

# Program Synthesis

Alex Polozov  
Microsoft Research AI

[polozov@microsoft.com](mailto:polozov@microsoft.com)

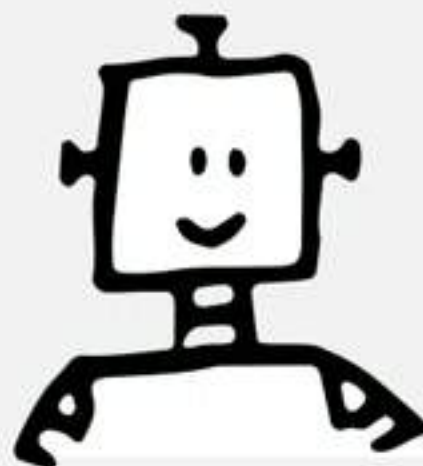
# Program Synthesis: “The Golden Dream of CS”

(circa 1950-1960)

## Sort

**Given:**  $A[1..n]$

**Produce:**  $B[1..n]$  ensuring  $B_1 \leq \dots \leq B_n$  and  $B$  is a permutation of  $A$



```
void bubblesort(int *array, int length)
{
    int i, j;
    for (i = 0; i < length - 1; ++i)
    {
        for (j = 0; j < length - i - 1; ++j)
        {
            if (array[j] > array[j + 1])
            {
                int tmp = array[j];
                array[j] = array[j + 1];
                array[j + 1] = tmp;
            }
        }
    }
}
```

# Modern Program Synthesis

- Input-output examples
- Natural language

- Programming language
- Domain-specific language

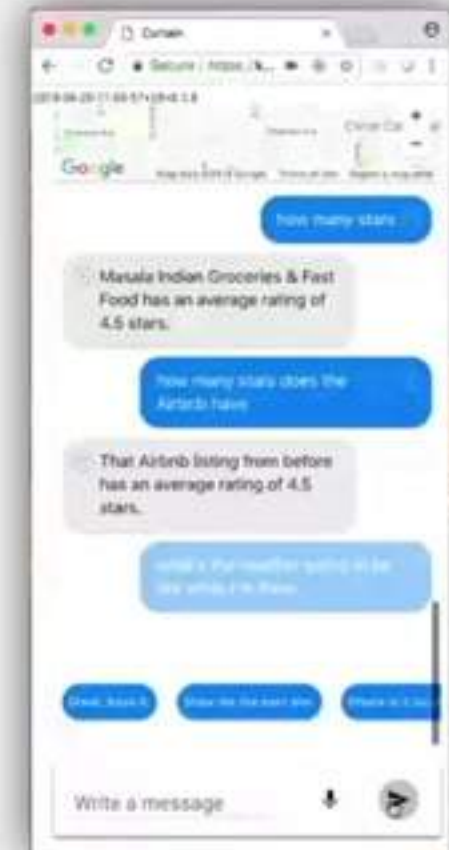


- Enumerative/deductive search
- Neural networks

# Application: Question answering

Natural language question

Command interpretations  
+ query plans



# Application: Data wrangling

Input-output examples



String → String data transforms



	A	B	C	D	E
1	<b>First Name</b>	<b>Middle Name</b>	<b>Last Name</b>	<b>Combined Name</b>	
2	Mary	Kristy	Mills	Mills, Mary K.	
3	Craig	John	Garland	Garland, Craig J.	
4	Andrew		Simpson		
5	Rupert	L	Richards	Richards, Rupert L.	
6	Joe		Bryant		
7	Jane		Hart		
8	Bob	JD	Bond	Bond, Bob JD.	
9					
10					

 Flash Fill

# Application: Code generation

Program context/template



Programming language



```
import java.io.*;

object Main {
  def main(args:Array[String]) = {
    var body = "small.txt"
    var sig = "signature.txt"

    var inputStreamSequenceInputStream =
      new SequenceInputStream(new FileInputStream(body), new FileInputStream(sig))

    var out:Boolean = false;
    var bytesCount:Int = 0;
    while (!eof) {
      var c:Char = inputStream.read()
      out = true;
      else {
        System.out.println(c.toChar);
        bytesCount+=1;
      }
    }
    System.out.println(bytesCount + " bytes were read");
    inputStream.close();
  }
}
```

# Outline

- Introduction
- Programming by Examples
- Neural-Guided Program Search
- Neural Program Synthesis

# Outline

- Introduction
- **Programming by Examples**
- Neural-Guided Program Search
- Neural Program Synthesis





# Flash Fill

[Gulwani, 2011]

	A	B	C	D	E
1	<b>First Name</b>	<b>Middle Name</b>	<b>Last Name</b>	<b>Combined Name</b>	
2	Mary	Kristy	Mills	Mills, Mary K.	
3	Craig	John	Garland	Garland, Craig J.	
4	Andrew		Simpson		
5	Rupert	L	Richards	Richards, Rupert L.	
6	Joe		Bryant		
7	Jane		Hart		
8	Bob	JD	Bond	Bond, Bob JD.	
9					
10					

```
Concat(input[2], Const(",", ""), input[0], Const(" ")),  
  let v = input[1] in Substring(v, AbsPos(v, 0), RegexPos(v, UpperCase, "'", -1)),  
  Const("."))
```



# Flash Fill

[Gulwani, 2011]

	A	B	C	D	E
1	<b>First Name</b>	<b>Middle Name</b>	<b>Last Name</b>	<b>Combined Name</b>	
2	Mary	Kristy	Mills	Mills, Mary K.	
3	Craig	John	Garland	Garland, Craig J.	
4	Andrew		Simpson	Simpson, A.	
5	Rupert	L	Richards	Richards, Rupert L.	
6	Joe		Bryant	Bryant, J.	
7	Jane		Hart	Hart, J.	
8	Bob	JD	Bond	Bond, Bob JD.	
9					
10					

```
if (input[1] == '') then Concat(input[2], Const(", "), input[0], Const(". ")) else
Concat(input[2], Const(", "), input[0], Const(" "),
  let v = input[1] in Substring(v, AbsPos(v, 0), RegexPos(v, UpperCase, '', -1)),
  Const(". "))
```

from the beginning till the last uppercase character



# Flash Fill

[Gulwani, 2011]

	A	B	C	D	E
1	<b>First Name</b>	<b>Middle Name</b>	<b>Last Name</b>	<b>Combined Name</b>	
2	Mary	Kristy	Mills	Mills, Mary K.	
3	Craig	John	Garland	Garland, Craig J.	
			Simpson	Simpson, A.	
		L	Richards	Richards, L.	
			Bryant	Bryant, J.	
			Hart	Hart, J.	
		JD	Bond	Bond, Bob.	

Domain-Specific Language (DSL)

Programming by Examples (PBE)

```
if (input[1] == '') then Concat(input[2], Const(", "), input[0], Const(". ")) else
Concat(input[2], Const(", "), input[0], Const(" "),
  let v = input[1] in Substring(v, AbsPos(v, 0), RegexPos(v, UpperCase, "", -1)),
  Const(". "))
```

from the beginning till the last uppercase character

# PBE Challenges

1. The synthesized program *must* satisfy the examples.
2. Examples are severely ambiguous.
  - Should learn the *intended* program, not just one that satisfies the examples
  - How do we know when to stop?
3. Interactive UX requires quick turnaround.
4. The DSL must be expressive but concise.

# Flash Fill DSL (excerpt)

```
@input string[] inputs;
```

```
@start string transform := atom | Concat(atom, transform);
```

```
string atom := Const(s)
```

```
    | let v = std.Kth(inputs, k) in  
      Substring(v, std.Pair(pos, pos));
```

```
uint? pos := AbsPos(v, k)
```

```
    | RegexPos(v, std.Pair(r, r), k);
```

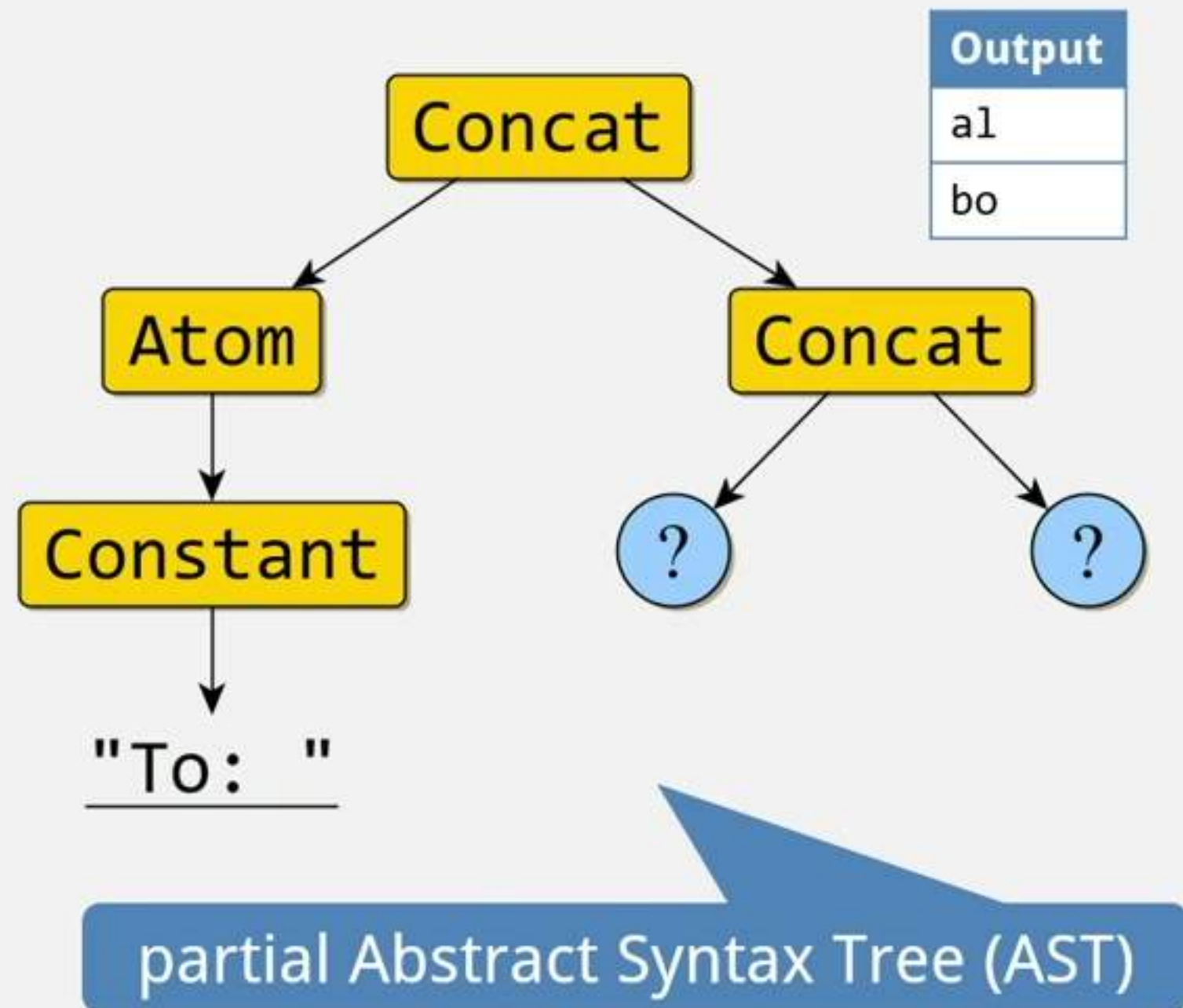
```
string s;
```

```
Regex r;
```

```
int k;
```

# Deductive Search in PBE

[Polozov & Gulwani, 2015]



Input $x$	Output $y$
-----------	------------

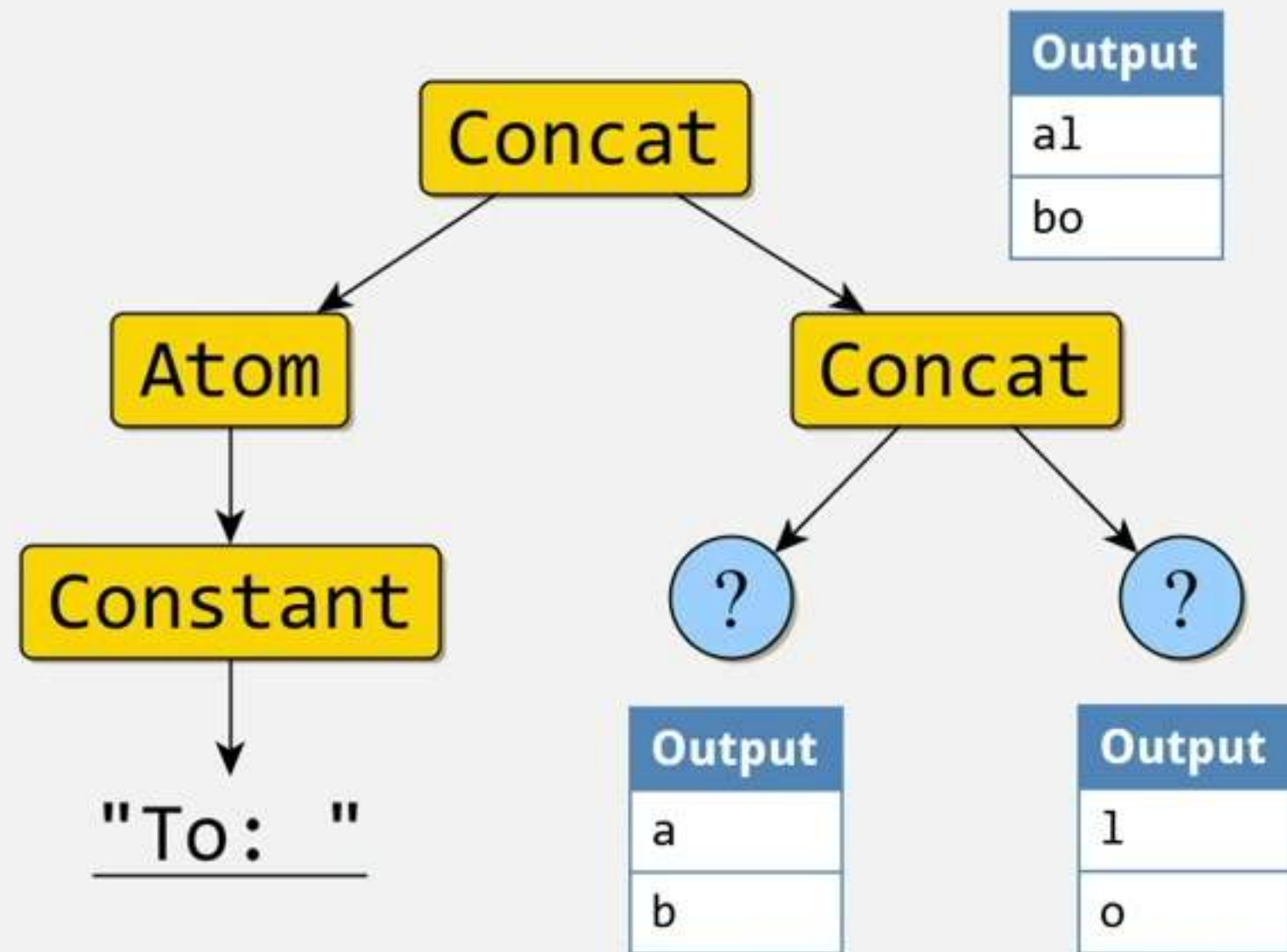
alice liddell	To: al
---------------	--------

bob o'reilly	To: bo
--------------	--------

1. Select a hole (initially root).
2. Select a DSL operator to insert in the hole.

# Deductive Search in PBE

[Polozov & Gulwani, 2015]



