Learning to Infer Program Sketches

Maxwell Nye, Luke Hewitt, Josh Tenenbaum, Armando Solar-Lezama

Goal: We want to automatically write code from the kinds of specifications humans can easily provide, such as examples or natural language instruction.

List Processing from IO:

```
[1, 2, 3, 4, 5] → [2, 4]
[7, 8, 0, 9] → [8, 0]
```

Text Editing from IO:

```
Max Nye → Nye, M.
Luke Hewitt → Hewitt, L.
```

Natural language + IO → code

```
“Consider an array of numbers,
find elements in the given array
not divisible by two”
[1, 2, 3, 4, 5] → [1, 3, 5]
[7, 8, 0, 9] → [7, 9]
```

Learning to Infer Program Sketches

Maxwell Nye, Luke Hewitt, Josh Tenenbaum, Armando Solar-Lezama

Given:

```
[1, 2, 3, 4, 5] → [2, 4]
[0, 6, 2, 7] → [0, 6, 2]
[5, 10, 5, 1, 8] → [10, 8]
```

Goal: Write a program which maps inputs to outputs

How might people solve problems like this?

Learning to Infer Program Sketches

Maxwell Nye, Luke Hewitt, Josh Tenenbaum, Armando Solar-Lezama

```
Given:
[1, 2, 3, 4, 5] → [2, 4]
[0, 6, 2, 7] → [0, 6, 2]
[5, 10, 5, 1, 8] → [10, 8]
```

```
Goal: Write a program
which maps inputs to
outputs
```

How might people solve problems like this?

People use a flexible trade-off between **pattern recognition** and **reasoning**

Learning to Infer Program Sketches

Maxwell Nye, Luke Hewitt, Josh Tenenbaum, Armando Solar-Lezama

Easy problem:

```
Spec:
[1, 2, 3, 4, 5] → [2, 4]
[0, 6, 2, 7] → [0, 6, 2]
[5, 10, 5, 1, 8] → [10, 8]

Solution:
```

Learning to Infer Program Sketches

Maxwell Nye, Luke Hewitt, Josh Tenenbaum, Armando Solar-Lezama

Easy problem:

Spec:

```
[1, 2, 3, 4, 5] → [2, 4]
[0, 6, 2, 7] → [0, 6, 2]
[5, 10, 5, 1, 8] → [10, 8]
```

Solution:

```
filter(lambda x: x%2==0, input)
```

Learning to Infer Program Sketches

Maxwell Nye, Luke Hewitt, Josh Tenenbaum, Armando Solar-Lezama

Easy problem:

Spec:

```
[1, 2, 3, 4, 5] → [2, 4]
[0, 6, 2, 7] → [0, 6, 2]
[5, 10, 5, 1, 8] → [10, 8]
```

Solution:

```
filter(lambda x: x%2==0, input)
```

Fast, using pattern recognition

Learning to Infer Program Sketches

Maxwell Nye, Luke Hewitt, Josh Tenenbaum, Armando Solar-Lezama

More difficult problem:

```
Spec:
[3, 4, 5, 6, 7] → [4, 7]
[10, 8, 7, 3, 2, 1] → [10, 7, 1]
[5, 1, 2, 13, 4] → [1, 13, 4]

Solution:
```

Learning to Infer Program Sketches

Maxwell Nye, Luke Hewitt, Josh Tenenbaum, Armando Solar-Lezama

More difficult problem:

Spec:

```
[3, 4, 5, 6, 7] → [4, 7]
[10, 8, 7, 3, 2, 1] → [10, 7, 1]
[5, 1, 2, 13, 4] → [1, 13, 4]
```

Solution:

```
filter(<SOMETHING>, input)
```

(Fast, using pattern recognition)

Learning to Infer Program Sketches

Maxwell Nye, Luke Hewitt, Josh Tenenbaum, Armando Solar-Lezama

More difficult problem:

Spec:

```
[3, 4, 5, 6, 7] → [4, 7]
[10, 8, 7, 3, 2, 1] → [10, 7, 1]
[5, 1, 2, 13, 4] → [1, 13, 4]
```

Solution:

```
filter(<SOMETHING>, input)
```

Symbolic reasoning

```
filter(lambda x: x%3==1,
       input)
```

(Fast, using pattern recognition)

Learning to Infer Program Sketches

Maxwell Nye, Luke Hewitt, Josh Tenenbaum, Armando Solar-Lezama

More difficult problem:

Spec:

```
[3, 4, 5, 6, 7] → [4, 7]
[10, 8, 7, 3, 2, 1] → [10, 7, 1]
[5, 1, 2, 13, 4] → [1, 13, 4]
```

Solution:

```
filter(<SOMETHING>, input)
```

Symbolic reasoning

```
filter(lambda x: x%3==1,
       input)
```

(Fast, using pattern recognition)

(Slow)

Learning to Infer Program Sketches

Maxwell Nye, Luke Hewitt, Josh Tenenbaum, Armando Solar-Lezama

Very difficult problem:

Spec:

```
[2, 5, 0, 16, 12] → 0
[4, 23, 11, 9, 25] → 25
[3, 29, 30, 14, 16] → 14
```

Learning to Infer Program Sketches

Maxwell Nye, Luke Hewitt, Josh Tenenbaum, Armando Solar-Lezama

Very difficult problem:

Spec:

```
[2, 5, 0, 16, 12] → 0
[4, 23, 11, 9, 25] → 25
[3, 29, 30, 14, 16] → 14
[1, 7, 6, 9, 5] → 7
[5, 5, 1, 8, 8, 12, 4] → 12
[0, 4, 8, 5, 1] → 0
[3, 7, 2, 9, 1] → 9
[1, 0, 3, 7, 3, 8] → 0
```

Learning to Infer Program Sketches

Maxwell Nye, Luke Hewitt, Josh Tenenbaum, Armando Solar-Lezama

Very difficult problem:

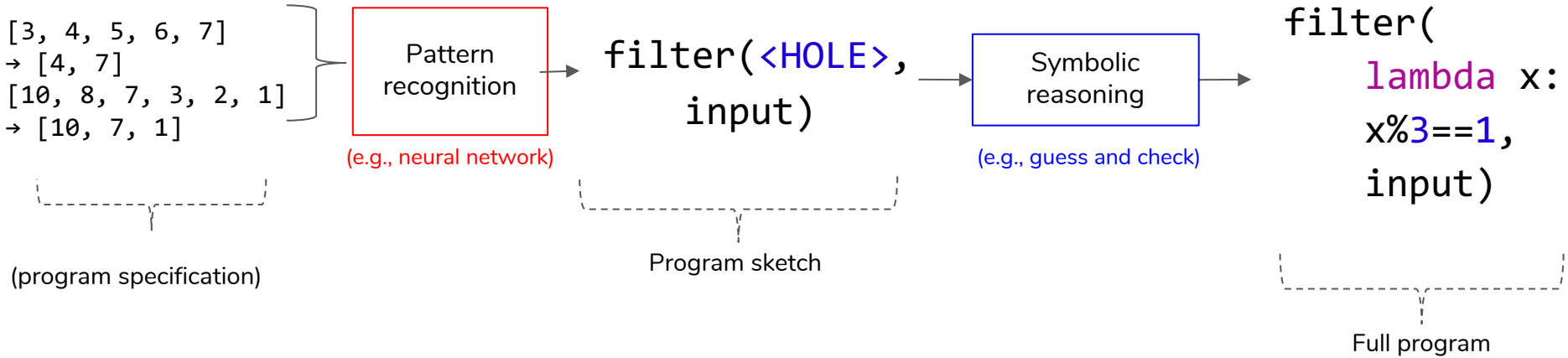
```
Spec:
[2, 5, 0, 16, 12] → 0
[4, 23, 11, 9, 25] → 25
[3, 29, 30, 14, 16] → 14

Solution:
      <SOMETHING>
```

