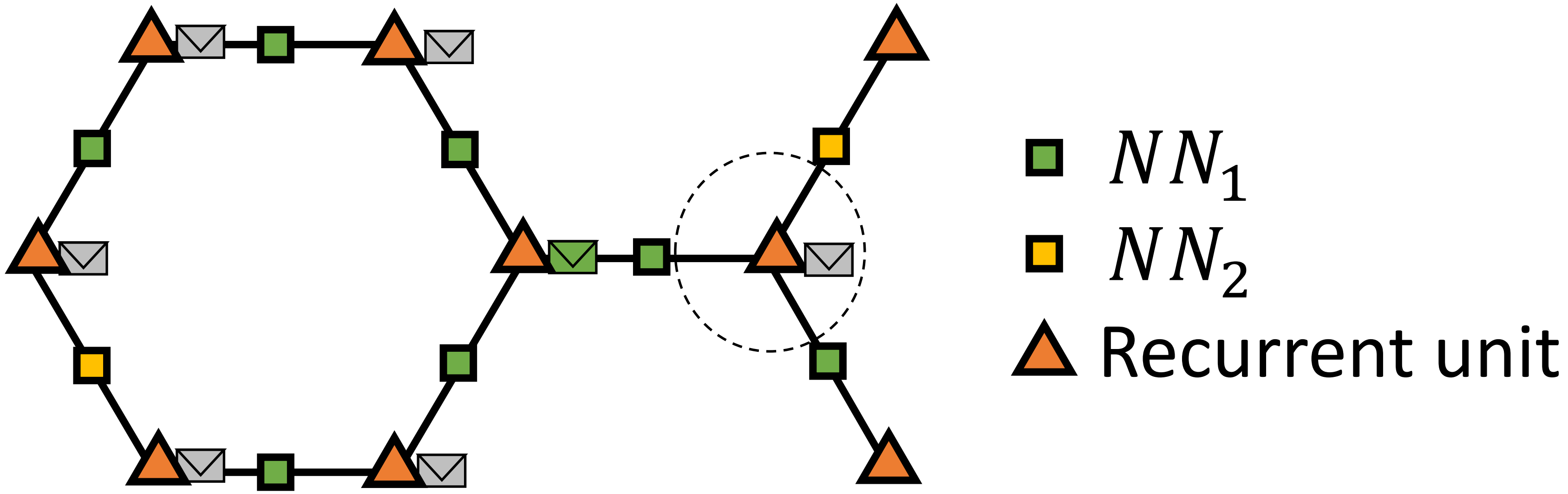
🟨 NN_2

🔺 Recurrent unit

Graph Neural Networks: Propagation



Graph Neural Networks: Propagation

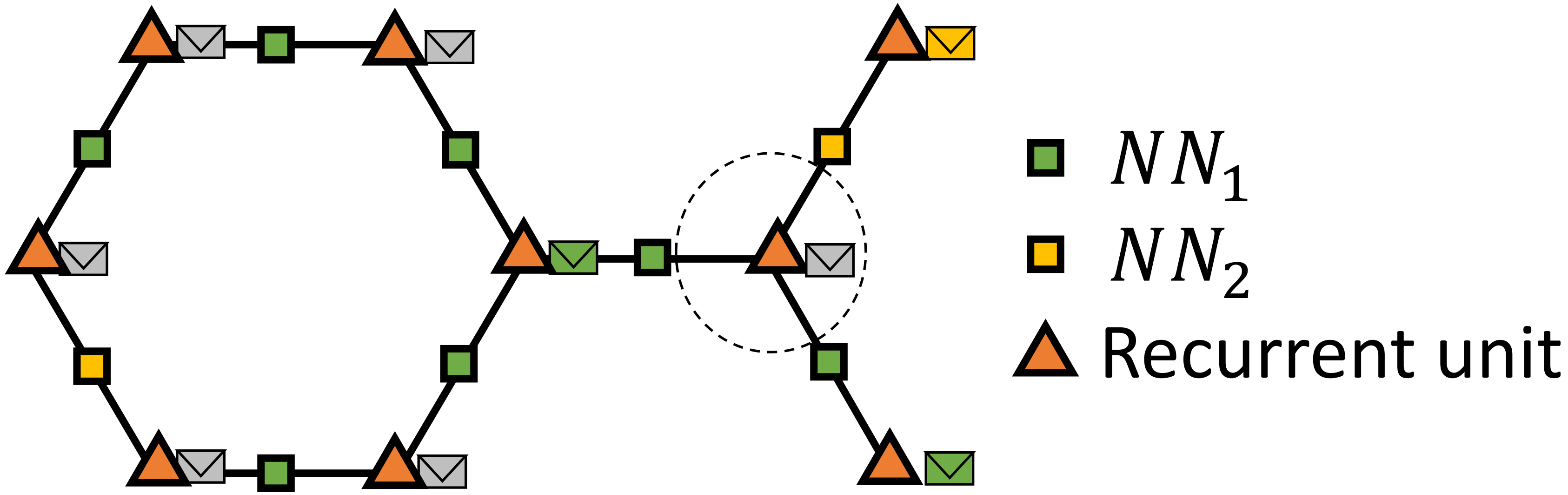


🟩 NN_1

🟨 NN_2

△ Recurrent unit

Graph Neural Networks: Propagation

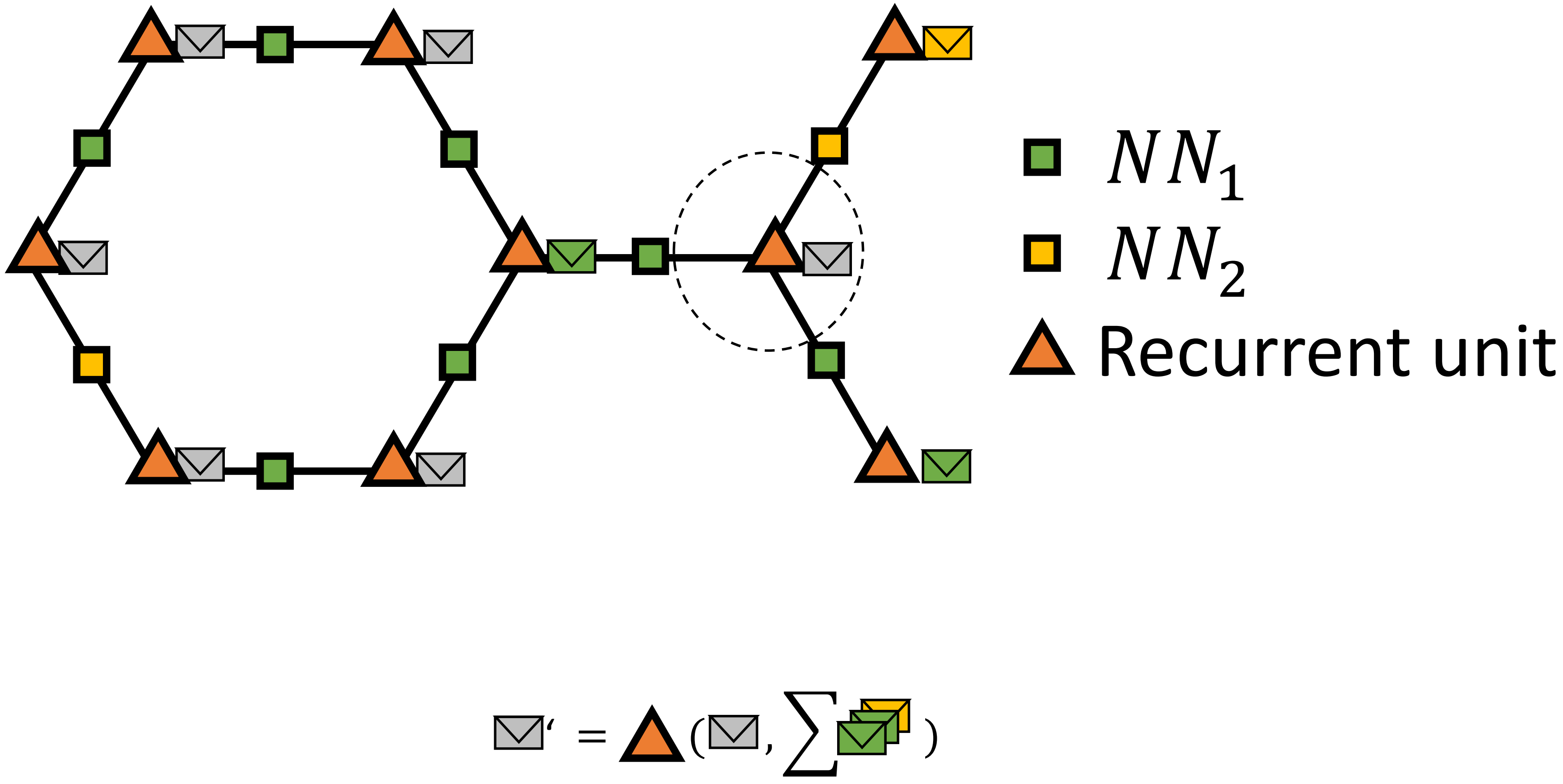


🟩 NN_1

🟨 NN_2

🔺 Recurrent unit

Graph Neural Networks: Propagation

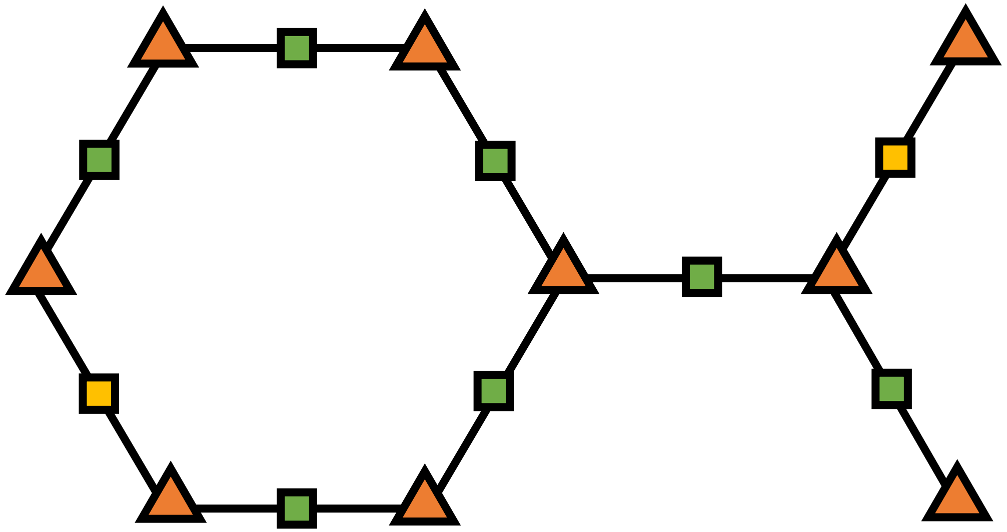


🟩 NN_1

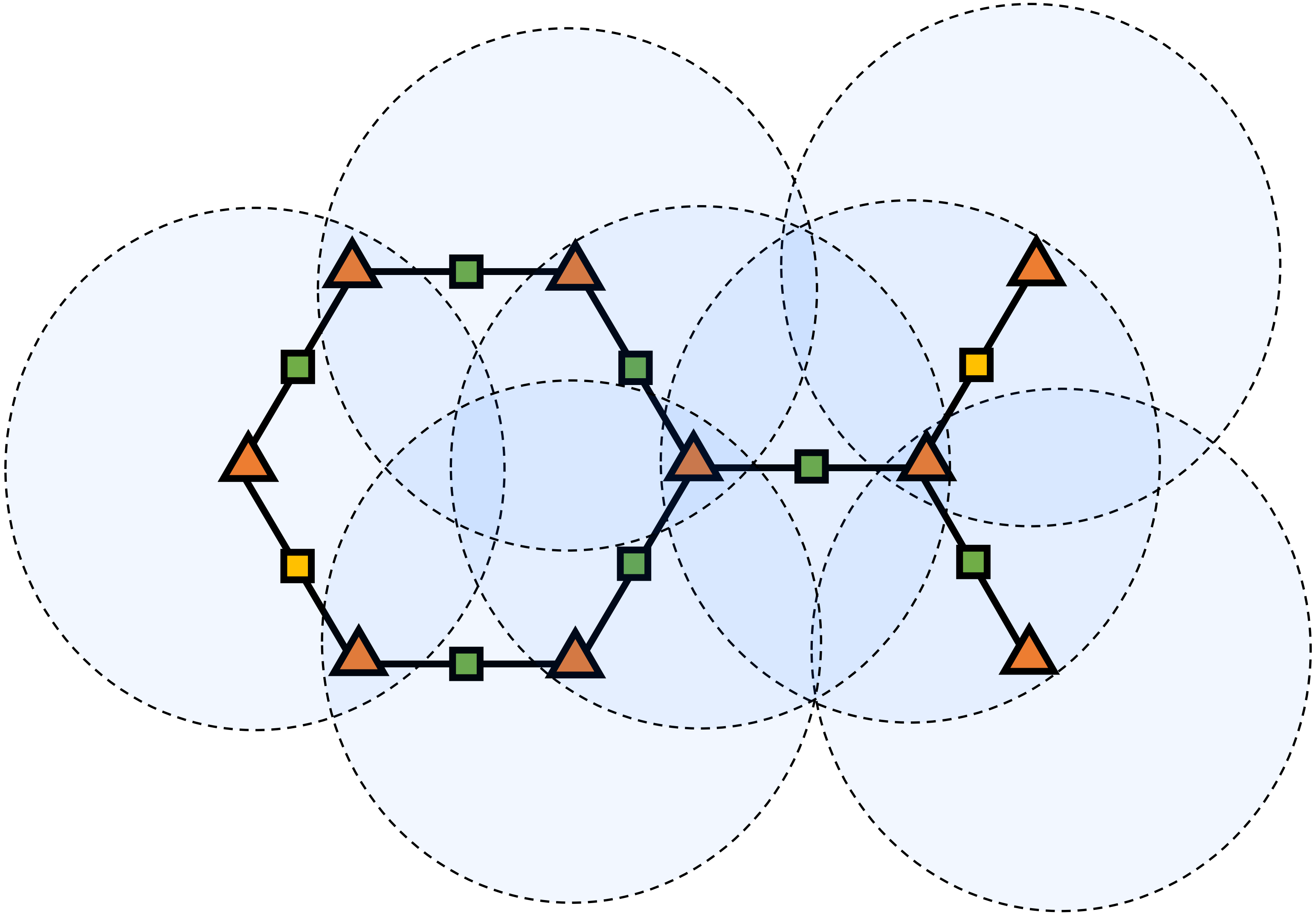
🟨 NN_2

▲ Recurrent unit

Graph Neural Networks: Unrolling



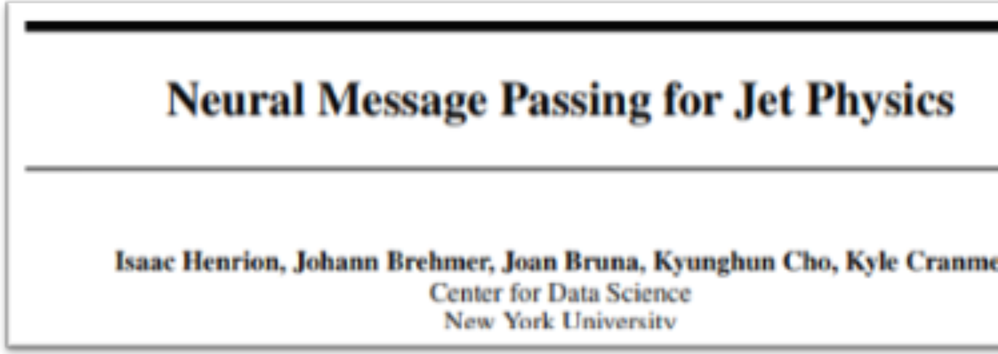
Graph Neural Networks: Unrolling



Graph Neural Networks: Uses

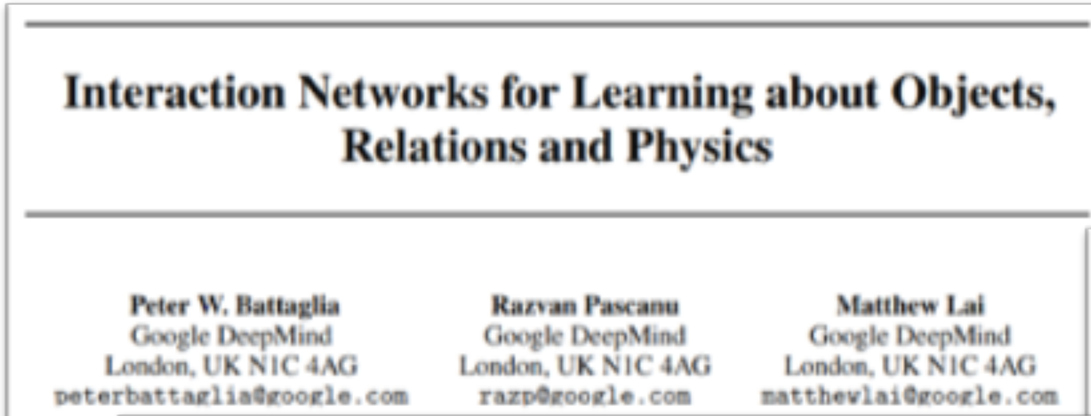
Graph Neural Networks: Uses

Gated Graph Sequence Neural Networks. In ICLR'16.



Neural Message Passing for Jet Physics

Isaac Henrion, Johann Brehmer, Joan Bruna, Kyunghun Cho, Kyle Cranme
Center for Data Science
New York University

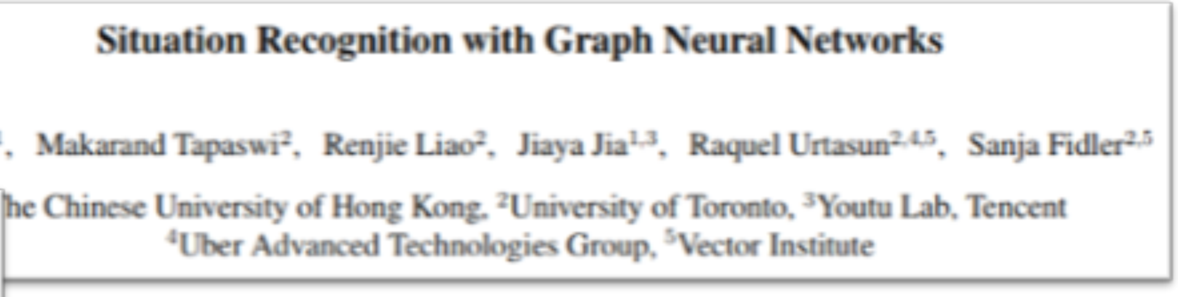


Interaction Networks for Learning about Objects, Relations and Physics

Peter W. Battaglia
Google DeepMind
London, UK N1C 4AG
peterbattaglia@google.com

Razvan Pascanu
Google DeepMind
London, UK N1C 4AG
razp@google.com

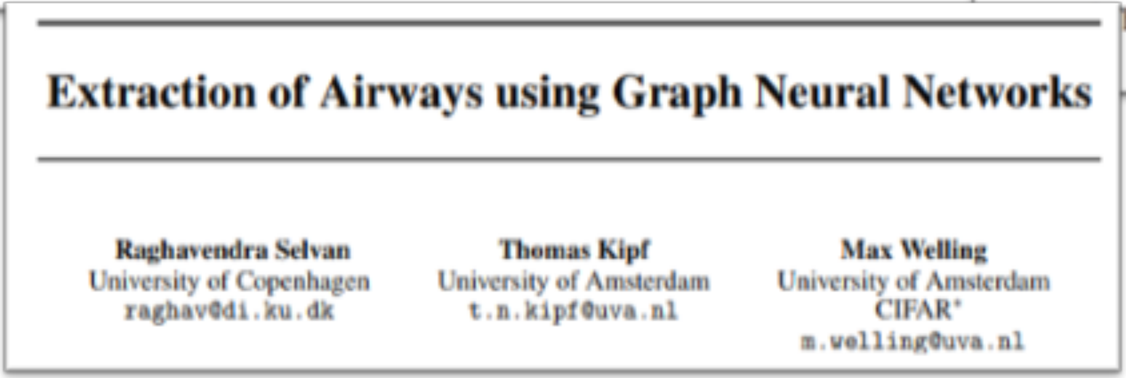
Matthew Lai
Google DeepMind
London, UK N1C 4AG
matthewlai@google.com



Situation Recognition with Graph Neural Networks

Ruiyu Li[1], Makarand Tapaswi[2], Renjie Liao[2], Jiaya Jia[1,3], Raquel Urtasun[2,4,5], Sanja Fidler[2,5]