**Input  $x$**

alice liddell

bob o'reilly

**Output  $y$**

To: a1

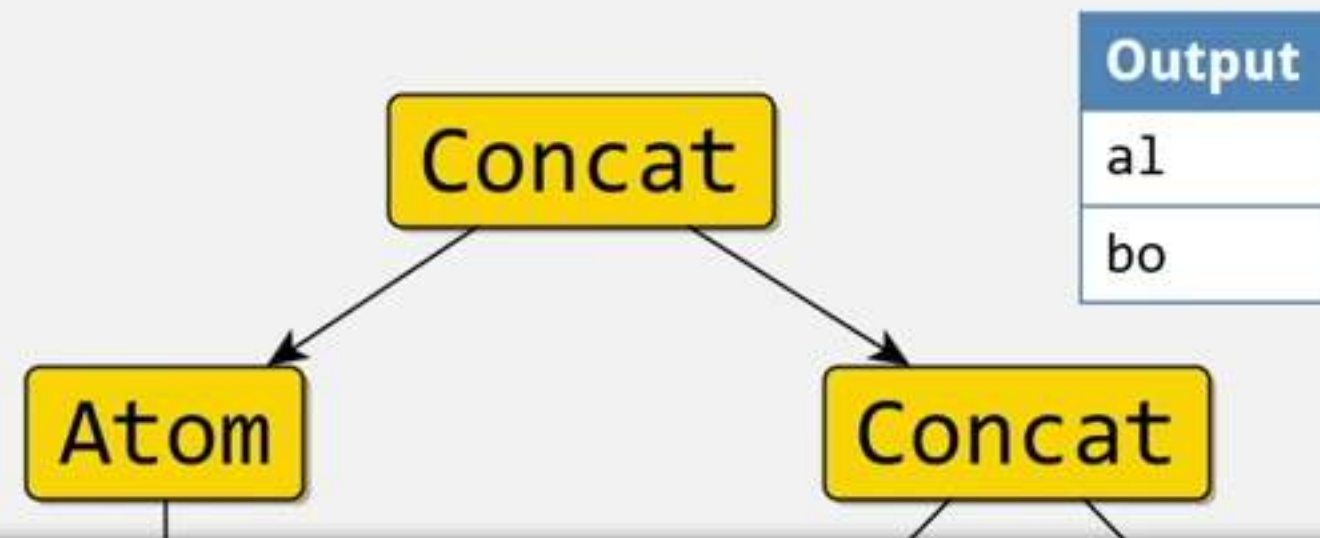
To: bo

1. Select a hole (initially root).
2. Select a DSL operator to insert in the hole.
3. Propagate the examples.

# Deductive Search in PBE

[Polozov & Gulwani, 2015]

Input $x$	Output $y$
alice liddell	To: al
bob o'reilly	To: bo



Output
al
bo

```
def propagate_concat(output spec  $\varphi = \{x \rightsquigarrow y\}$ ):  
    return  $\varphi_{prefix} = \{x \rightsquigarrow y[:1] \vee y[:2] \vee \dots\}$ 
```

"To: "

Output
a
b

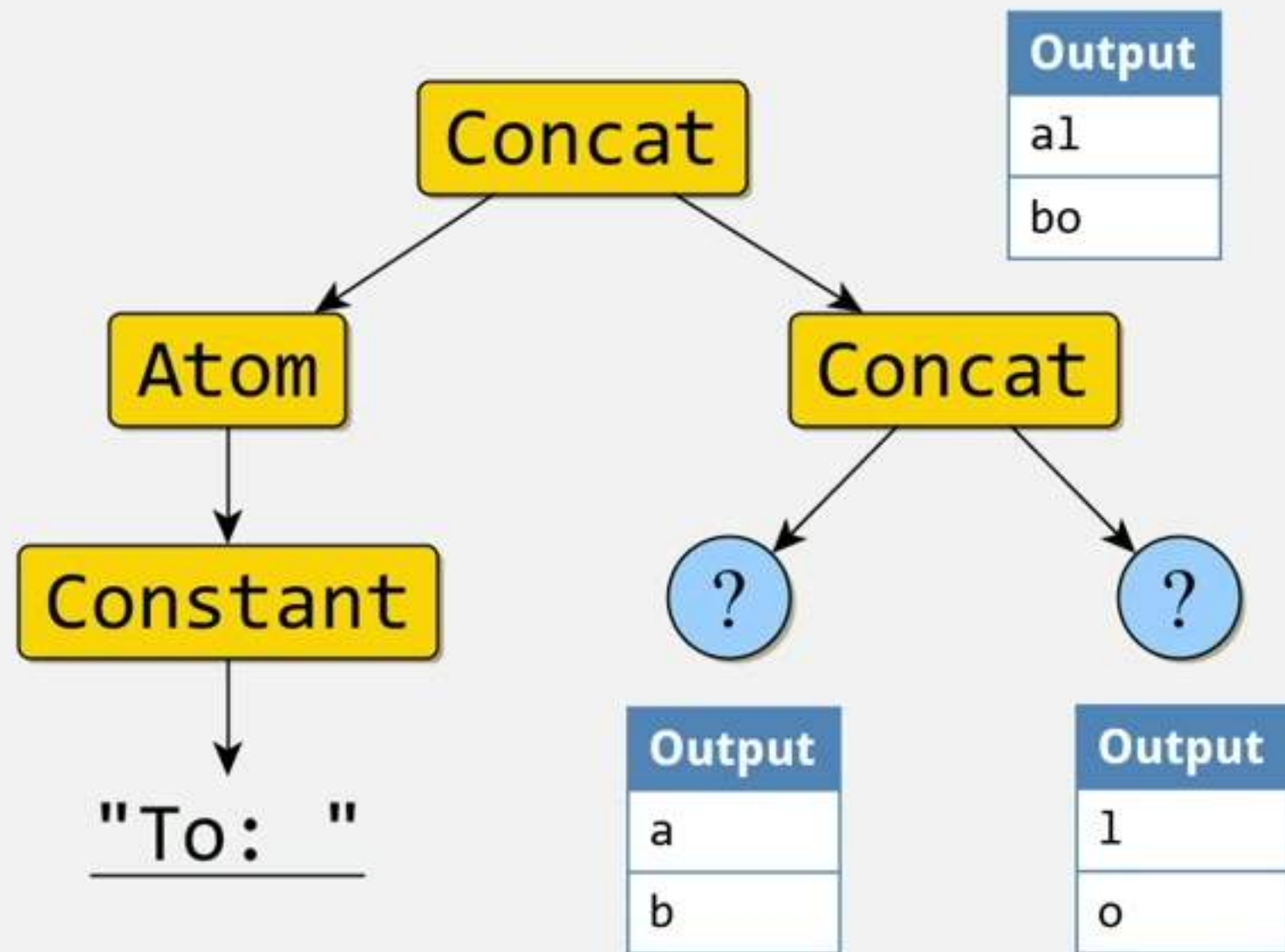
Output
l
o

1. Select a hole (initially root).
2. Select a DSL operator to insert in the hole.
3. Propagate the examples.



# Deductive Search in PBE

[Polozov & Gulwani, 2015]



Input $x$	Output $y$
alice liddell	To: a1
bob o'reilly	To: bo

1. Select a hole (initially root).
2. Select a DSL operator to insert in the hole.
3. Propagate the examples.

- ✓ Correct by construction
- ✓ Can incorporate program ranking
- ✓ Example propagation easy to write for many operations & domains

# Microsoft PROSE SDK

<https://microsoft.github.io/prose/>

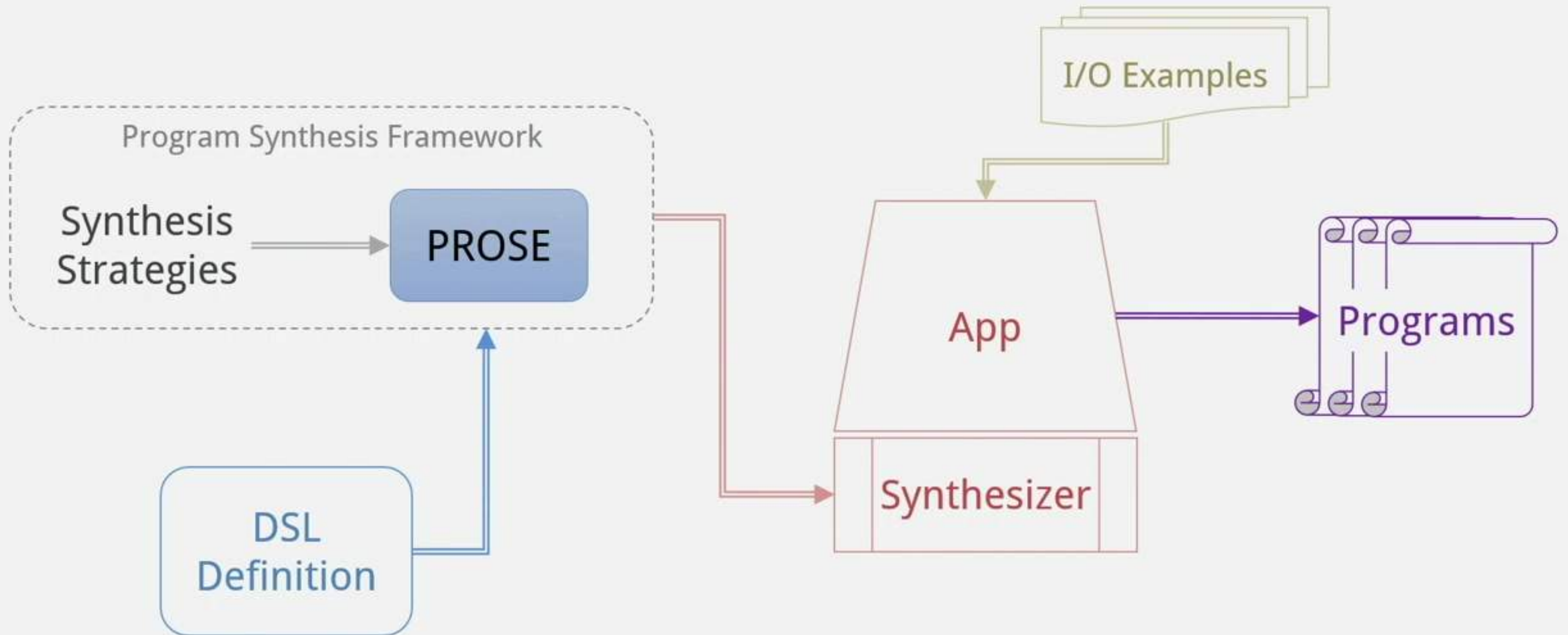
The screenshot shows the Microsoft PROSE SDK website. On the left is a navigation menu with sections: PROSE SDK, Tutorial, FAQ, Demos (Text Splitting), Videos (Overview, PROSE Showcase, Flash Fill/Extract, Data Wrangling), and Documentation (PROSE Framework, Text Extraction, Json Extraction, Text Splitting). The main content area features the title "Microsoft Program Synthesis using Examples SDK" and a subtitle "A framework for automatic programming or data wrangling from input-output examples." Below this are links for "Latest version: 6.7.0", "Release notes", "NuGet package", "Yeoman generator", and "Samples".

The central diagram illustrates the workflow:

- Program Synthesis Framework:** Takes "Intent spec  $\varphi$ : input-output examples" as input. It uses a "Program ranking function  $h(P)$ " and a "Domain-specific language  $\mathcal{L}$ ". It outputs a "Ranked set of valid programs  $\mathcal{S}$ ".
- Data Wrangling DSLs:** Involves a "User (Debugging)" who provides "Interactive questions" and "Test inputs". The user selects the "Best program" from the ranked set.
- Refinement:** The user's input leads to a "Refined intent", which is fed back into the Program Synthesis Framework.
- Output:** The "Best program" is passed to a "Translator", which produces an "Intended program  $P \in \mathcal{L}$ " and finally "Deployable code in Python/R/C#/..."