Symbolic reasoning

(Slow)

```
input[input[0]]
```

Q: How do we model this? A: Program sketches



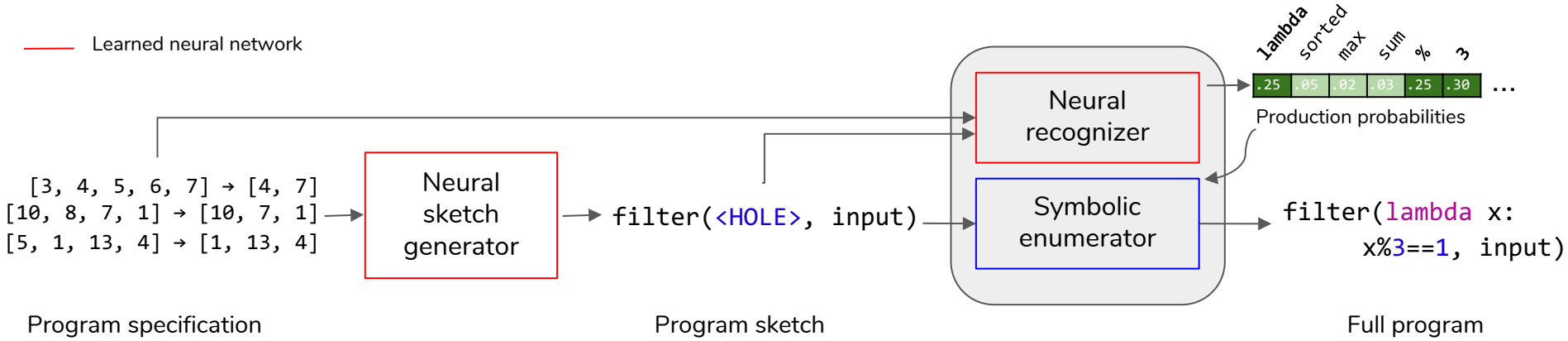
Flexible trade-off between pattern recognition and reasoning