he Chinese University of Hong Kong, [2]University of Toronto, [3]Youtu Lab, Tencent
[4]Uber Advanced Technologies Group, [5]Vector Institute



Extraction of Airways using Graph Neural Networks

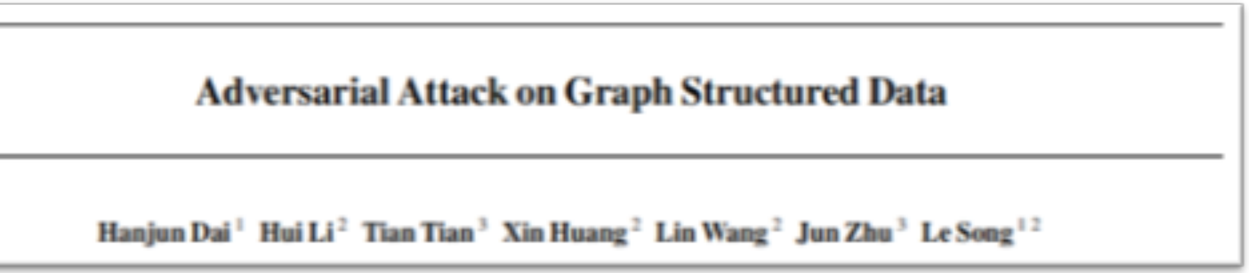
Raghavendra Selvan
University of Copenhagen
raghav@di.ku.dk

Thomas Kipf
University of Amsterdam
t.n.kipf@uva.nl

Max Welling
University of Amsterdam
CIFAR*
m.welling@uva.nl

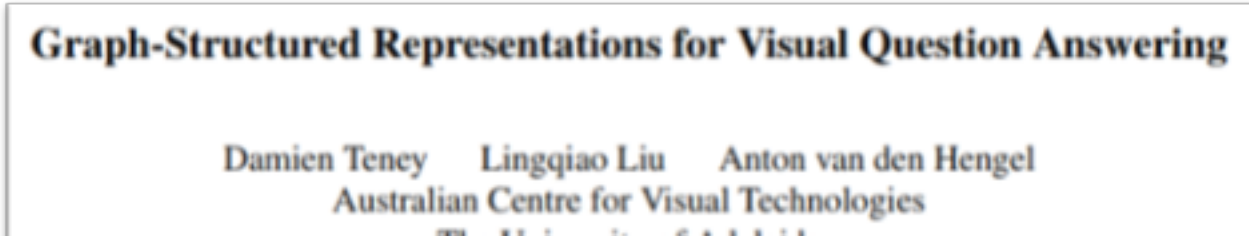
Learning to Verify the Heap

Marc Brockschmidt[1], Yuxin Chen[2], Byron Cook[3], Pushmeet Kohli[1], Siddharth Krishna[4], Daniel Tarlow[1], and He Zhu[5]



Adversarial Attack on Graph Structured Data

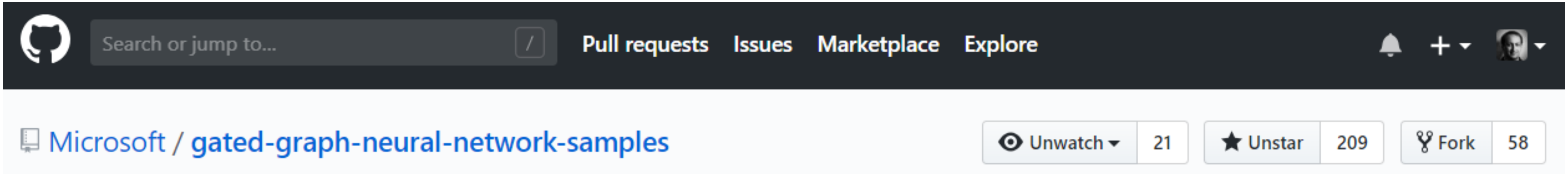
Hanjun Dai[1] Hui Li[2] Tian Tian[3] Xin Huang[2] Lin Wang[2] Jun Zhu[3] Le Song[1 2]



Graph-Structured Representations for Visual Question Answering

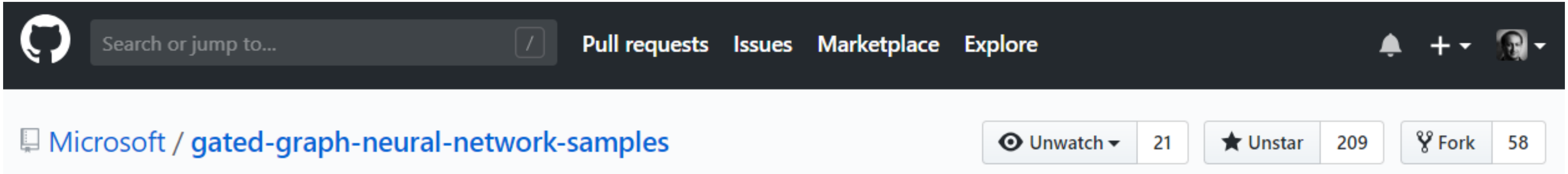
Damien Teney Lingqiao Liu Anton van den Hengel
Australian Centre for Visual Technologies

Graph Neural Networks: Implementation



Microsoft / gated-graph-neural-network-samples

Graph Neural Networks: Implementation



Train Performance:

On Titan X: 250 000 nodes/s (80 graphs/s)

On V100: 750 000 nodes/s (250 graphs/s)

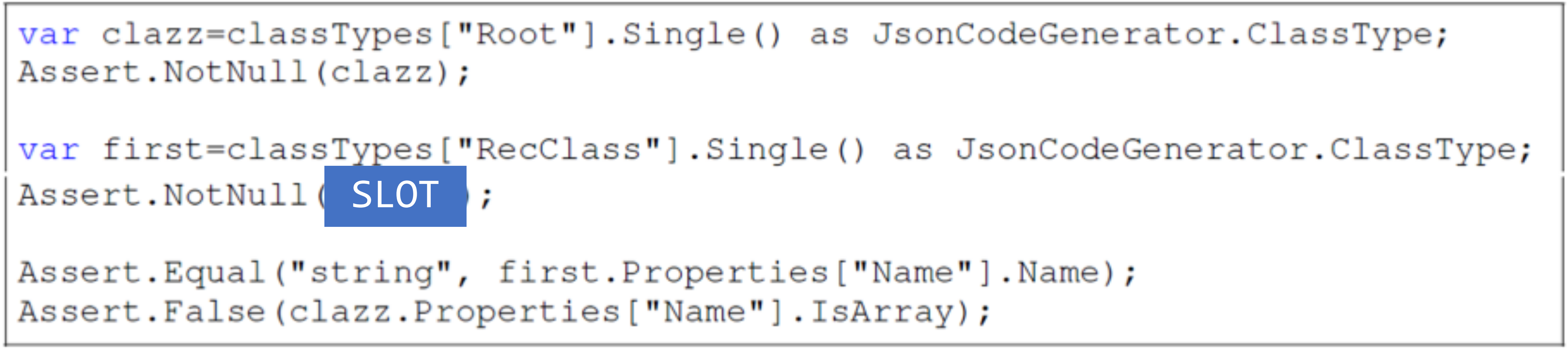
Test Performance:

On Titan X: 660 000 nodes/s (220 graphs/s)

On V100: 1 350 000 nodes/s (450 graphs/s)