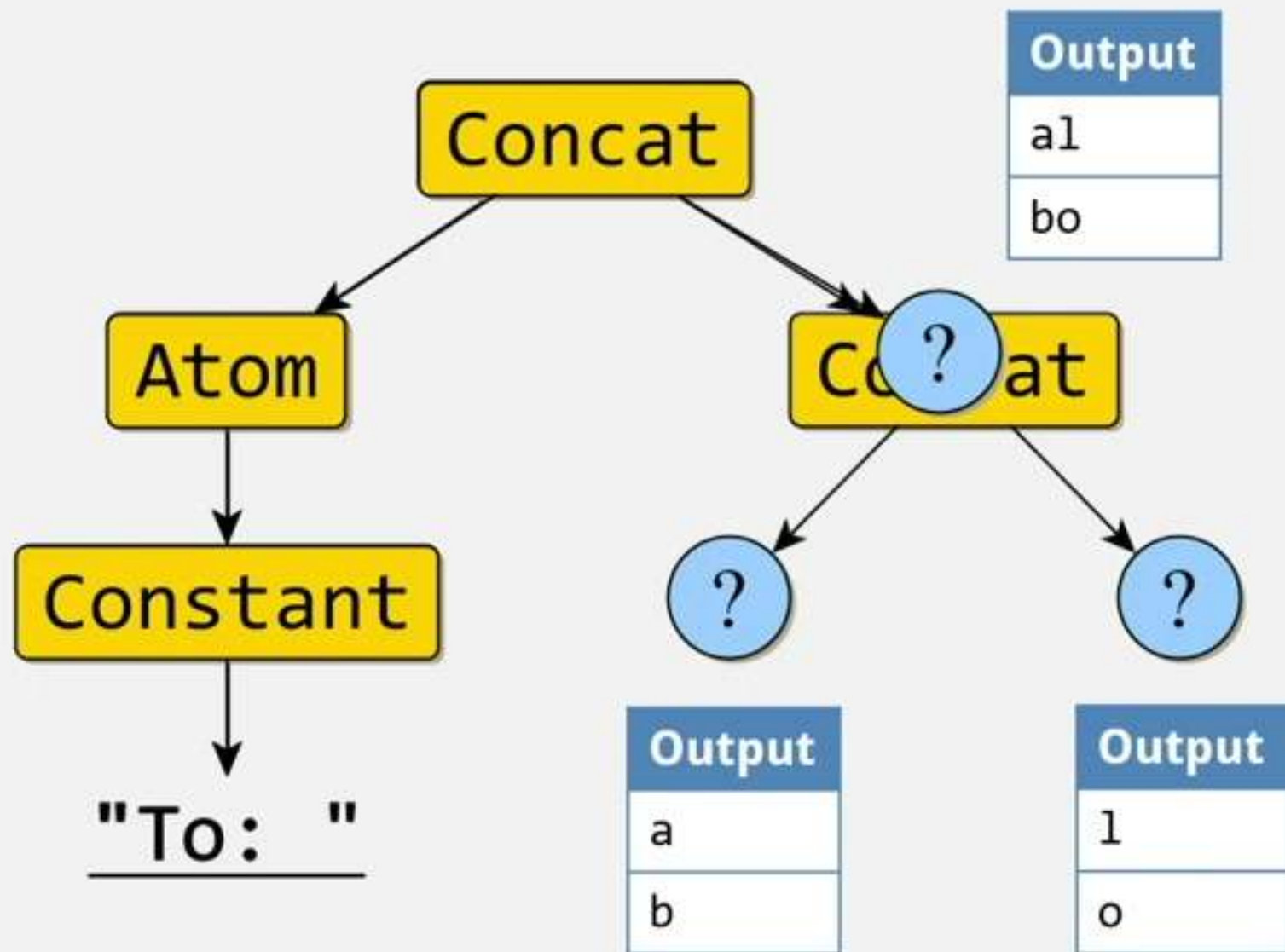
Additional labels in the diagram include "Disambiguation" (represented by a robot icon) and "Intended program  $P \in \mathcal{L}$ ".

# Microsoft PROSE SDK



# Deductive Search in PBE

[Polozov & Gulwani, 2015]



Input $x$	Output $y$
alice liddell	To: a1
bob o'reilly	To: bo

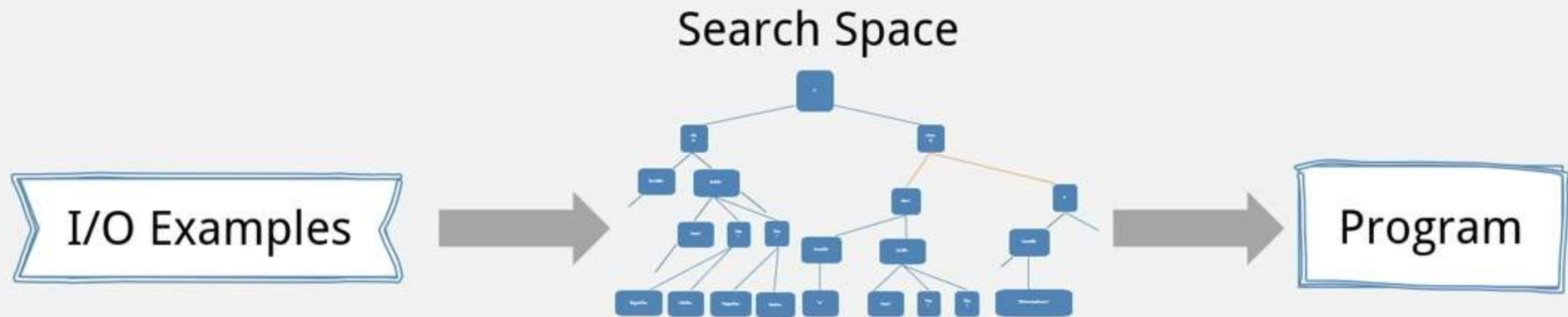
1. Select a hole (initially root).
2. Select a DSL operator to insert in the hole.
3. Propagate the examples.

- ✓ Correct by construction
- ✓ Can incorporate program ranking
- ✓ Example propagation easy to write for many operations & domains

# Outline

- Introduction
- Programming by Examples
- **Neural-Guided Program Search**
- Neural Program Synthesis

# Deductive Search



Why so slow? Explores the entire search space  
(unless deduction prunes some of it)

# Machine-learned insights

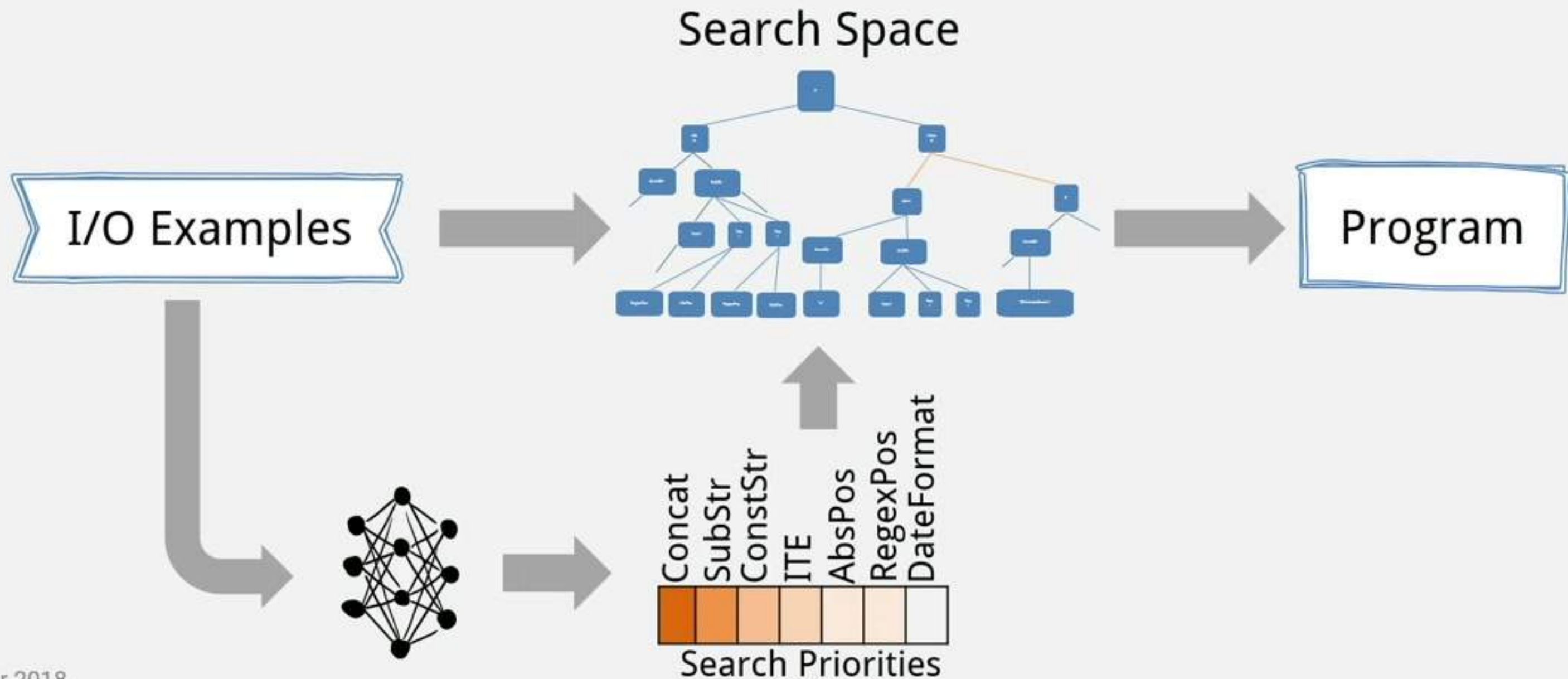
Input	Output
alice liddell	To: al
bob o'reilly	To: bo

Can't be a substring, requires concatenation

# DeepCoder: Learning to Write Programs

[Balog et al., 2017]

**Idea:** Order the search space based on a priority list from DNN *before starting*

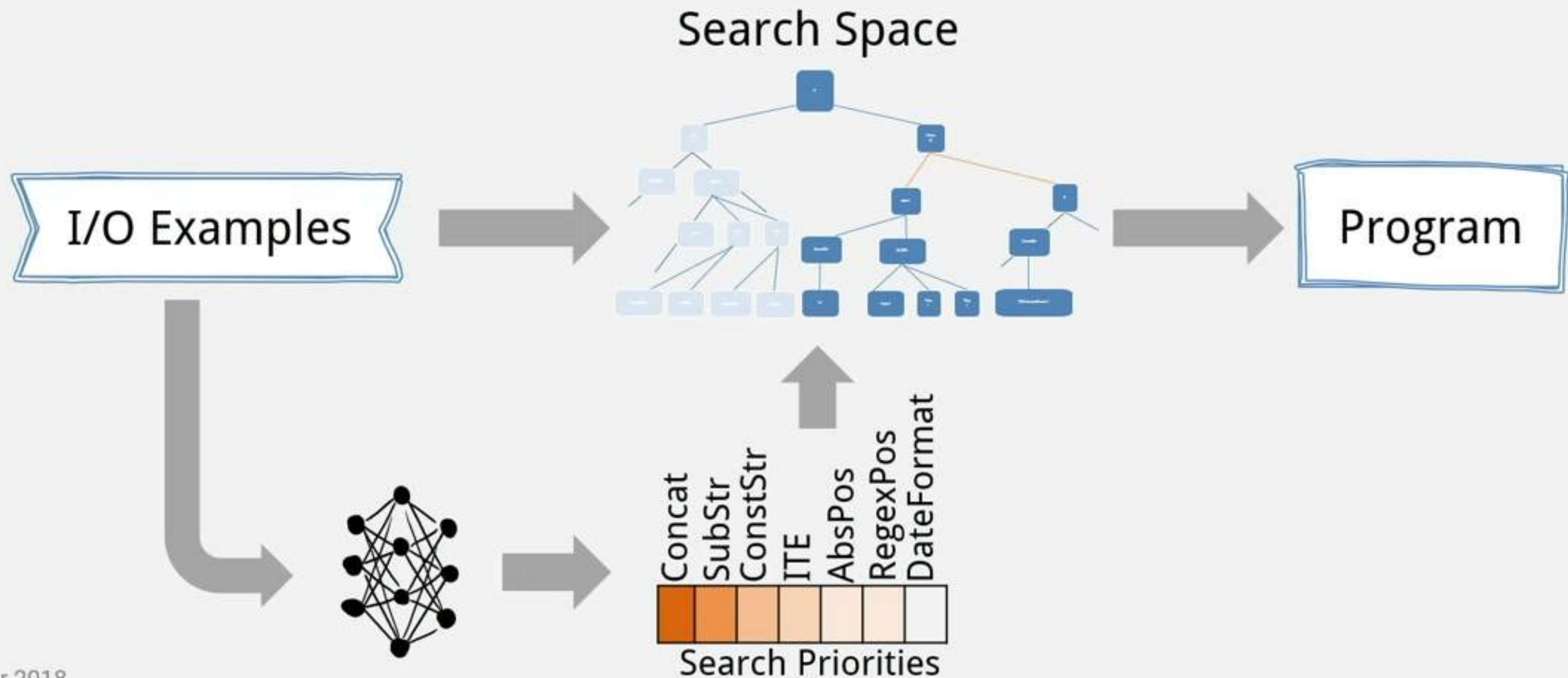




# Neural-Guided Deductive Search

[Kalyan et al., 2018]

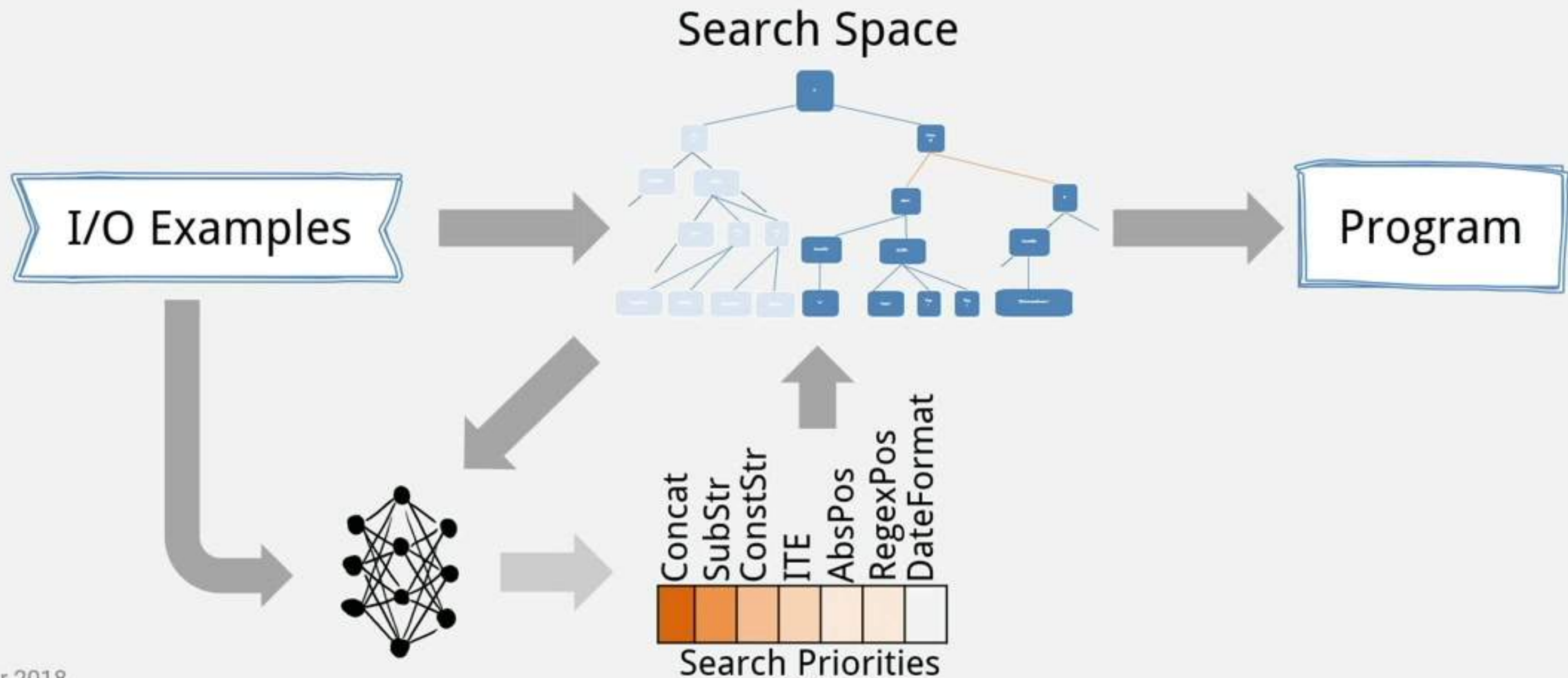
**Idea:** Order the search space based on a priority list from DNN *at each step*