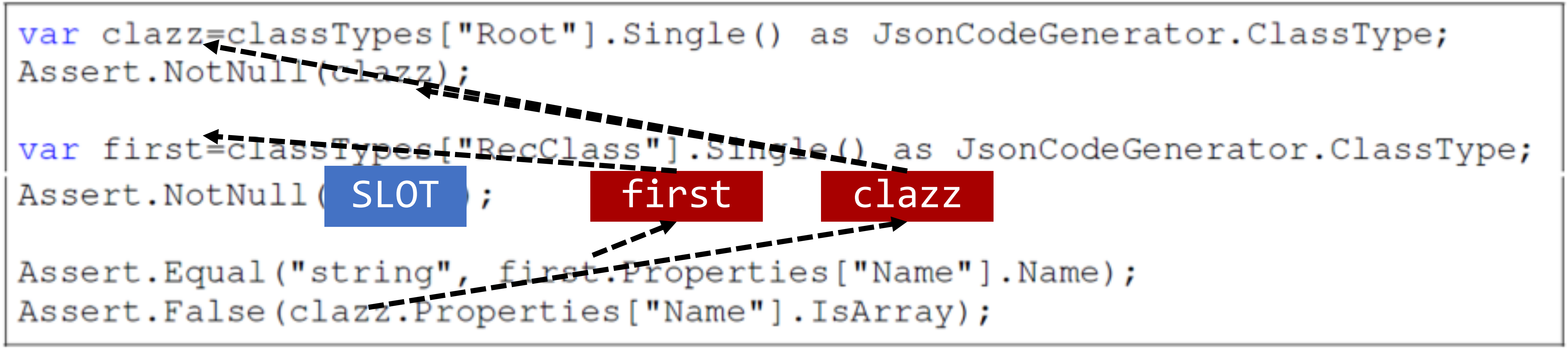
Detecting Variable Misuse

Detecting Variable Misuse



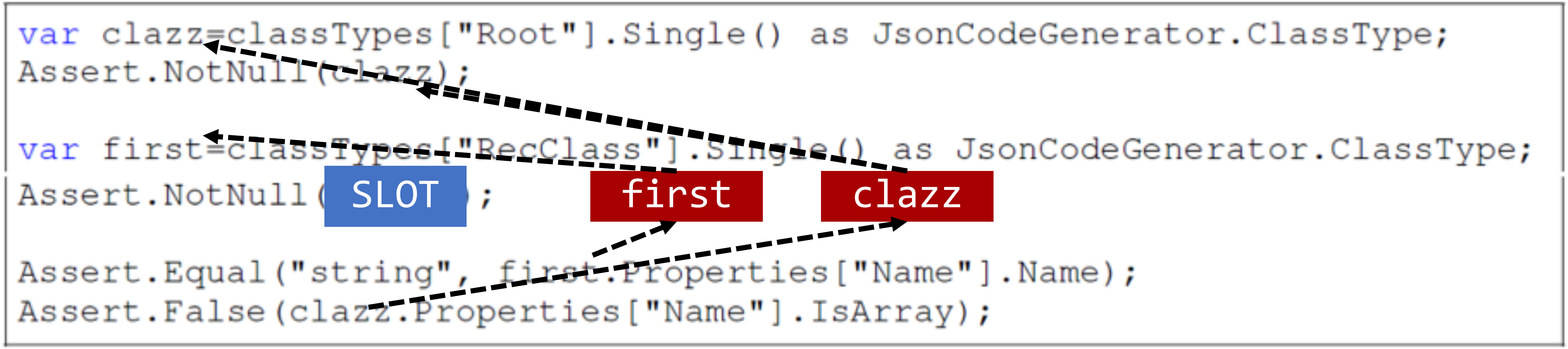
```
var clazz=classTypes["Root"].Single() as JsonCodeGenerator.ClassType;
Assert.NotNull(clazz);

var first=classTypes["RecClass"].Single() as JsonCodeGenerator.ClassType;
Assert.NotNull( SLOT );

Assert.Equal("string", first.Properties["Name"].Name);
Assert.False(clazz.Properties["Name"].IsArray);
```

Detecting Variable Misuse



```
var clazz=classTypes["Root"].Single() as JsonCodeGenerator.ClassType;
Assert.NotNull(clazz);

var first=classTypes["RecClass"].Single() as JsonCodeGenerator.ClassType;
Assert.NotNull( SLOT );

Assert.Equal("string", first.Properties["Name"].Name);
Assert.False(clazz.Properties["Name"].IsArray);
```

Detecting Variable Misuse



```
var clazz=classTypes["Root"].Single() as JsonCodeGenerator.ClassType;
Assert.NotNull(clazz);

var first=classTypes["RecClass"].Single() as JsonCodeGenerator.ClassType;
Assert.NotNull(SLOT);

Assert.Equal("string", first.Properties["Name"].Name);
Assert.False(clazz.Properties["Name"].IsArray);
```

Objective: Given representation of SLOT, choose between “`first`” and “`clazz`”

Variable Naming: Quantitative Results

F1 (%)	Sequence	Seq.+Dataflow	Graph
Seen Projects	44.0	50.1	**65.8**
Unseen Projects	30.6	32.0	**62.0**

Seen Projects: 24 F/OSS C# projects (2060 kLOC): Used for train and test
Unseen Projects: 3 F/OSS C# projects (228 kLOC): Used only for test

Variable Misuse: Quantitative Results

Accuracy (%)	Sequence	Seq.+Dataflow	Graph
Seen Projects	50.0	73.7	**86.5**
Unseen Projects	28.9	60.2	**82.0**

Seen Projects: 24 F/OSS C# projects (2060 kLOC): Used for train and test
Unseen Projects: 3 F/OSS C# projects (228 kLOC): Used only for test
3.8 type-correct alternative variables per slot (median 3, σ= 2.6)

Variable Misuse: Quantitative Results

Accuracy (%)	**Sequence**	**Seq.+Dataflow**	**Graph**
Seen Projects	50.0	73.7	**86.5**
Unseen Projects	28.9	60.2	**82.0**
255 Proj. – Seen	-	-	**91.8**
255 Proj. – Unseen	-	-	**89.4**

Task: Extracting Best Practices

Objective: Given many commits, extract common kinds of changes

Task: Extracting Best Practices

Objective: Given many commits, extract common kinds of changes

```
emps.Where(e => m.ReportsTo == e.EmployeeID).FirstOrDefault();
                              ↓
emps.FirstOrDefault(e => m.ReportsTo == e.EmployeeID);
```

```
sources = sources == null ? new object[0] : sources.ToArray();
                              ↓
sources = sources?.ToArray() ?? new object[0];
```

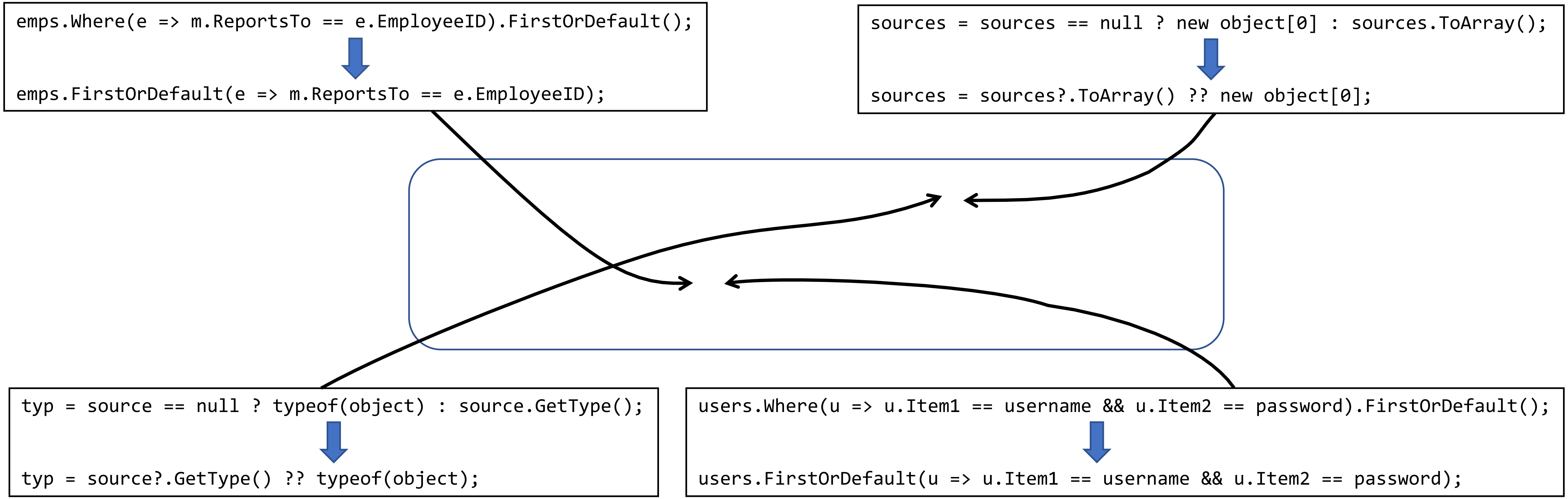
```
typ = source == null ? typeof(object) : source.GetType();
                           ↓
typ = source?.GetType() ?? typeof(object);
```

```
users.Where(u => u.Item1 == username && u.Item2 == password).FirstOrDefault();
                                    ↓
users.FirstOrDefault(u => u.Item1 == username && u.Item2 == password);
```

Task: Extracting Best Practices

Objective: Given many commits, extract common kinds of changes

Idea: Learn to embed similar diffs nearby in vector space (as in word2vec)



```
emps.Where(e => m.ReportsTo == e.EmployeeID).FirstOrDefault();

emps.FirstOrDefault(e => m.ReportsTo == e.EmployeeID);
```

```
sources = sources == null ? new object[0] : sources.ToArray();

sources = sources?.ToArray() ?? new object[0];
```

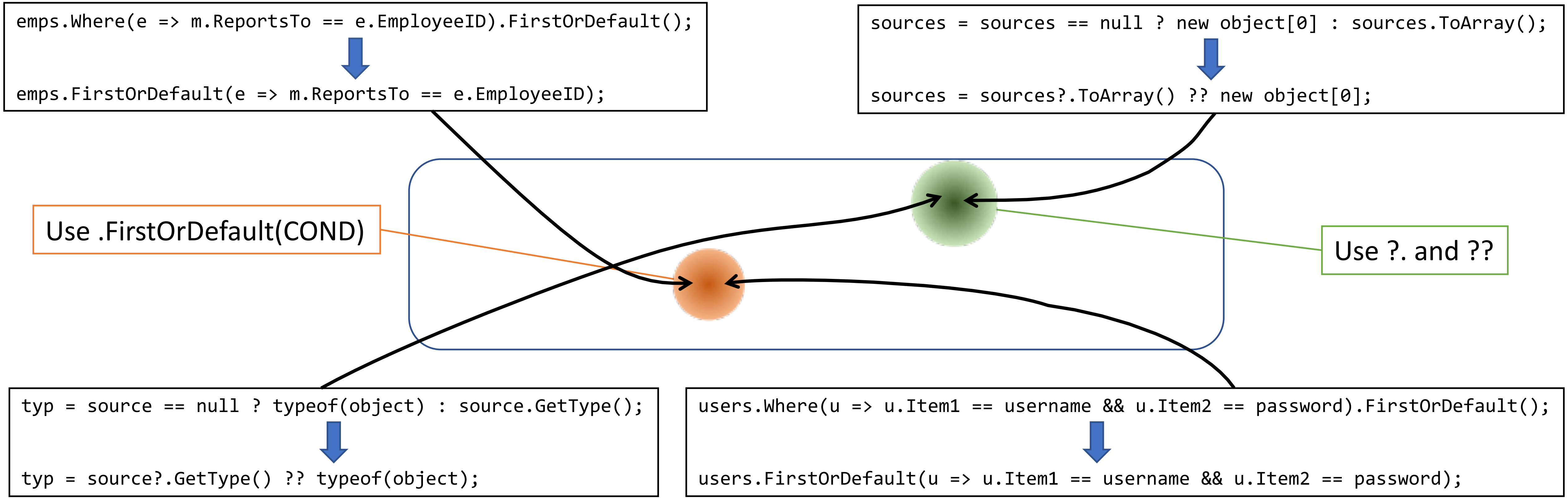
```
typ = source == null ? typeof(object) : source.GetType();

typ = source?.GetType() ?? typeof(object);
```

```
users.Where(u => u.Item1 == username && u.Item2 == password).FirstOrDefault();

users.FirstOrDefault(u => u.Item1 == username && u.Item2 == password);
```

Task: Extracting Best Practices

Objective: Given many commits, extract common kinds of changes

Idea: Learn to embed similar diffs nearby in vector space (as in word2vec)



```
emps.Where(e => m.ReportsTo == e.EmployeeID).FirstOrDefault();

emps.FirstOrDefault(e => m.ReportsTo == e.EmployeeID);
```

```
sources = sources == null ? new object[0] : sources.ToArray();

sources = sources?.ToArray() ?? new object[0];
```

Use .FirstOrDefault(COND)

Use ?. and ??

```
typ = source == null ? typeof(object) : source.GetType();

typ = source?.GetType() ?? typeof(object);
```

```
users.Where(u => u.Item1 == username && u.Item2 == password).FirstOrDefault();

users.FirstOrDefault(u => u.Item1 == username && u.Item2 == password);
```

Learning From Programs: Key Points

Insight: GNNs successful at learning with code semantics

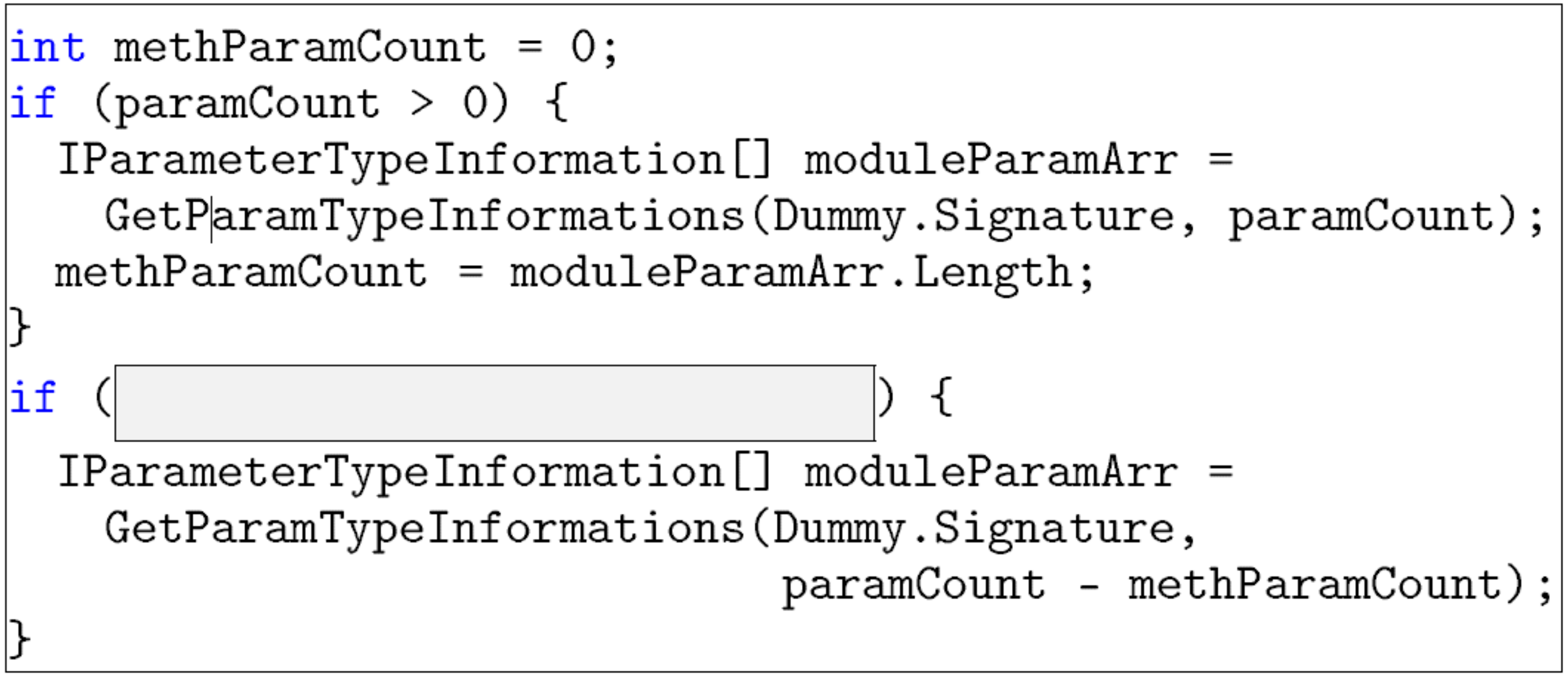
Outcomes:

- Machinery can be re-used for many tasks
- Learns “soft” rules from data, no rule definitions required
- Found number of bugs in mature code

Generating Programs

Task: Filling in Blanks

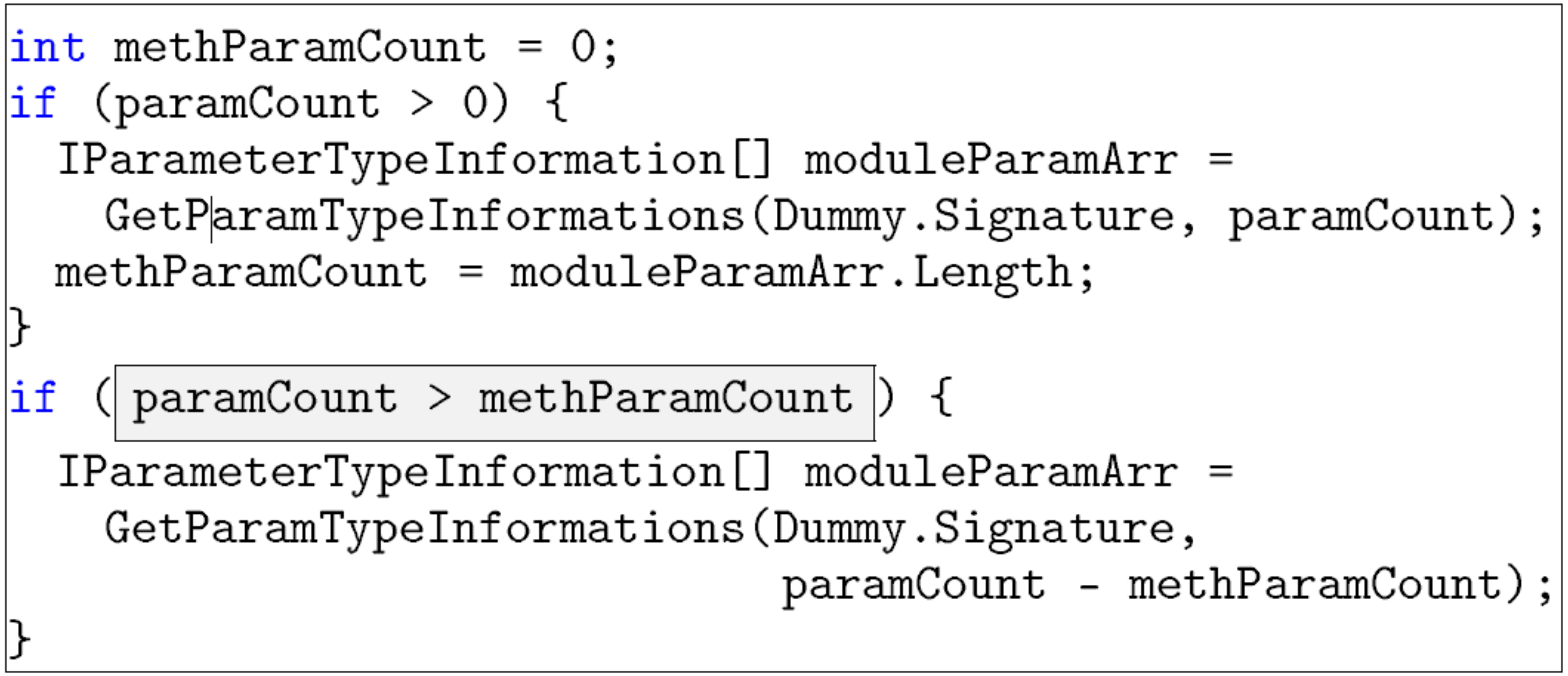
Given location in program code, generate expression:



```
int methParamCount = 0;
if (paramCount > 0) {
  IParameterTypeInformation[] moduleParamArr =
    GetParamTypeInformations(Dummy.Signature, paramCount);
  methParamCount = moduleParamArr.Length;
}
if (                              ) {
  IParameterTypeInformation[] moduleParamArr =
    GetParamTypeInformations(Dummy.Signature,
                             paramCount - methParamCount);
}
```

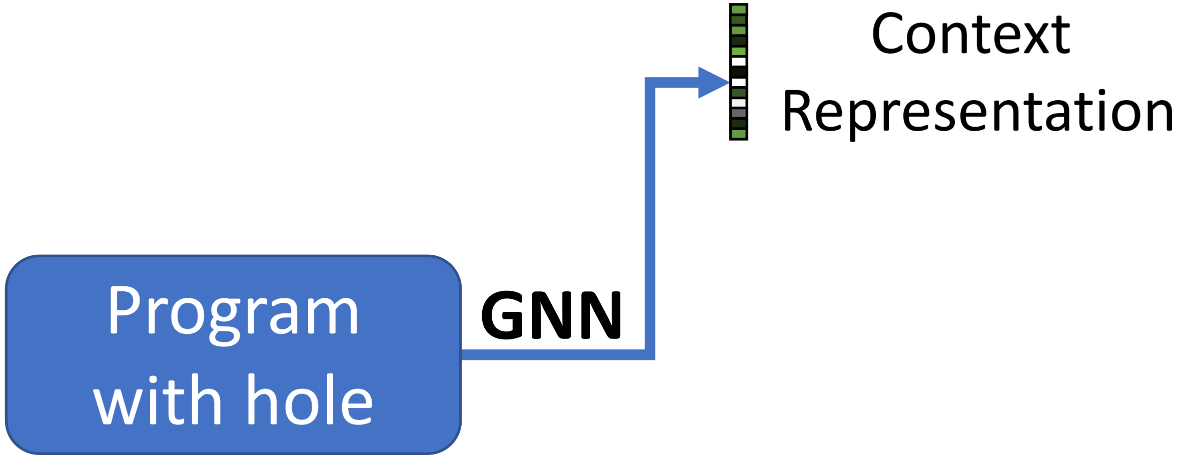
Task: Filling in Blanks

Given location in program code, generate expression:

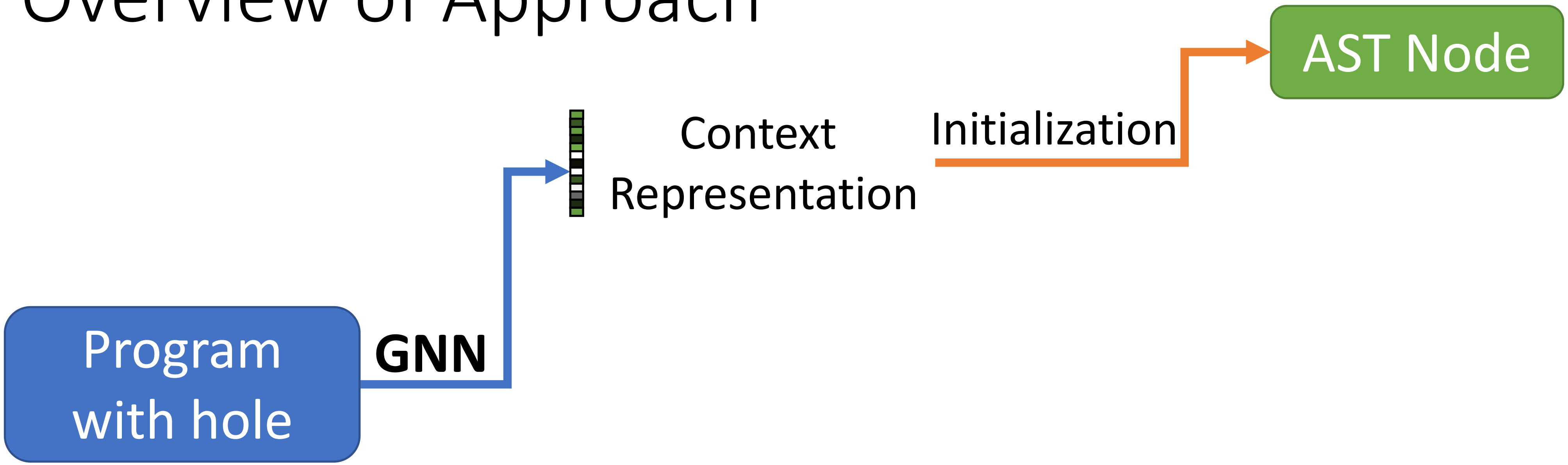


```
int methParamCount = 0;
if (paramCount > 0) {
  IParameterTypeInformation[] moduleParamArr =
    GetParamTypeInformations(Dummy.Signature, paramCount);
  methParamCount = moduleParamArr.Length;
}
if ( paramCount > methParamCount ) {
  IParameterTypeInformation[] moduleParamArr =
    GetParamTypeInformations(Dummy.Signature,
                             paramCount - methParamCount);
}
```