# Neural-Guided Deductive Search

[Kalyan et al., 2018]

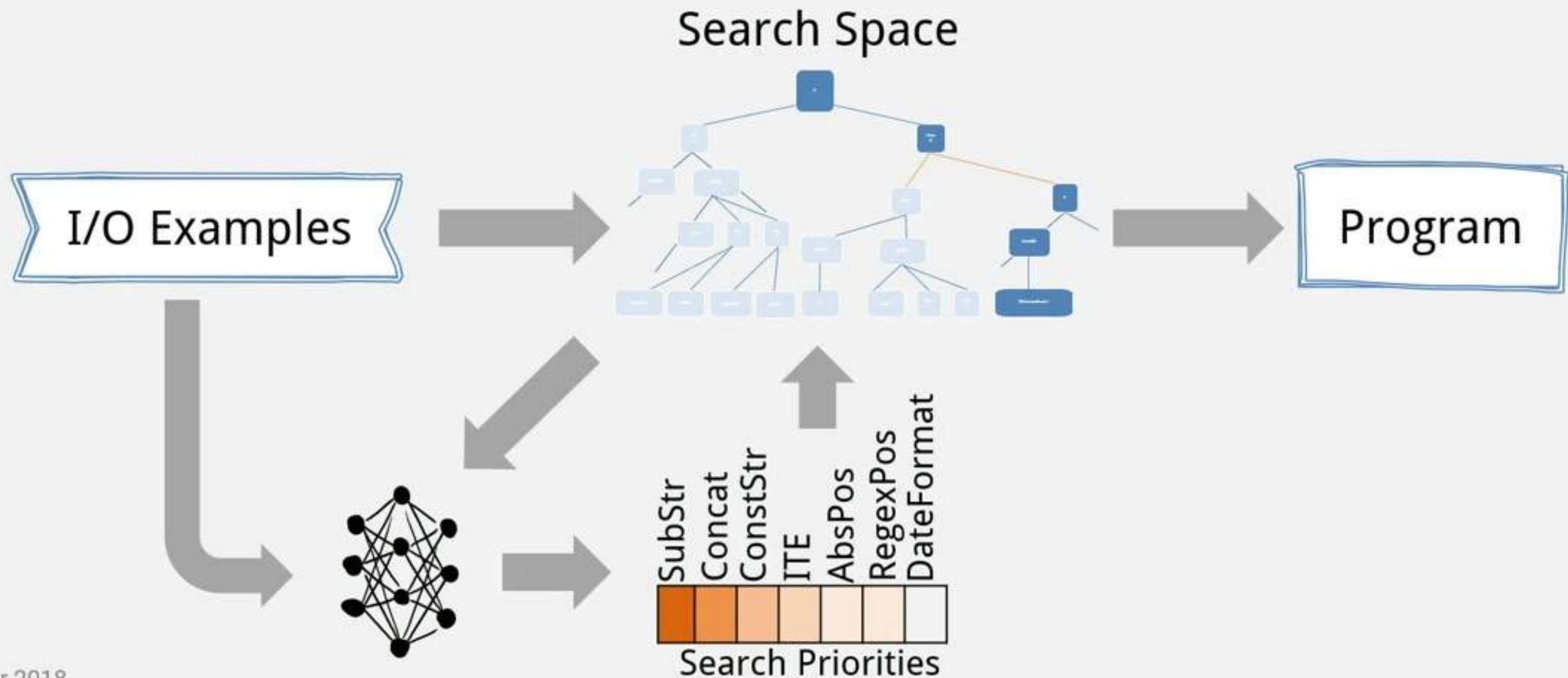
**Idea:** Order the search space based on a priority list from DNN *at each step*



# Neural-Guided Deductive Search

[Kalyan et al., 2018]

**Idea:** Order the search space based on a priority list from DNN *at each step*



# Search branch prediction

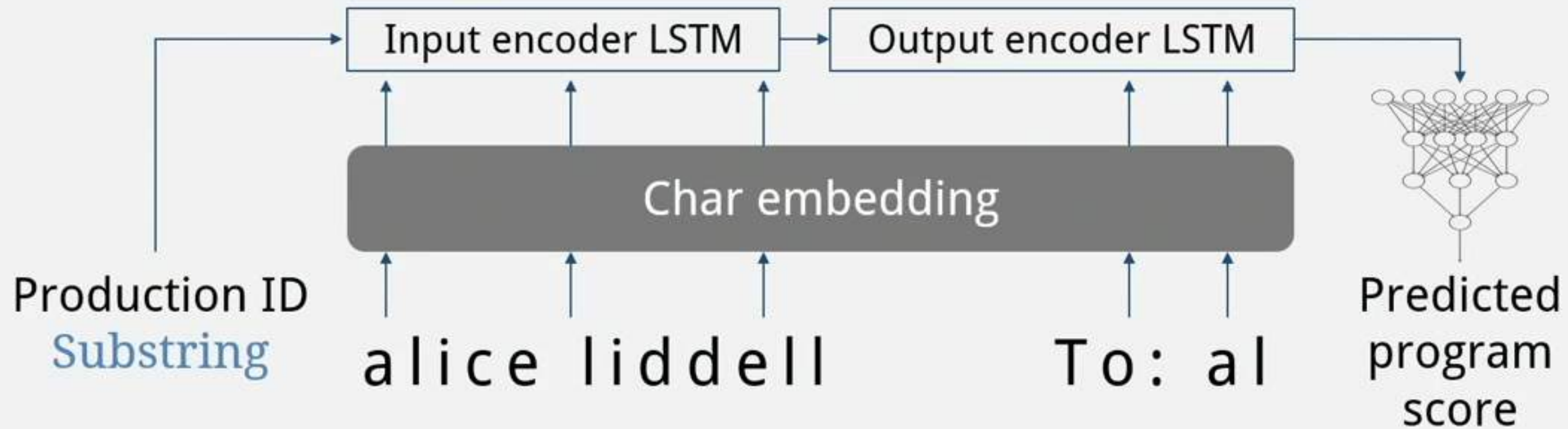
- Collect a complete dataset of intermediate search results:

at a search branch  $N := F_1(\dots) \mid F_2(\dots) \mid \dots \mid F_k(\dots)$   
given a spec  $\varphi = \{x \rightsquigarrow y\}$   
produced programs  $P_1, \dots, P_k$  with scores  $h(P_1, \varphi), \dots, h(P_k, \varphi)$

- Learn a predictive model  $f$  s.t.  $f(F_j, \varphi) \approx h(P_j, \varphi)$
- Train using squared-error loss over program scores:

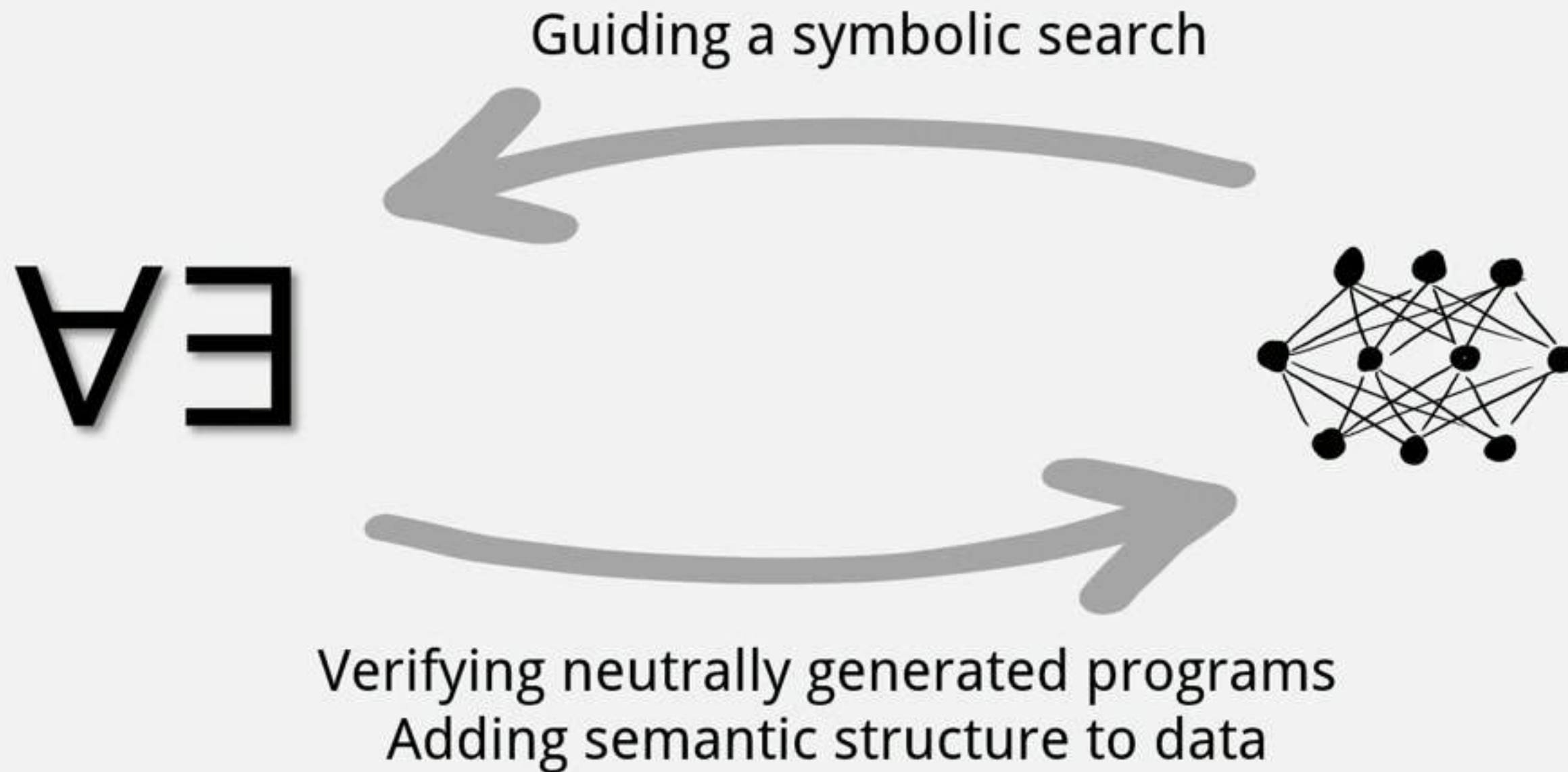
$$\mathcal{L}(f; F_j, \varphi) = [f(F_j, \varphi) - h(P_j, \varphi)]^2$$

# Model



- Guides the search toward higher-quality programs
- Eliminates unproductive search branches
- Balances performance vs. generalization using branch-and-bound

# “Neuro-Symbolic” Architectures



# Natural Language → SQL

Year	City	Country	Nations
1896	Athens	Greece	14
1900	Paris	France	24
1904	St. Louis	USA	12
...	...	...	...
2004	Athens	Greece	201
2008	Beijing	China	204
2012	London	UK	204

**Q:** When did Greece hold its last Summer Olympics?

```
SELECT MAX(Year) FROM SummerOlympics WHERE Country='Greece';
```

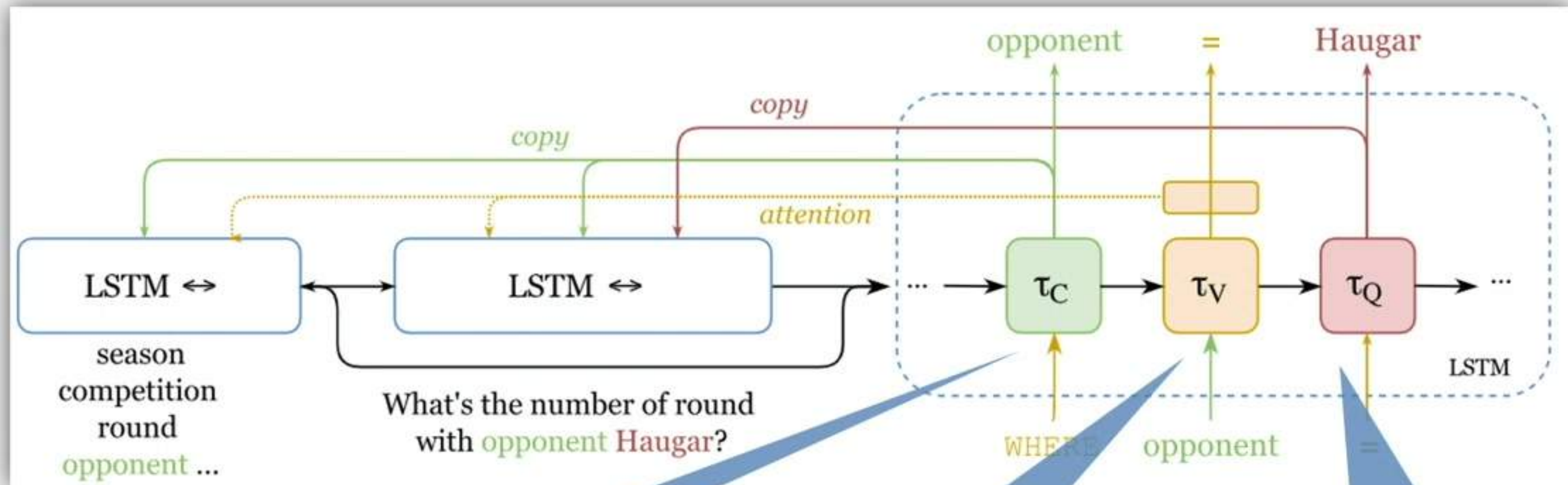
**A:** 2004

Relevant Information:

1. Column names
2. Column data types
3. Matches between table data and question

# Pointer-SQL

[Wang et al., 2017]



Copying mechanism  
for column names

Constant token

Copying mechanism  
for words/values



# Natural Language → SQL

Year	City	Country	Nations
1896	Athens	Greece	14
1900	Paris	France	24
1904	St. Louis	USA	12
...	...	...	...
2004	Athens	Greece	201
2008	Beijing	China	204
2012	London	UK	204

Q: When did Greece hold its last Summer Olympics?

```
SELECT MAX(Year) FROM SummerOlympics WHERE Country='Greece';
```

A: 2004

**Execution Guidance** during inference:

1. Backtrack if partial query has type error
2. Backtrack if "... WHERE COND" is empty
3. Constrain the decoder using grammar types
4. Nondeterministic oracles during training

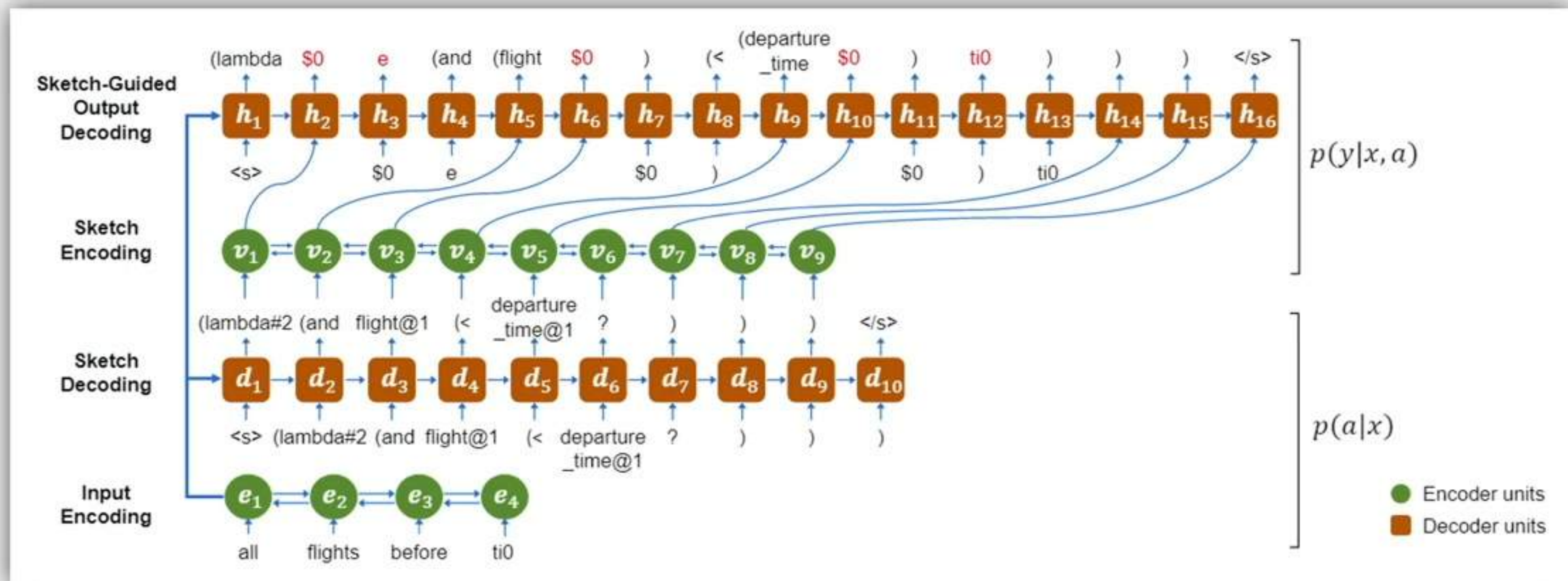
Relevant Information:

1. Column names
2. Column data types
3. Matches between table data and question



# Coarse→Fine sketch generation

[Solar-Lezama, 2006] [Murali et al., 2018] [Dong & Lapata, 2018]



# Outline

- Introduction
- Programming by Examples
- Neural-Guided Program Search
- **Neural Program Synthesis**

# Programs: the ML Perspective

**Approach 1: Sequence of Words**

(re-using NLP ideas)

# Programs: the ML Perspective

## **Approach 1: Sequence 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

# Programs: the ML Perspective

## **Approach 1.1: Sequence/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

# Programs: the ML Perspective

## **Approach 1.1: Sequence/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



# Programs: the ML Perspective

## **Approach 1.1: Sequence/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**

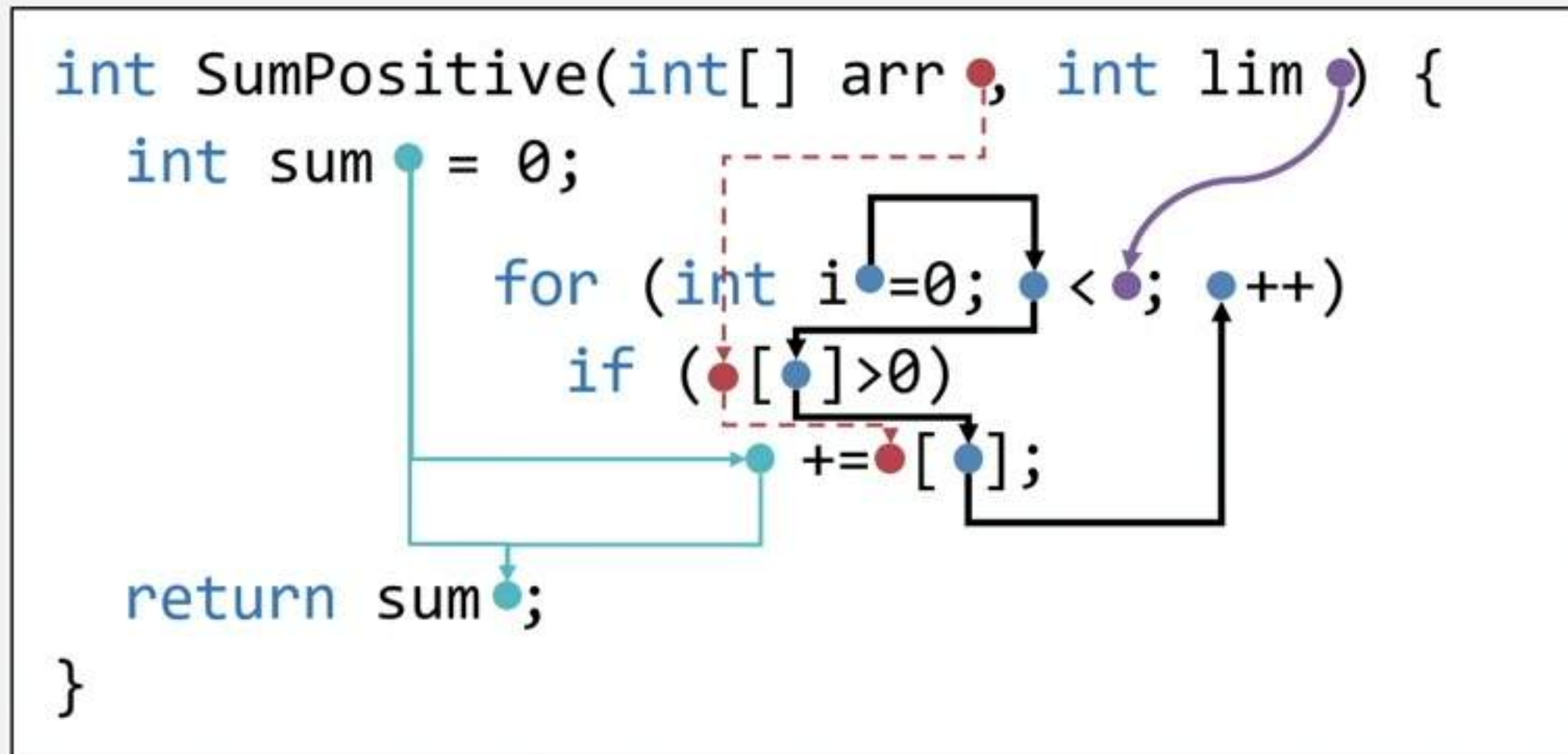
- Nodes labelled by semantic information
- Edges for semantic relationships

} From Static Analysis

# Programs as Graphs: Key Idea

```
int SumPositive(int[] arr, int lim) {  
    int sum = 0;  
    for (int i = 0; i < lim; i++)  
        if (arr[i] > 0)  
            sum += arr[i];  
  
    return sum;  
}
```

# Programs as Graphs: Key Idea

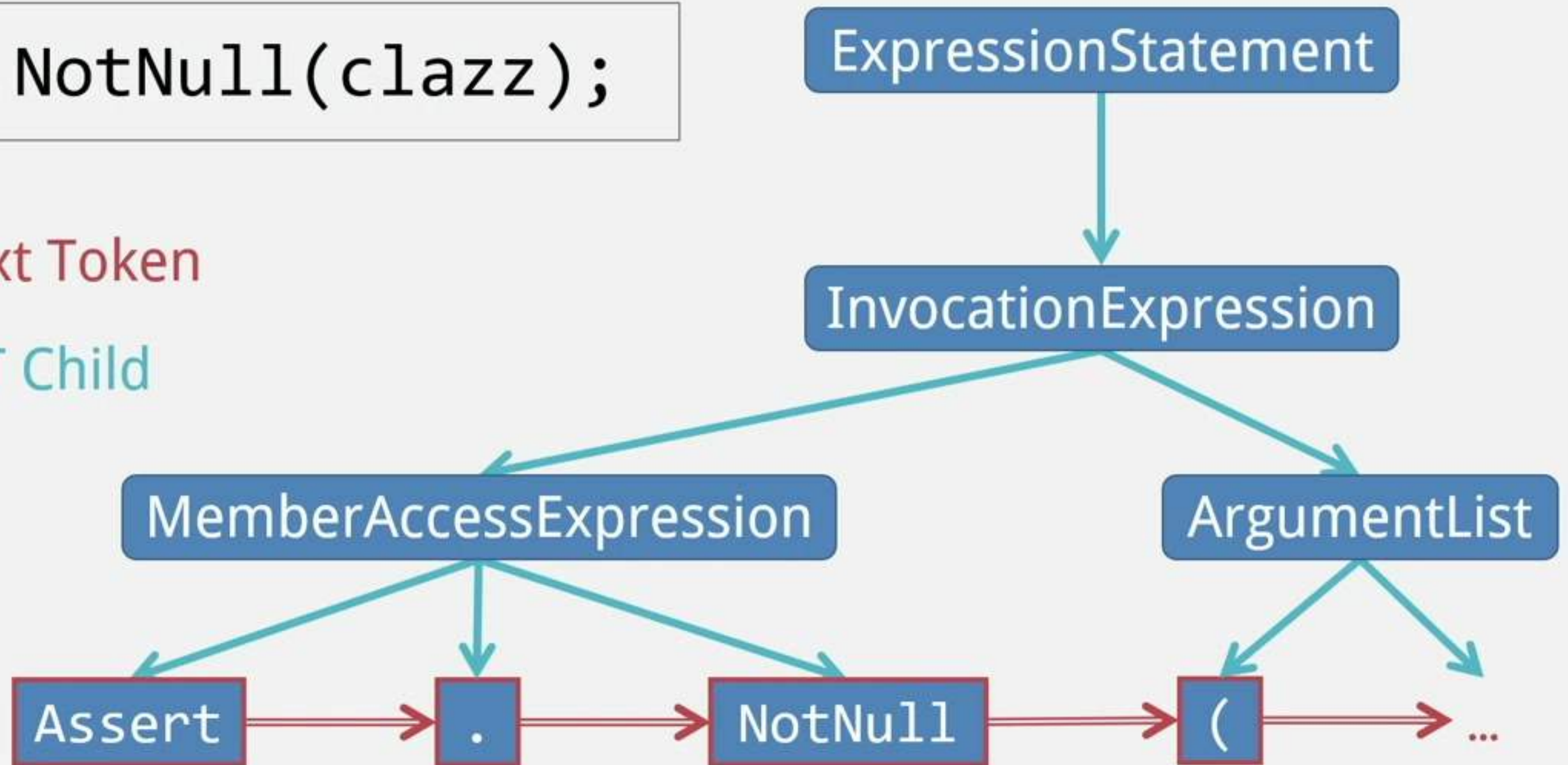


# Programs as Graphs: Syntax

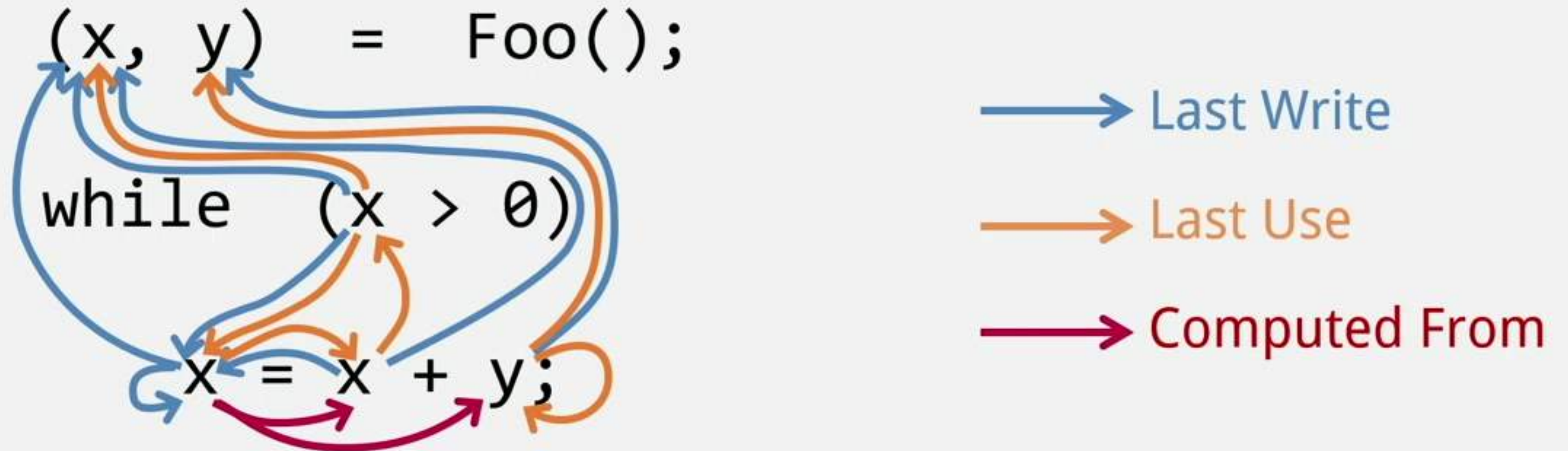
```
Assert.NotNull(clazz);
```

⇒ Next Token

→ AST Child

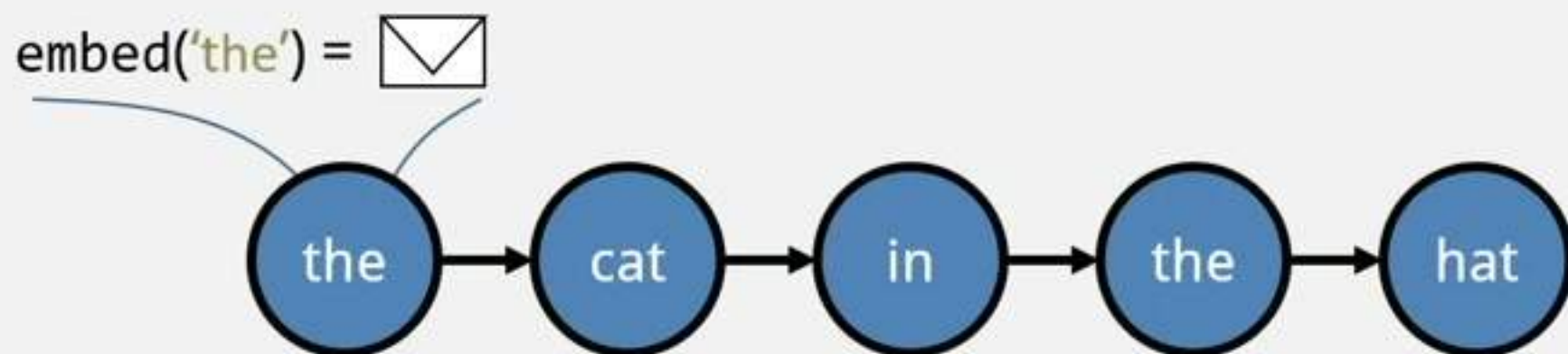


# Programs as Graphs: Data Flow




# Graph Neural Networks: Extending RNNs

[Li et al., 2016] [Allamanis et al., 2018]

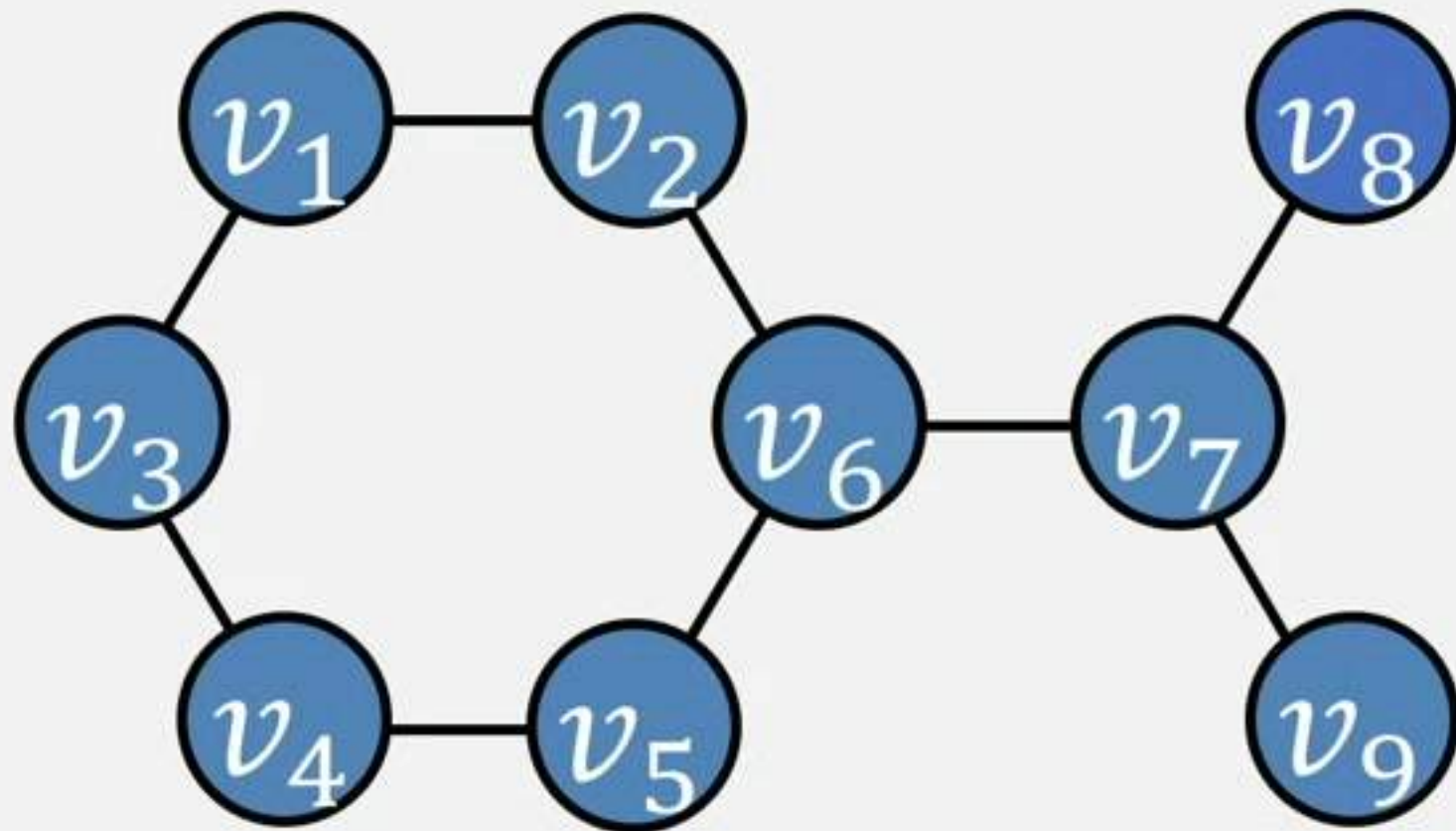


Chain structured data  
(e.g. text)