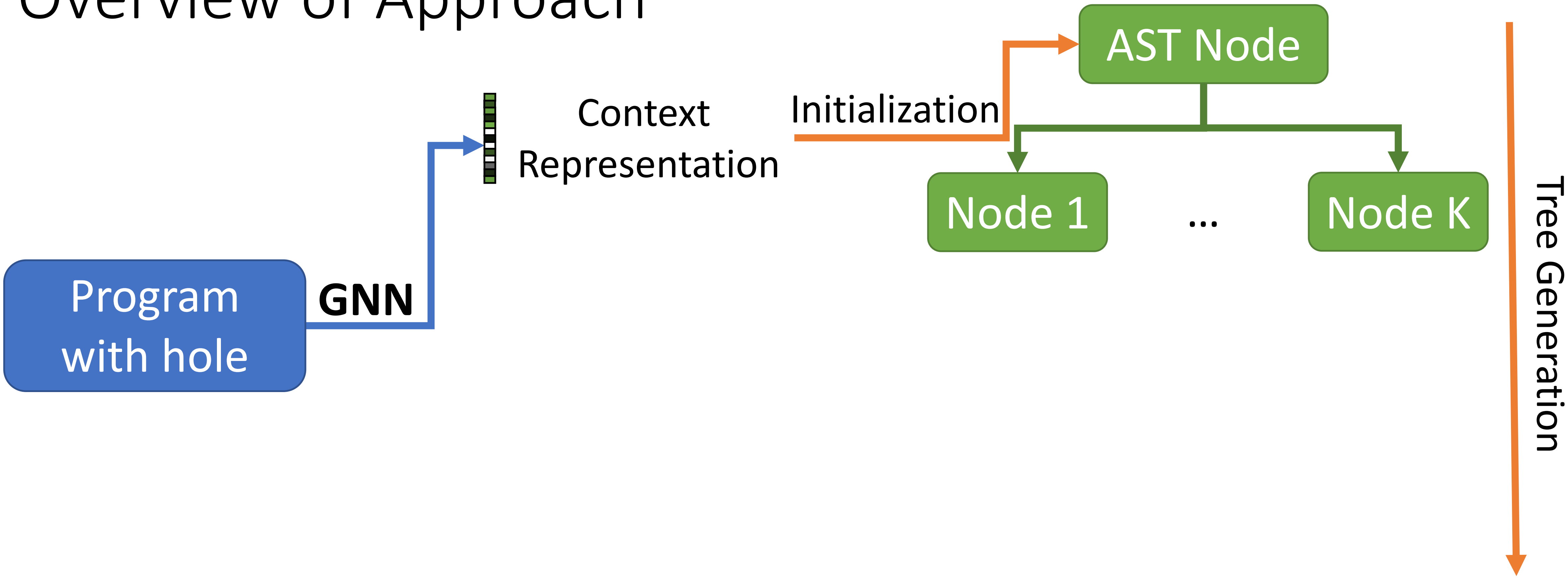
Overview of Approach



Overview of Approach



Overview of Approach



Program with hole

GNN

Context Representation

Initialization

AST Node

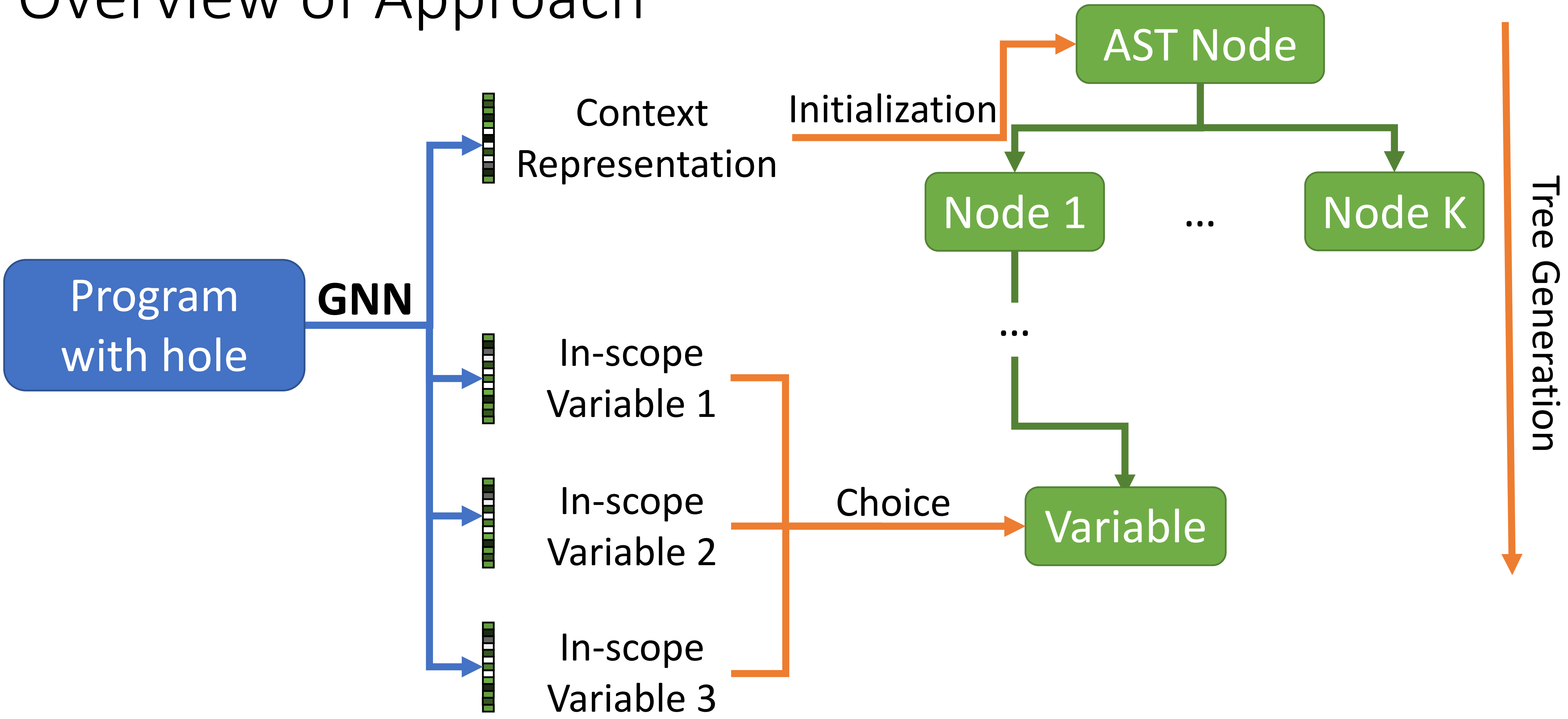
Node 1

…

Node K

Tree Generation

Overview of Approach



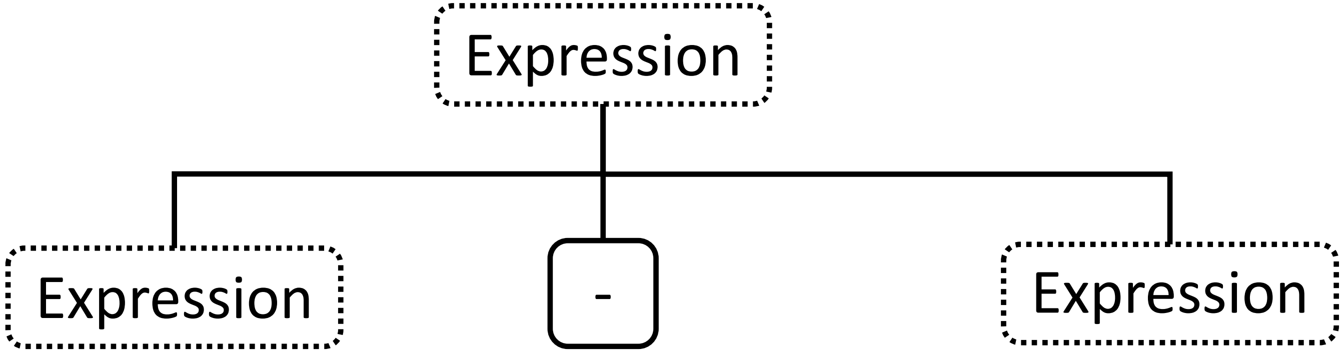
Generating Trees

Variables in scope

Expression

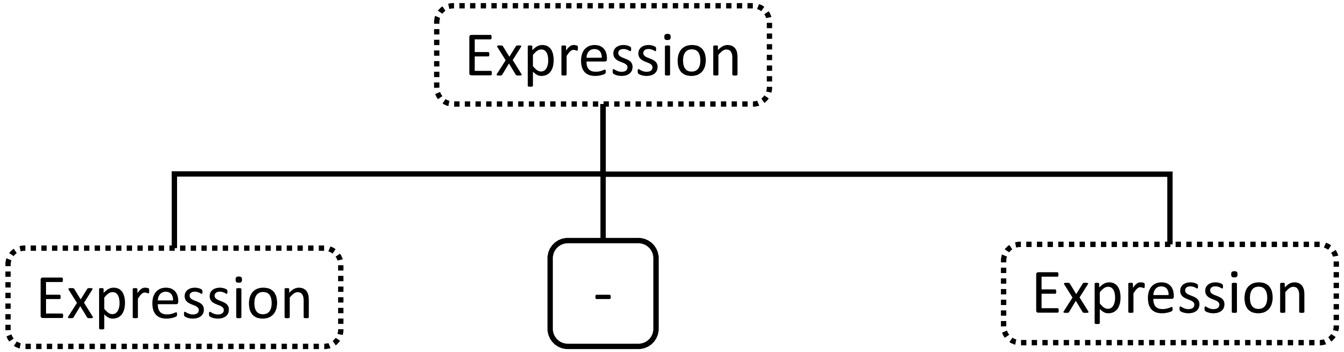
Generating Trees

Variables in scope

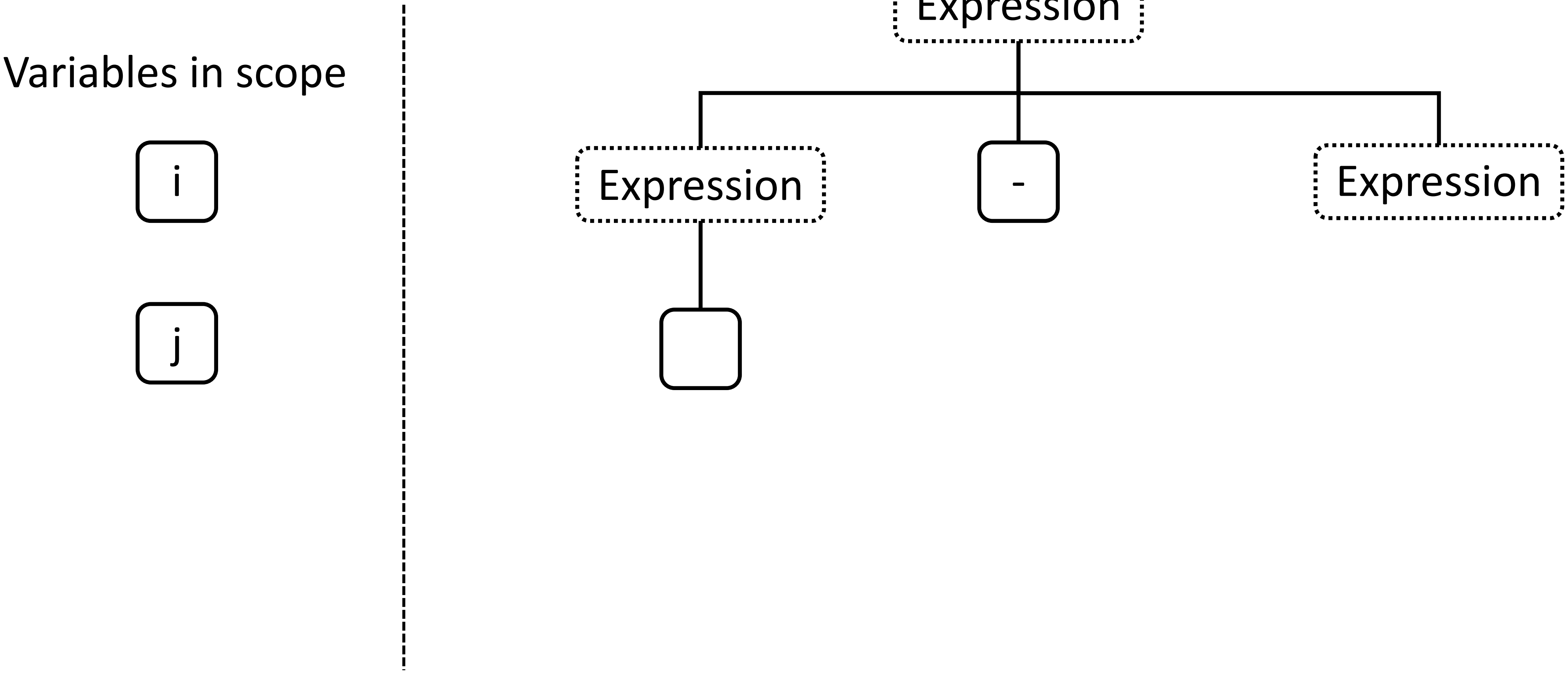


Generating Trees

Variables in scope

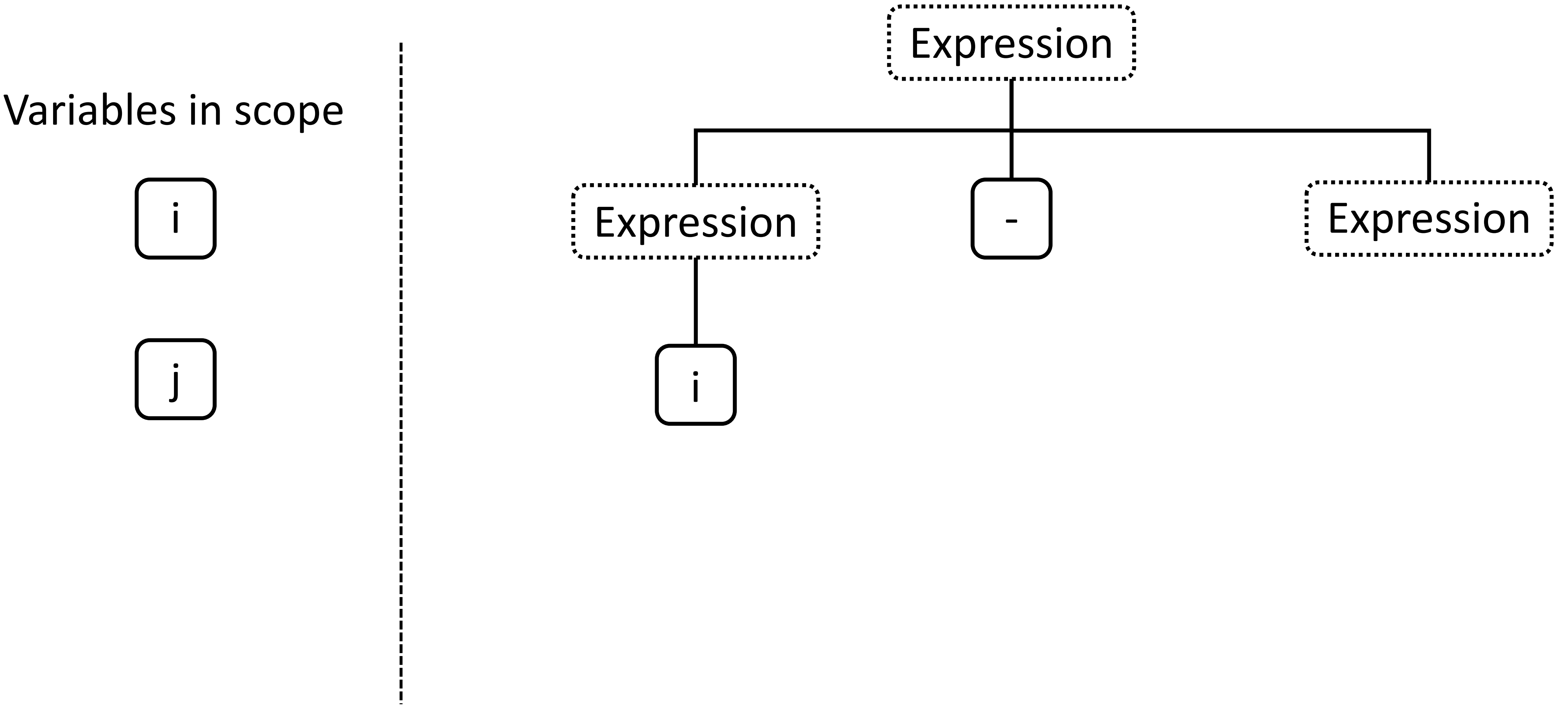


Generating Trees

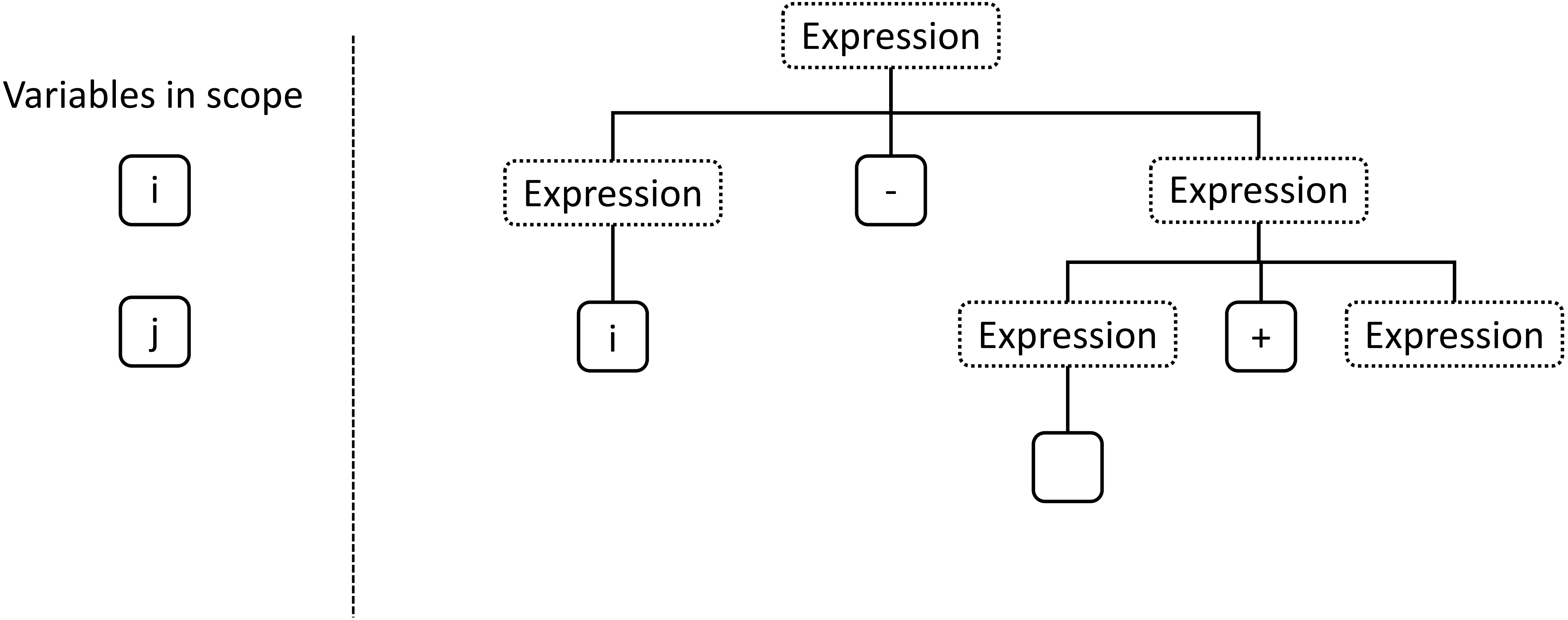


Generating Trees

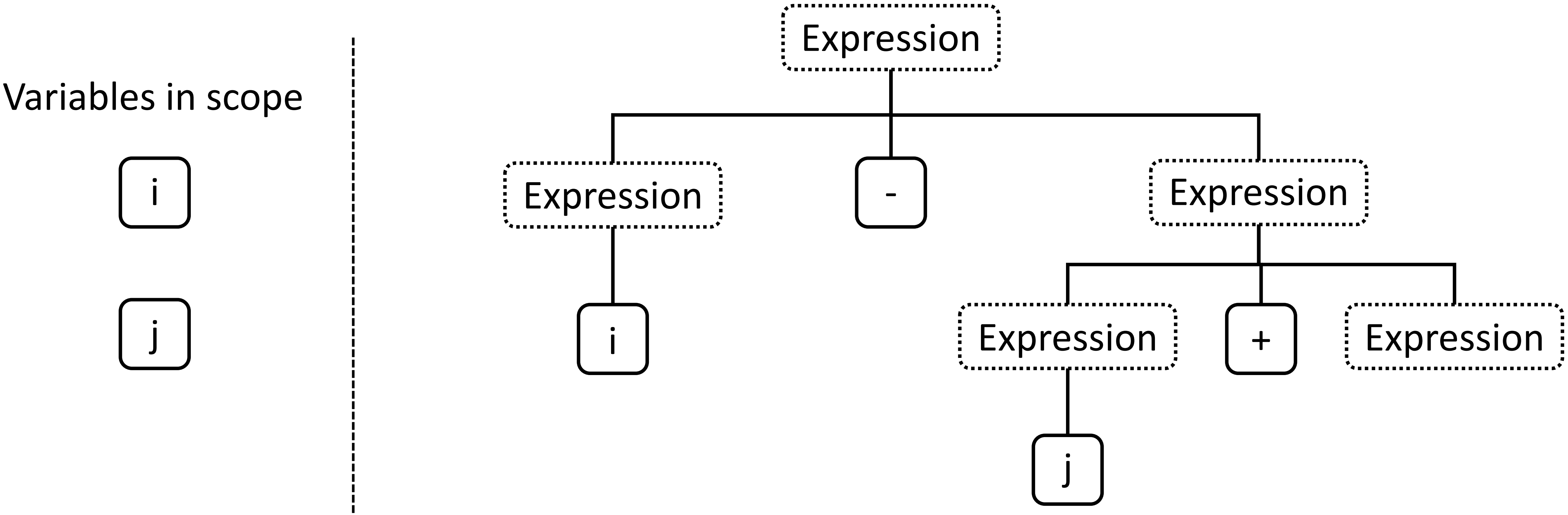
Variables in scope



Generating Trees



Generating Trees

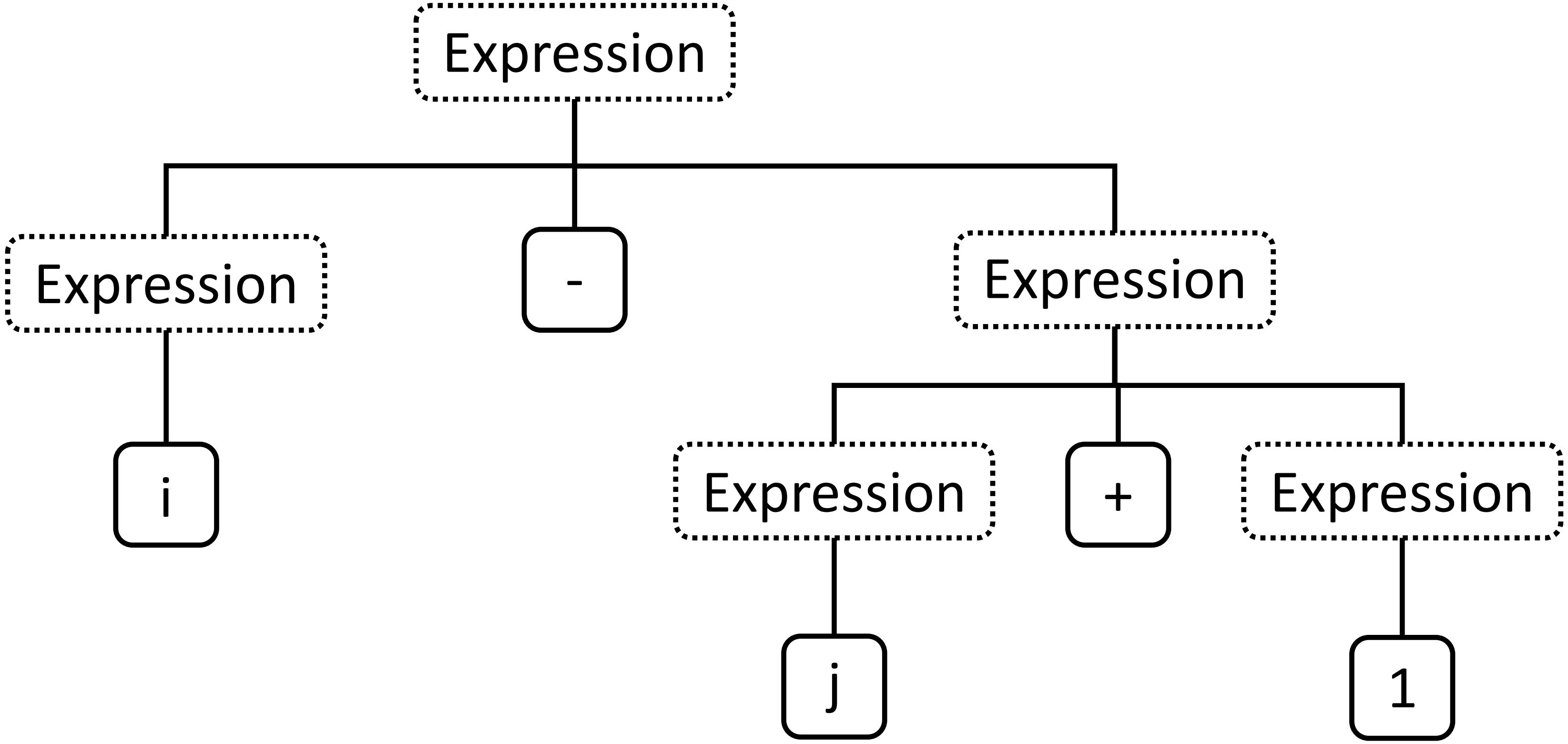


Generating Trees

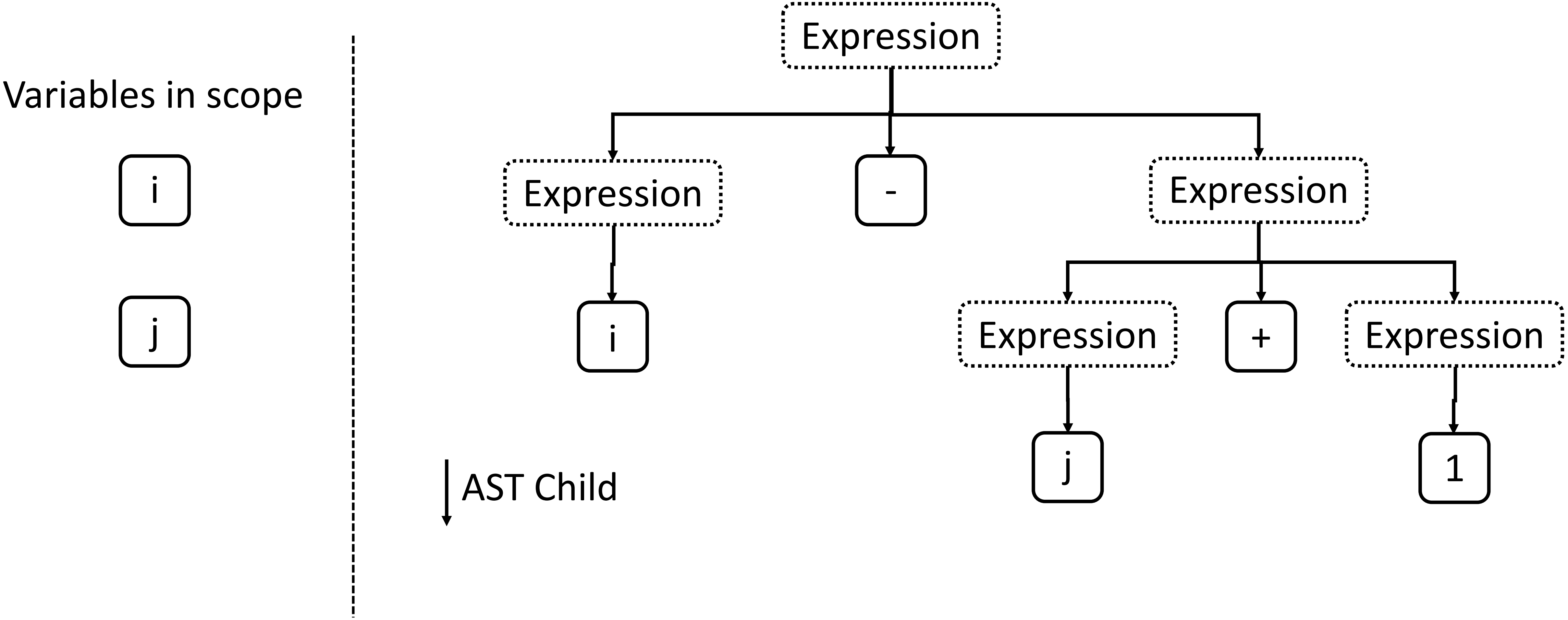
Variables in scope

i

j