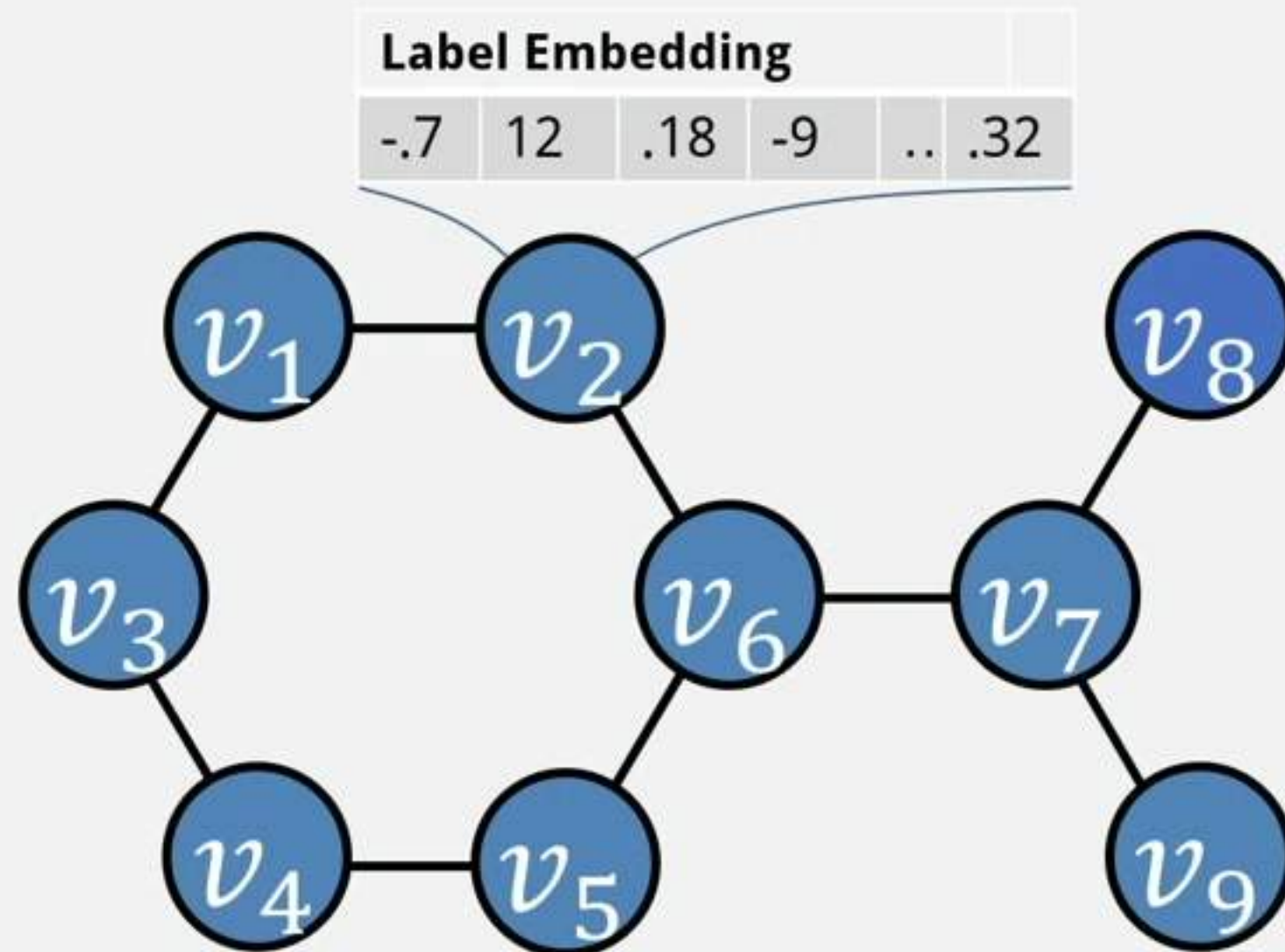


 Recurrent unit

# Graph Neural Networks: States

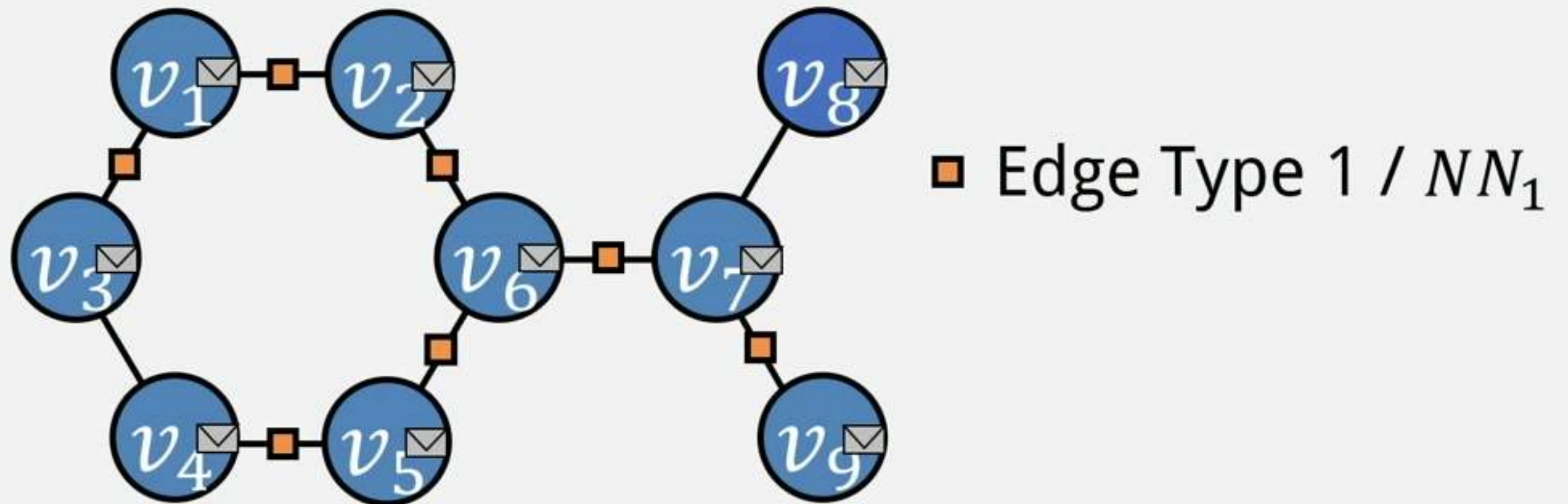


# Graph Neural Networks: States

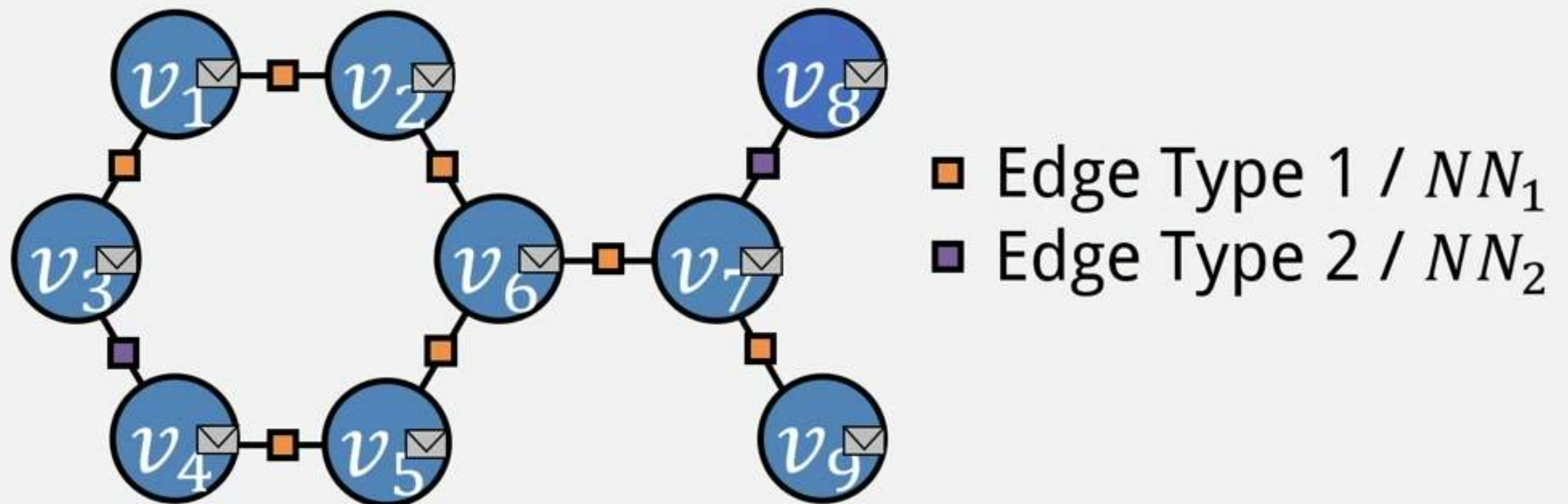




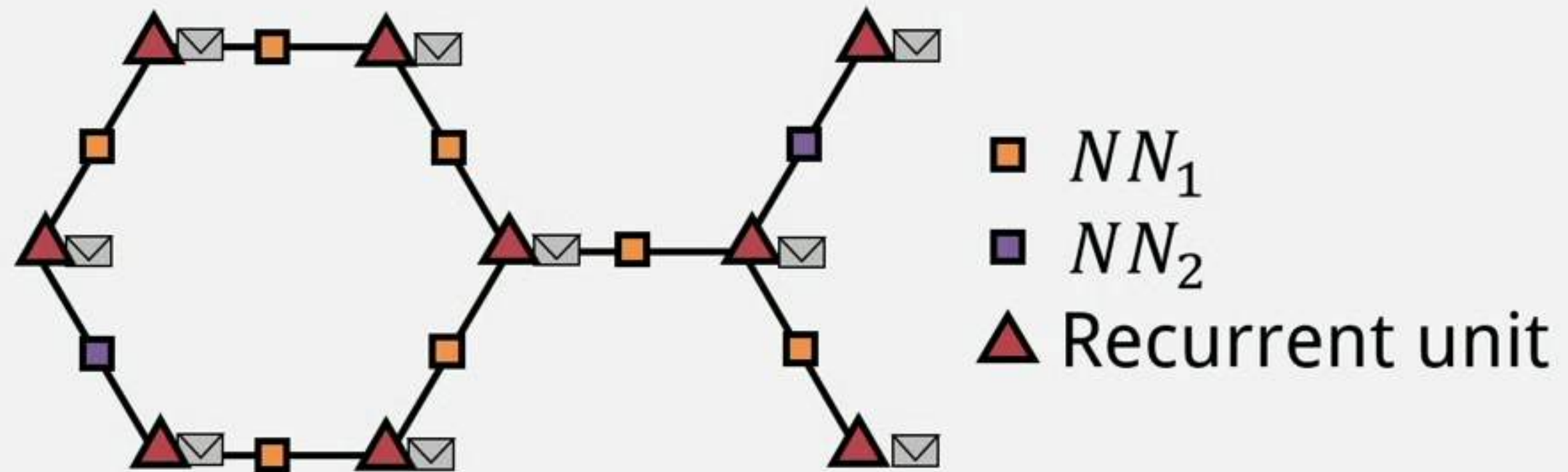
# Graph Neural Networks: States



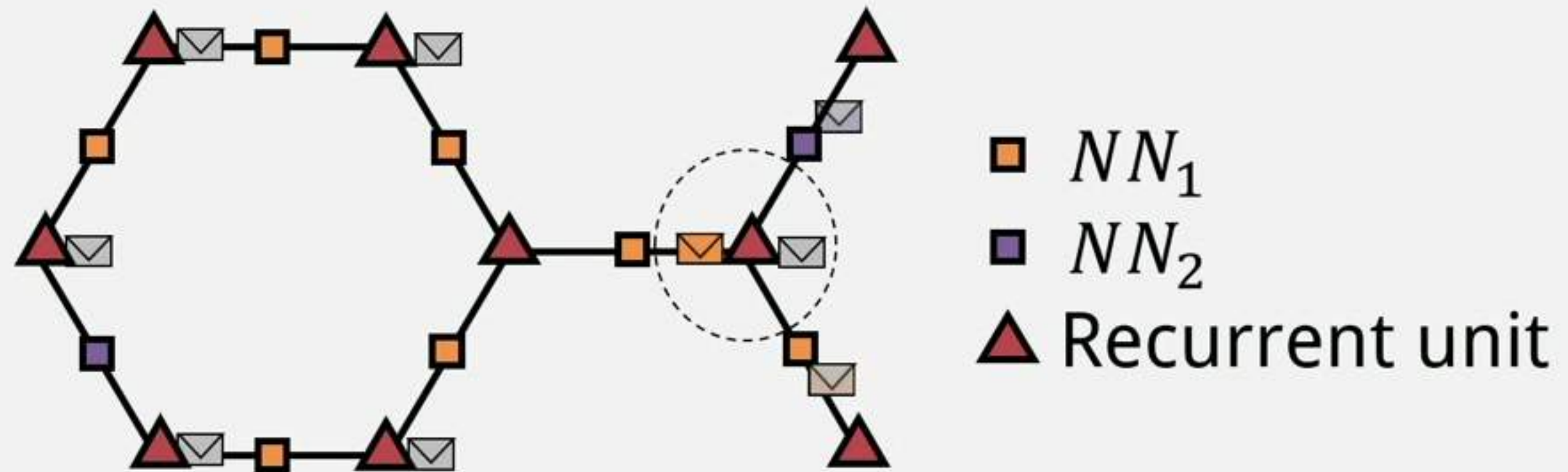
# Graph Neural Networks: States



# Graph Neural Networks: Propagation



# Graph Neural Networks: Propagation



# Expression Completion Task

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









# Expression Completion Task

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

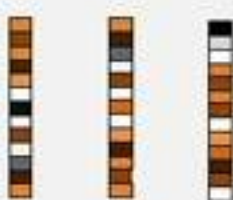
Code  
Context

“Classic”  
Graph  
Neural  
Network

Hole  
Representation



In-Context  
Variable  
Representations



Sequential  
Graph  
Neural  
Network

Generated  
Code

# Sequential Graph Neural Network

[Brockschmidt et al., 2018]

Expression

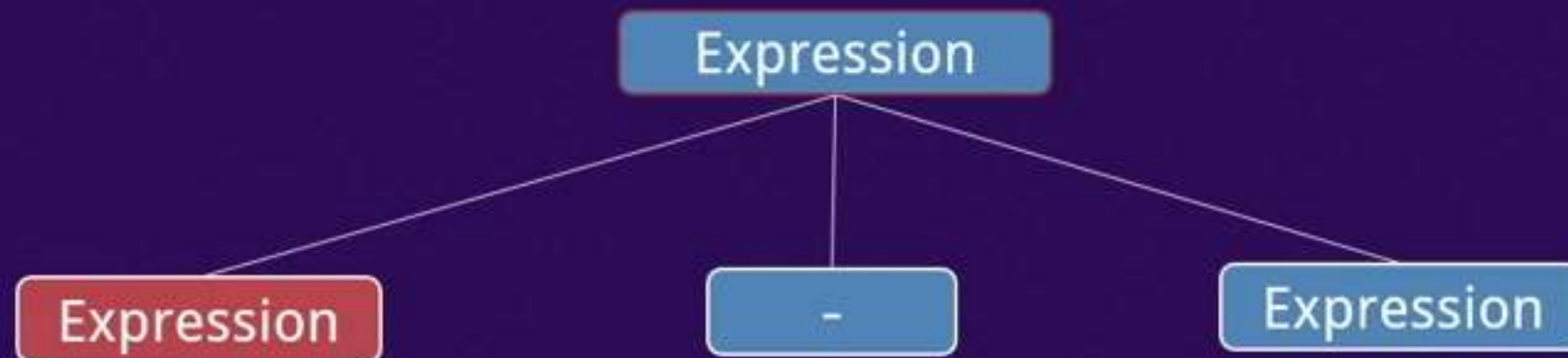
Variables in  
Scope

$i$

$j$

# Sequential Graph Neural Network

[Brockschmidt et al., 2018]



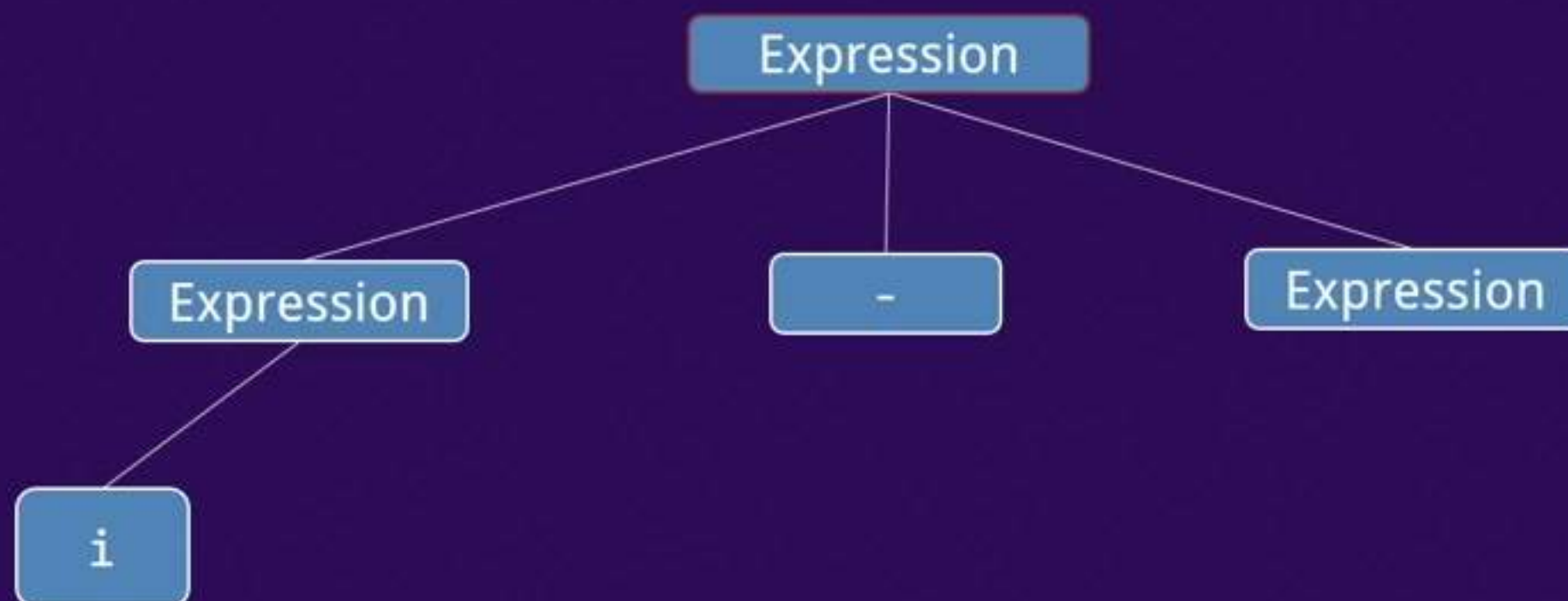
Variables in  
Scope

$i$

$j$

# Sequential Graph Neural Network

[Brockschmidt et al., 2018]