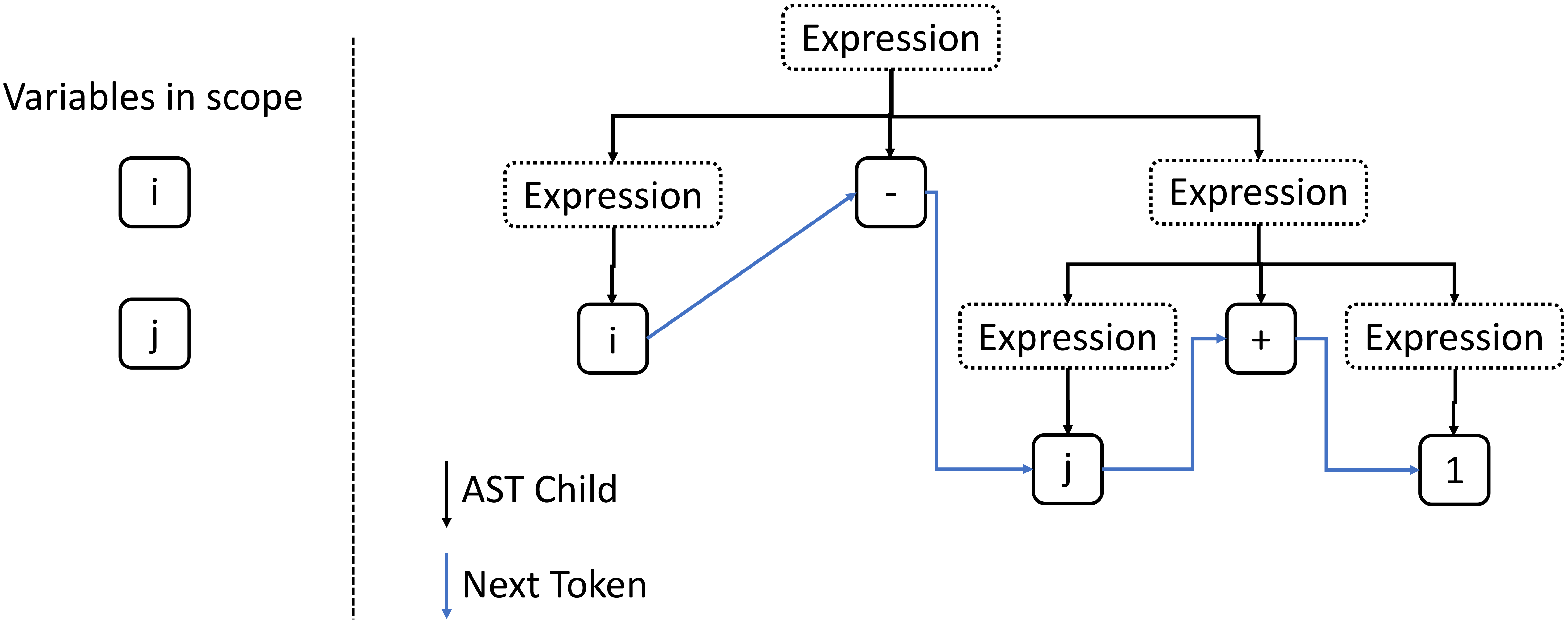


Generating Graphs

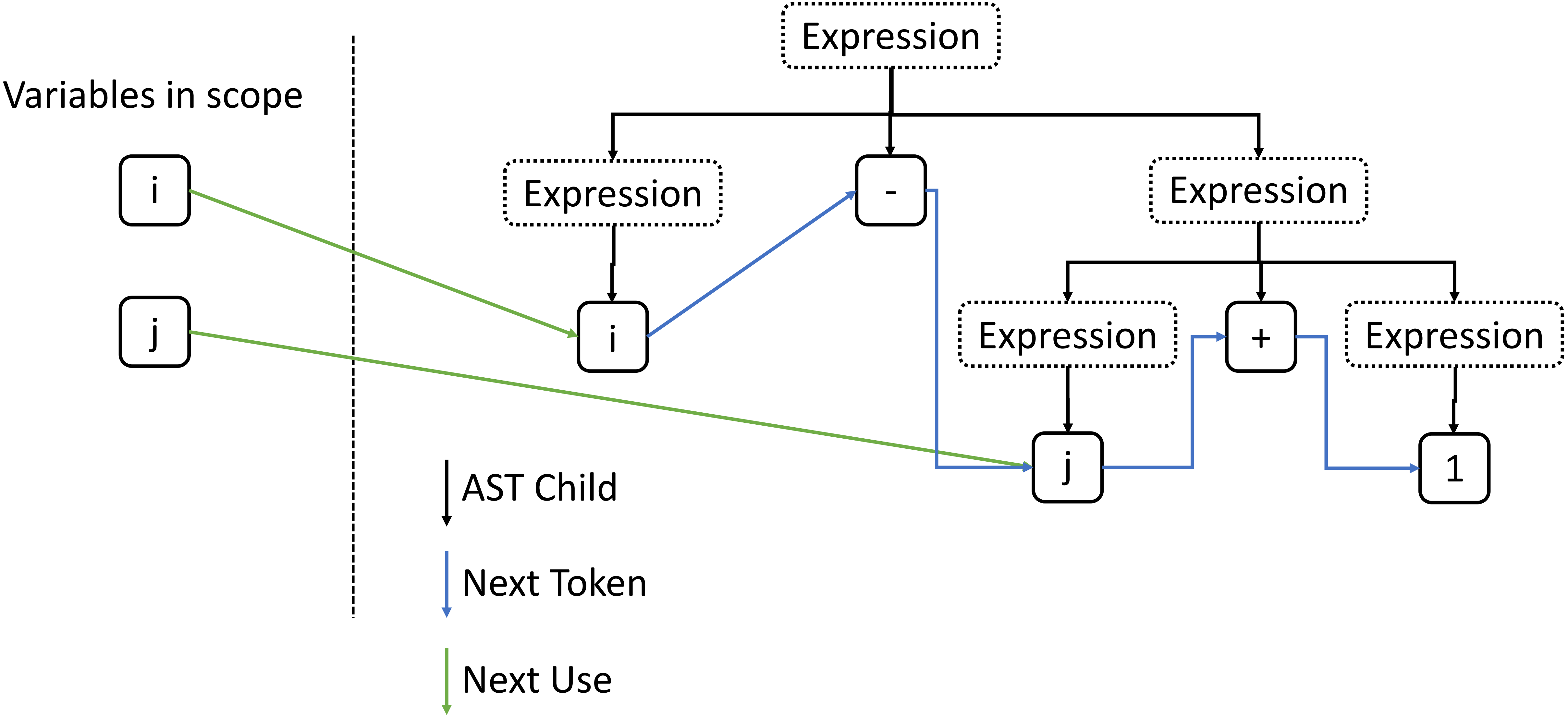


Generating Graphs

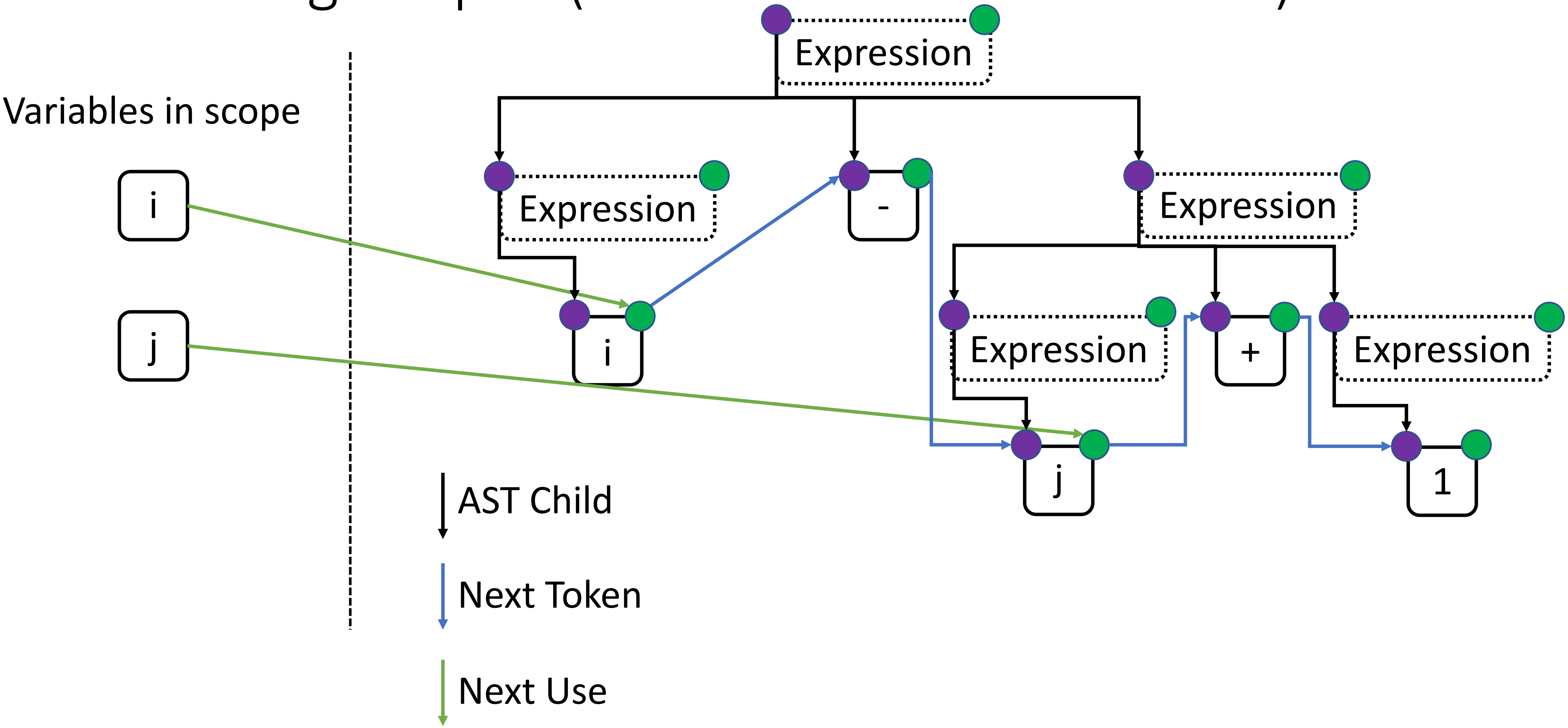
Variables in scope



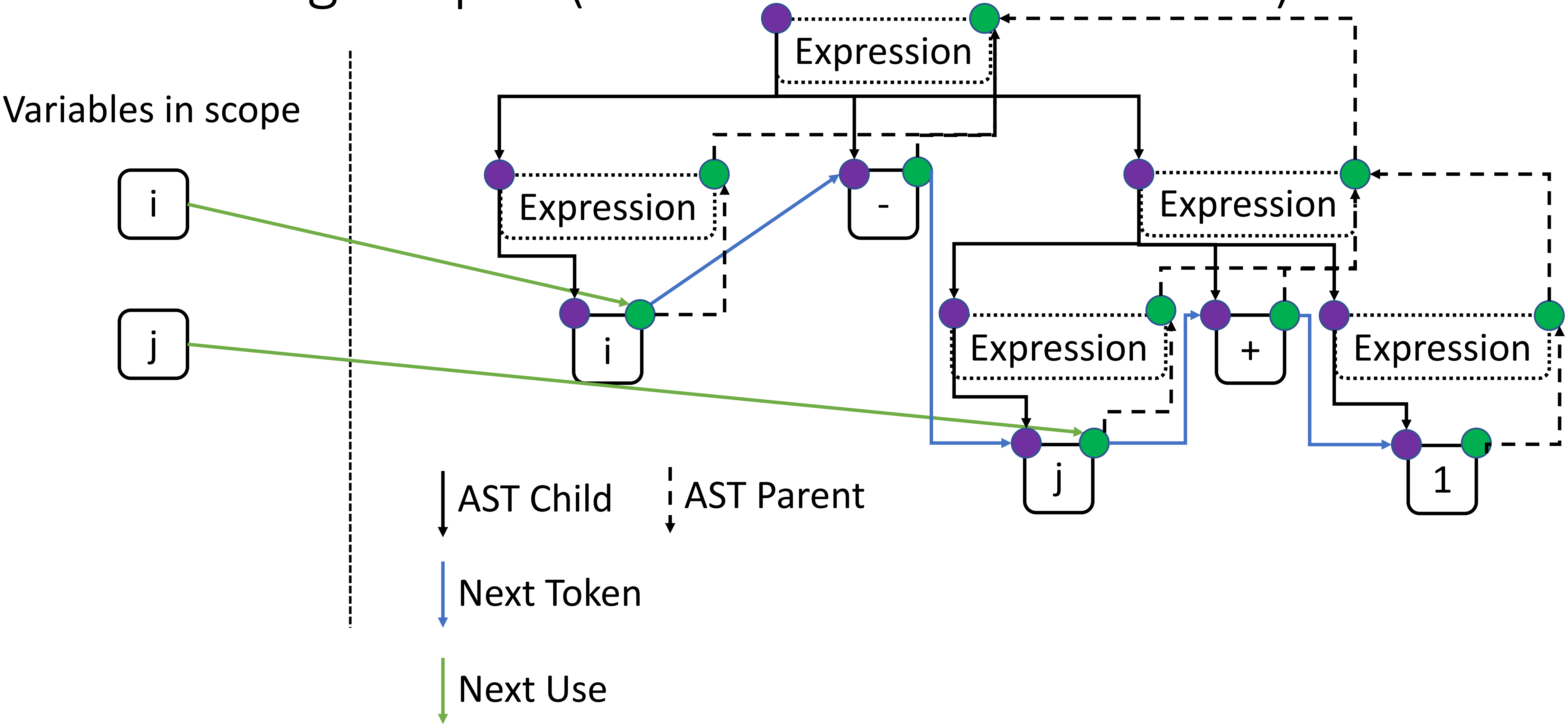
Generating Graphs



Generating Graphs (with Attribute Grammars)

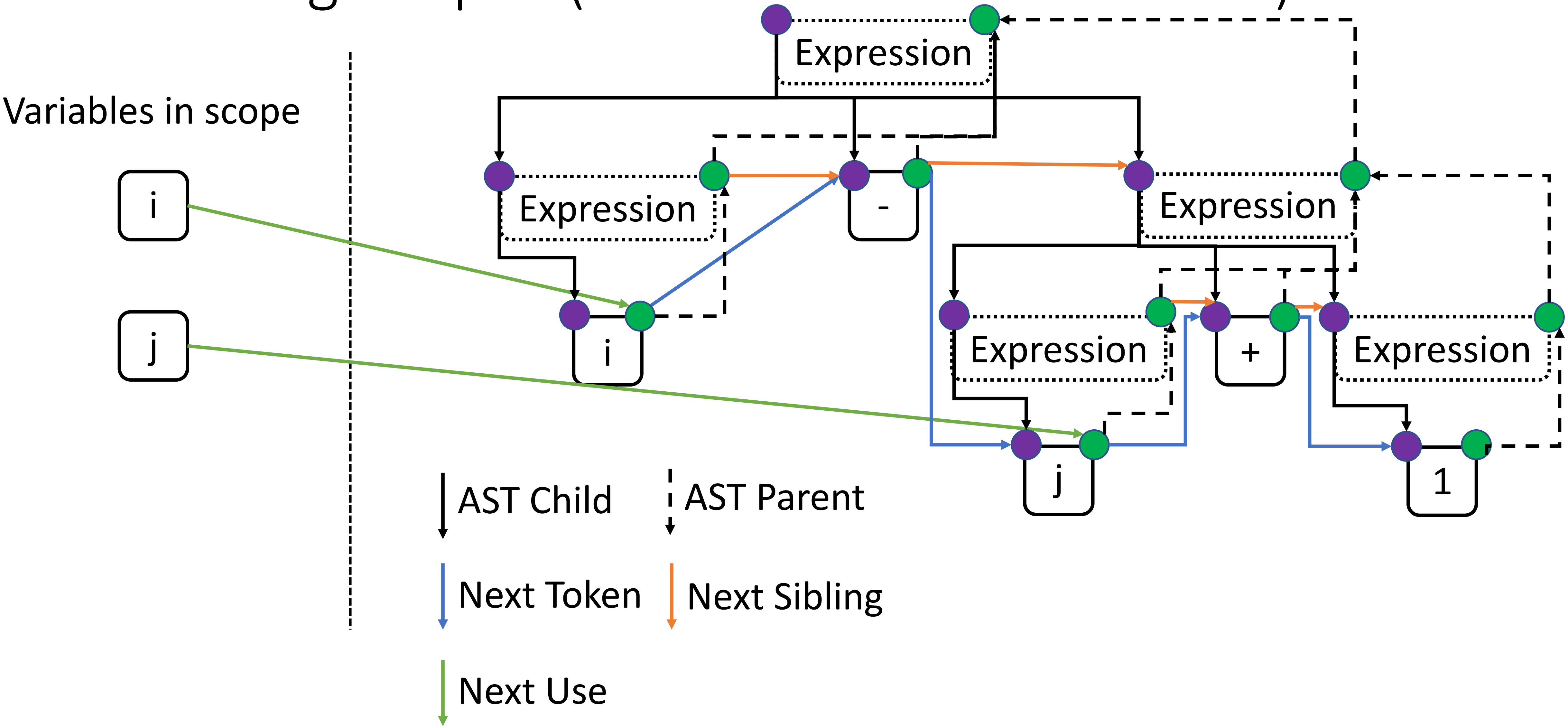


Generating Graphs (with Attribute Grammars)



Variables in scope

Generating Graphs (with Attribute Grammars)



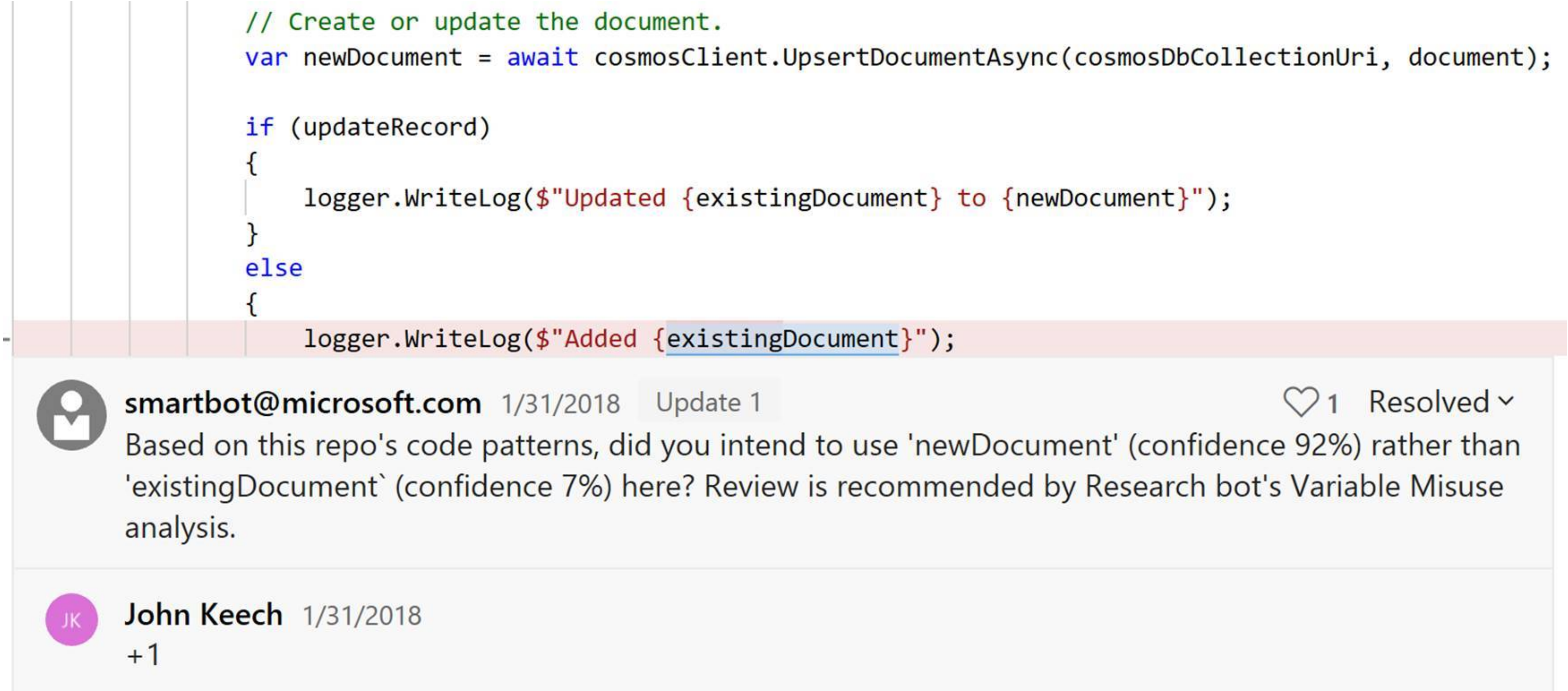
Filling in Blanks: Quantitative Results

Model	Perplexity	Type-Correct	Match@1	Match @5
Seq → NAG	8.38	40.4	8.4	15.8
Graph → Tree	5.37	41.2	19.9	36.8
Graph → Syntax Networks	3.03	74.7	32.4	48.1
Graph → Sequentalised Tree	3.48	84.5	36.0	52.7
Graph → Neural Attr. Gram.	3.07	84.5	38.8	57.0

Training data: 479 C# projects from GitHub
Test data: 114 C# projects from GitHub (~100 000 samples)

UX Lessons Learned

Dogfooding Tales: The Good



```
// Create or update the document.
var newDocument = await cosmosClient.UpsertDocumentAsync(cosmosDbCollectionUri, document);

if (updateRecord)
{
    logger.WriteLog($"Updated {existingDocument} to {newDocument}");
}
else
{
    logger.WriteLog($"Added {existingDocument}");
```

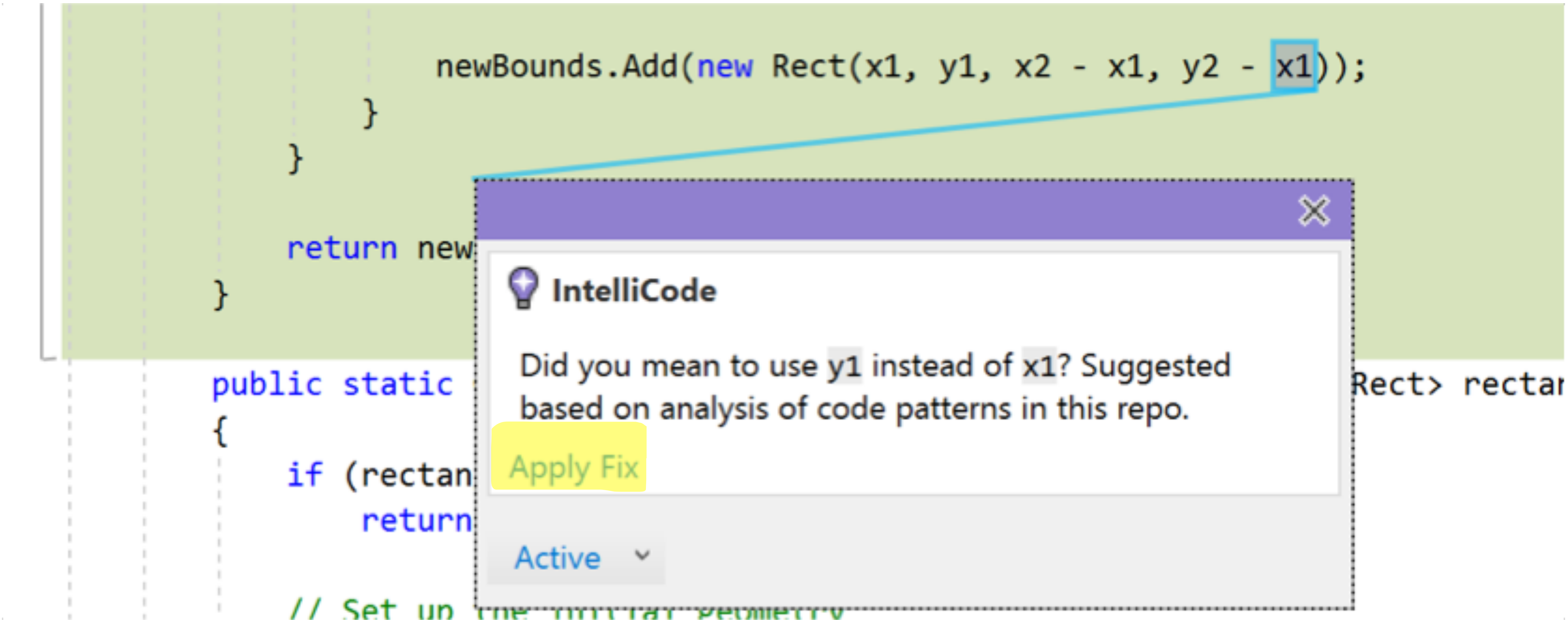
smartbot@microsoft.com 1/31/2018 Update 1 ♡ 1 Resolved

Based on this repo's code patterns, did you intend to use 'newDocument' (confidence 92%) rather than 'existingDocument` (confidence 7%) here? Review is recommended by Research bot's Variable Misuse analysis.

JK **John Keech** 1/31/2018

+1

Dogfooding Tales: The Good

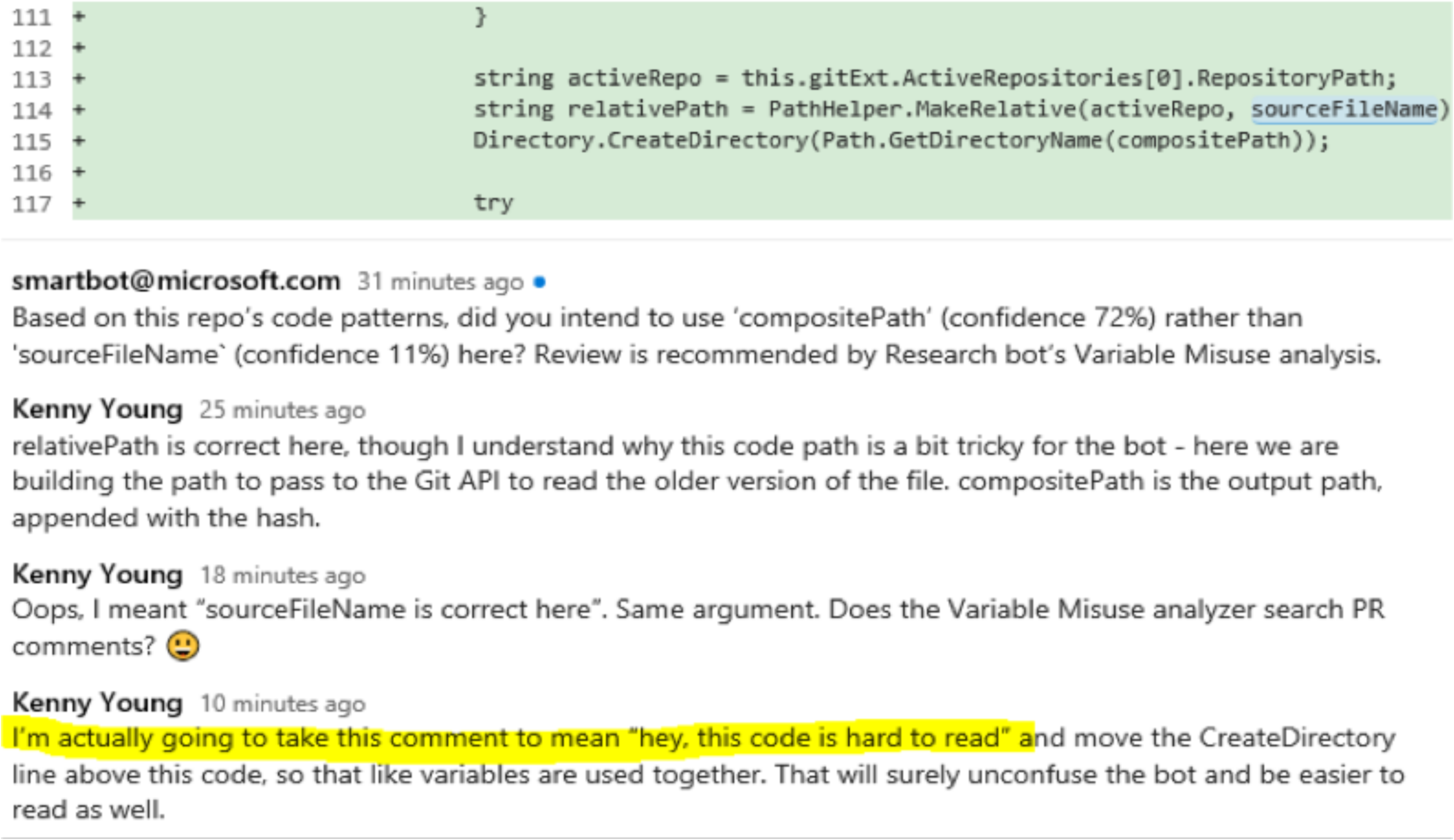


```
                newBounds.Add(new Rect(x1, y1, x2 - x1, y2 - x1));
            }
        }

        return new
    }
    public static                                    Rect> rectar
    {
        if (rectan
            return

        // Set up the initial geometry
```

IntelliCode

Did you mean to use y1 instead of x1? Suggested based on analysis of code patterns in this repo.

Dogfooding Tales: The Strange



```
+                          }
+
+                          string activeRepo = this.gitExt.ActiveRepositories[0].RepositoryPath;
+                          string relativePath = PathHelper.MakeRelative(activeRepo, sourceFileName)
+                          Directory.CreateDirectory(Path.GetDirectoryName(compositePath));
+
+                          try
```

smartbot@microsoft.com 31 minutes ago •

Based on this repo's code patterns, did you intend to use 'compositePath' (confidence 72%) rather than 'sourceFileName` (confidence 11%) here? Review is recommended by Research bot's Variable Misuse analysis.

Kenny Young 25 minutes ago

relativePath is correct here, though I understand why this code path is a bit tricky for the bot - here we are building the path to pass to the Git API to read the older version of the file. compositePath is the output path, appended with the hash.

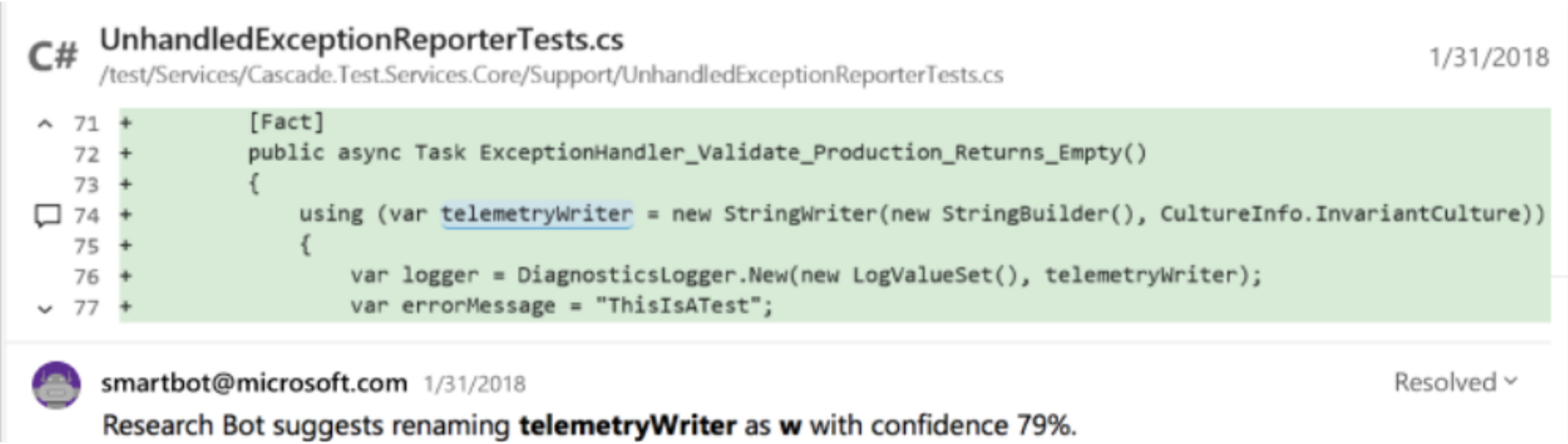
Kenny Young 18 minutes ago

Oops, I meant "sourceFileName is correct here". Same argument. Does the Variable Misuse analyzer search PR comments? 😃

Kenny Young 10 minutes ago

I'm actually going to take this comment to mean "hey, this code is hard to read" and move the CreateDirectory line above this code, so that like variables are used together. That will surely unconfuse the bot and be easier to read as well.

Dogfooding Tales: The Bad



C# **UnhandledExceptionReporterTests.cs**

/test/Services/Cascade.Test.Services.Core/Support/UnhandledExceptionReporterTests.cs

1/31/2018

```
+         [Fact]
+         public async Task ExceptionHandler_Validate_Production_Returns_Empty()
+         {
+             using (var telemetryWriter = new StringWriter(new StringBuilder(), CultureInfo.InvariantCulture))
+             {
+                 var logger = DiagnosticsLogger.New(new LogValueSet(), telemetryWriter);
+                 var errorMessage = "ThisIsATest";
```

smartbot@microsoft.com 1/31/2018 Resolved

Research Bot suggests renaming **telemetryWriter** as **w** with confidence 79%.

Understanding and Generating Source Code

Question: How to learn from code with semantics?

Hypothesis: Code is natural, targets people **and** machines

Our Solution: Graphs representing all modalities

Understanding and Generating Source Code

Question: How to learn from code with semantics?

Hypothesis: Code is natural, targets people **and** machines

Our Solution: Graphs representing all modalities

Marc Brockschmidt

@mmjb86

Microsoft