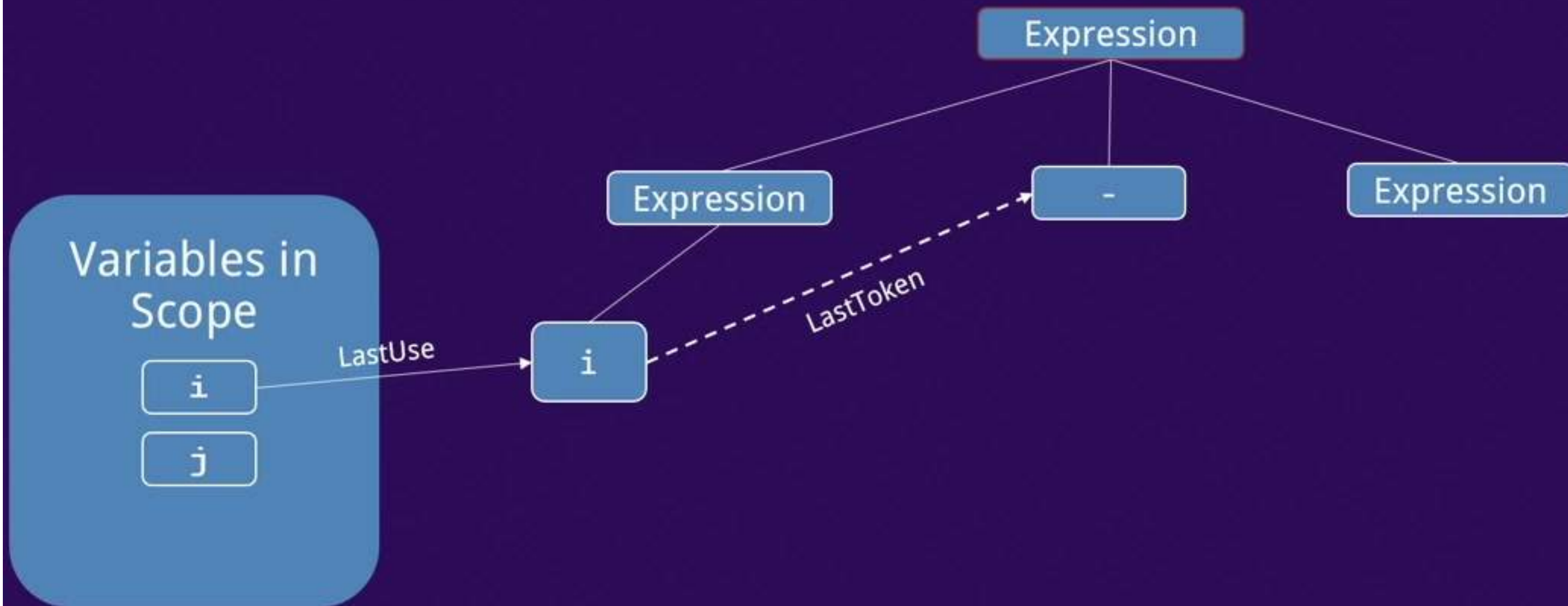
Variables in Scope

i

j

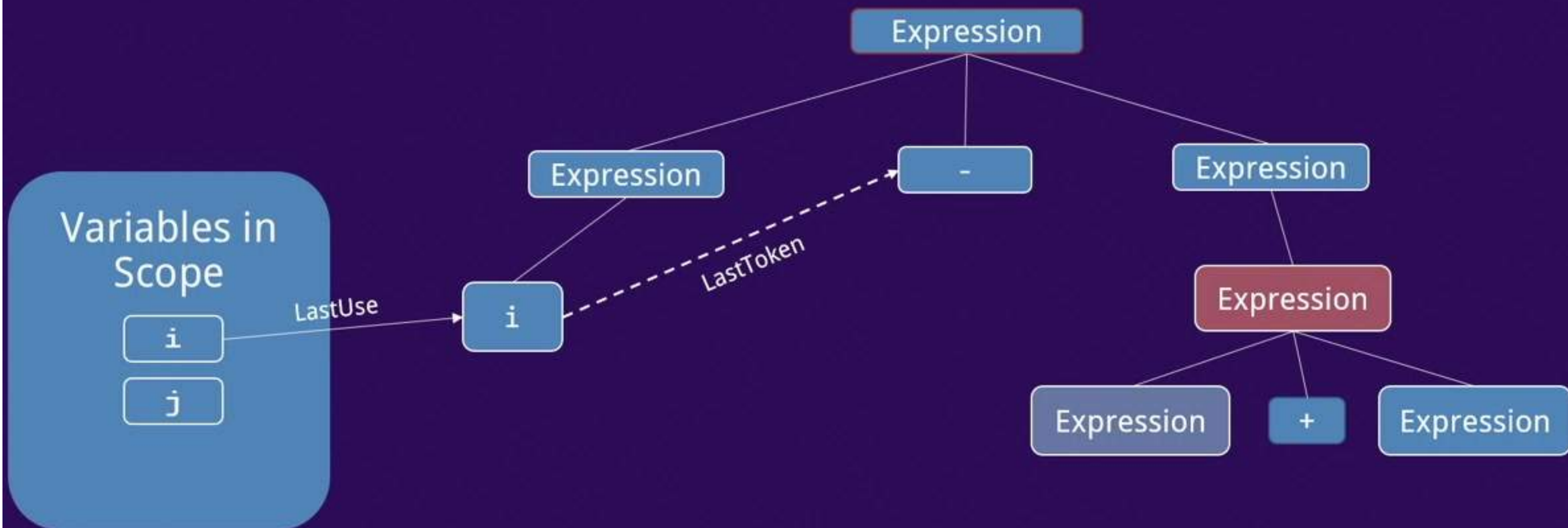
# Sequential Graph Neural Network

[Brockschmidt et al., 2018]



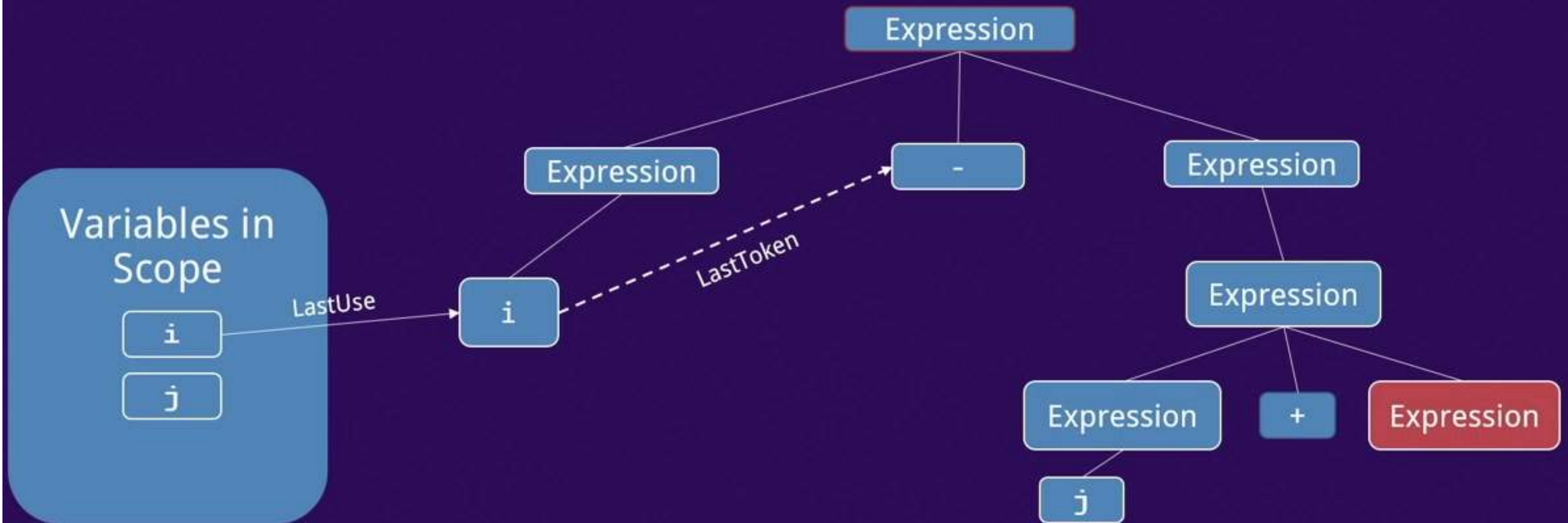
# Sequential Graph Neural Network

[Brockschmidt et al., 2018]



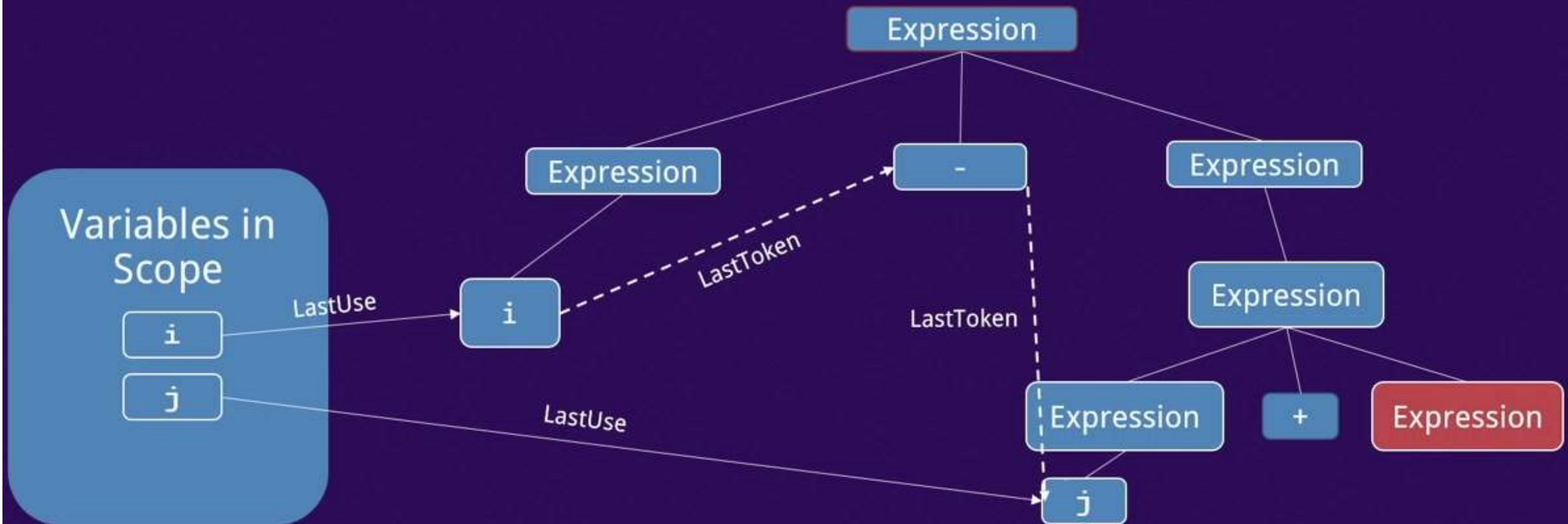
# Sequential Graph Neural Network

[Brockschmidt et al., 2018]



# Sequential Graph Neural Network

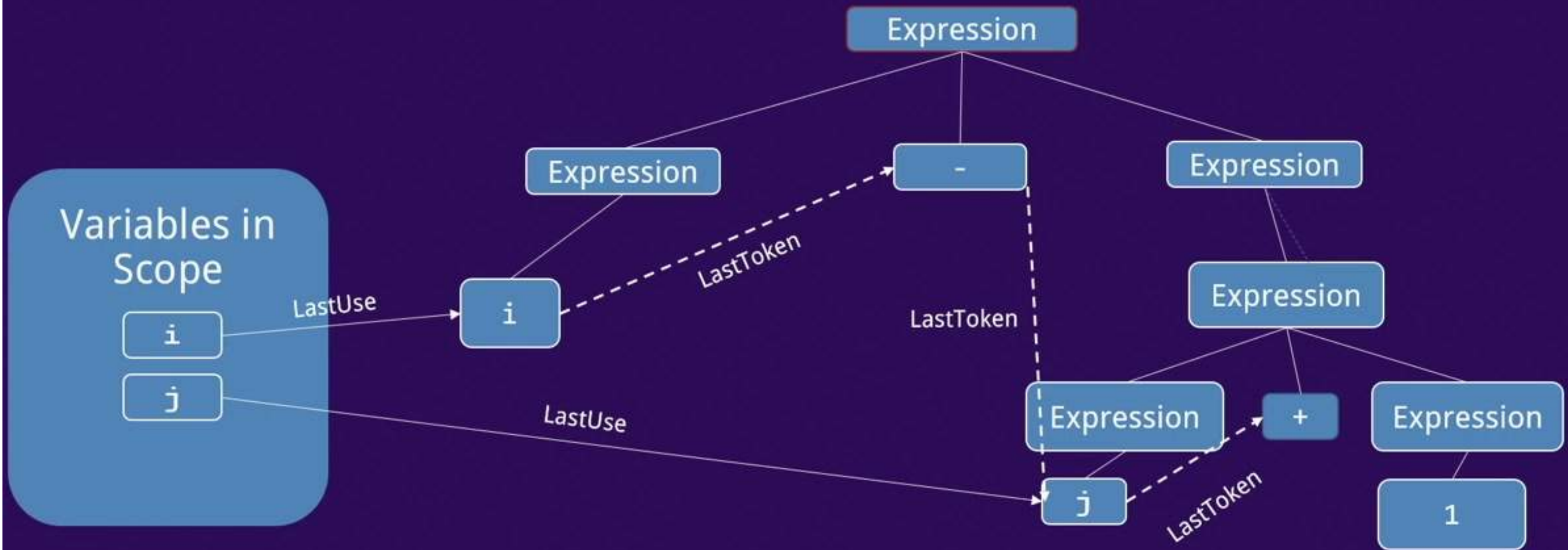
[Brockschmidt et al., 2018]





# Sequential Graph Neural Network

[Brockschmidt et al., 2018]



# Examples

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

$\mathcal{G} \rightarrow \mathcal{NAG}$ :

paramCount > methParamCount (34.4%)  
paramCount == methParamCount (11.4%)  
paramCount < methParamCount (10.0%)

```
caretPos--;
if (caretPos < 0) {
    caretPos = 0;
}

int len = inputString.Length;
if (caretPos >= len) {
    caretPos = len - 1;
}
```

$\mathcal{G} \rightarrow \mathcal{NAG}$ :

len++ (33.6%)  
len-1 (21.9%)  
len+1 (14.6%)

```
char character = originalName[i];
if (character == '<') {
    ++startTagCount;
    builder.Append(' ');
} else if (startTagCount > 0) {
    if (character == '>') {
        --startTagCount;
    }
}
```

$\mathcal{G} \rightarrow \mathcal{NAG}$ :

character == UNK\_CHAR\_LITERAL (75.5%)  
character == ' ' (2.6%)  
character != 'UNK\_CHAR' (2.5%)

# Application: Code generation

Program context/template



Programming language



```
import java.io.*;

object Main {
  def main(args:Array[String]) = {
    var body = "small.txt"
    var sig = "signature.txt"

    var inputStream:InputStream =
      new FileInputStream(body)
    var outputStream:OutputStream =
      new FileOutputStream(sig)
    var seqInputStream:InputStream =
      new SequenceInputStream(inputStream, outputStream)
    var seqOutputStream:OutputStream =
      new SequenceOutputStream(outputStream, seqInputStream)

    var eof:Boolean = false;
    var bytesCount:Int = 0;
    while (!eof) {
      var c:Char = seqInputStream.read()
      if (c == -1)
        eof = true;
      else {
        System.out.println(c.toChar);
        bytesCount += 1;
      }
    }
    System.out.println(bytesCount + " bytes were read");
    seqInputStream.close();
  }
}
```

# Further reading

Gulwani, S., Polozov, O., and Singh, R. (2017). Program synthesis. *Foundations and Trends in Programming Languages*, 4(1-2), 1-119.

Microsoft PROSE SDK. <https://microsoft.github.io/prose/>

Bornholt, J. Program synthesis explained.  
<https://homes.cs.washington.edu/~bornholt/post/synthesis-explained.html>

Polozov, O. Program Synthesis in 2017-18.  
<https://alexpolozov.com/blog/program-synthesis-2018/>

Solar-Lezama, A. The Sketch Programmers Manual.

# Takeaways

- Program synthesis = Language + Spec + Search algorithm
- For PBE, enumerative/deductive search ensures correctness
  - Performance improved using neural-guided search
- For other specs (language, context), use graph-based or recurrent neural networks
  - Ensure correctness using execution guidance & program structure
- Solicit or learn hints: sketches, grammars, ranking functions
- Make use of existing frameworks: PROSE, Sketch, Rosette