Solar-Lezama et al, 2008,
Murali et al, 2017

Learning to Infer Program Sketches

Maxwell Nye, Luke Hewitt, Josh Tenenbaum, Armando Solar-Lezama

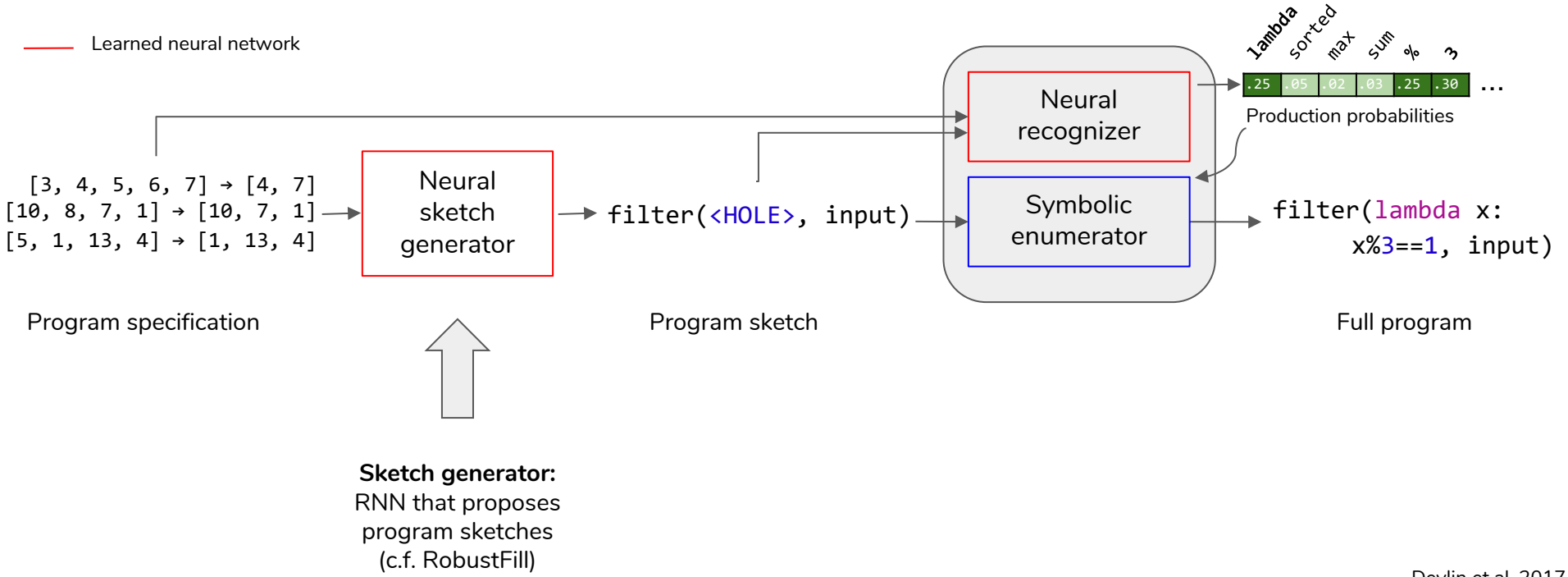
Our system: **SketchAdapt**



Learning to Infer Program Sketches

Maxwell Nye, Luke Hewitt, Josh Tenenbaum, Armando Solar-Lezama

Our system: **SketchAdapt**

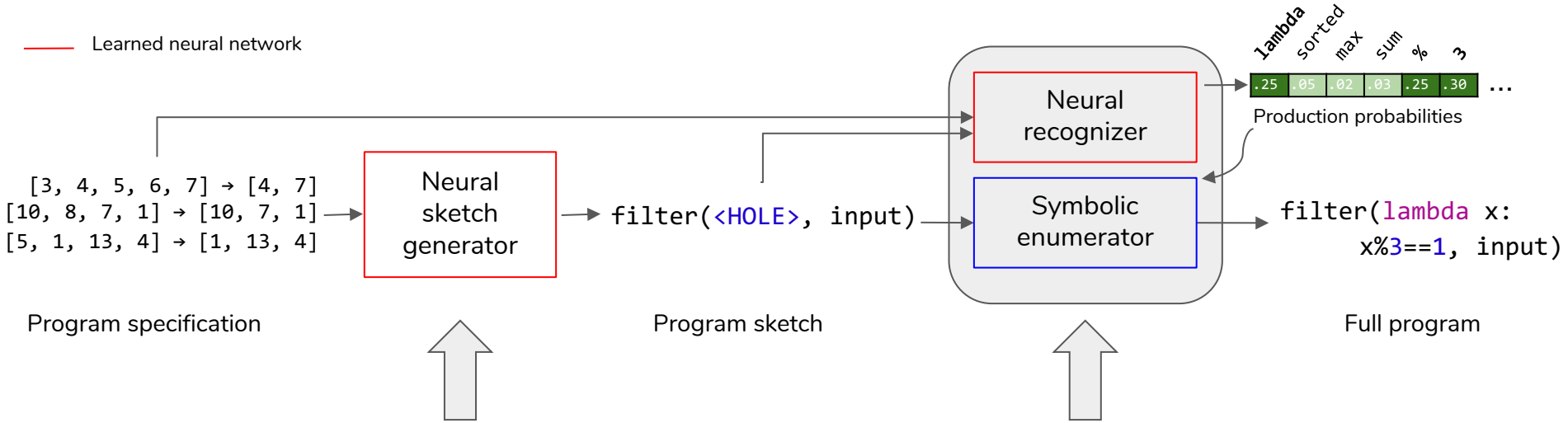


Devlin et al, 2017
Balog et al, 2016

Learning to Infer Program Sketches

Maxwell Nye, Luke Hewitt, Josh Tenenbaum, Armando Solar-Lezama

Our system: **SketchAdapt**



Sketch generator: RNN that proposes program sketches (c.f. RobustFill)

Symbolic synthesizer: **enumerator** that fills in sketches, guided by **neural recognizer** (c.f DeepCoder)

Devlin et al, 2017
Balog et al, 2016

Results: list processing

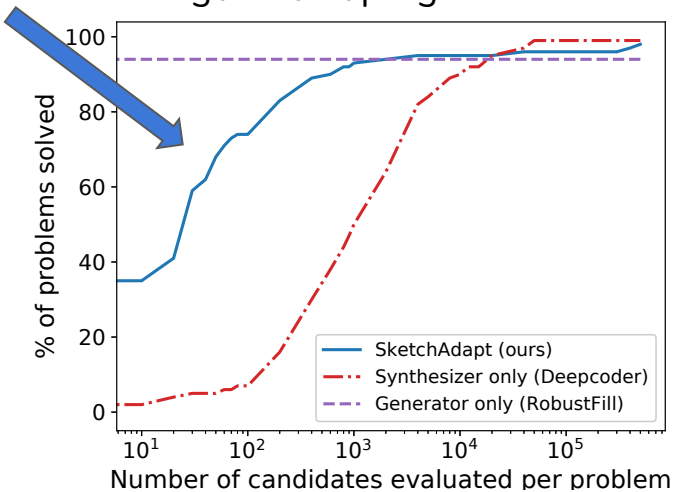
Ours
Pattern recognition only (neural network)
Reasoning only (symbolic enumeration)

SketchAdapt can recognize familiar problems and generalize to unfamiliar problems

Trained on length 3 programs

SketchAdapt

Length 3 test programs:



Results: list processing

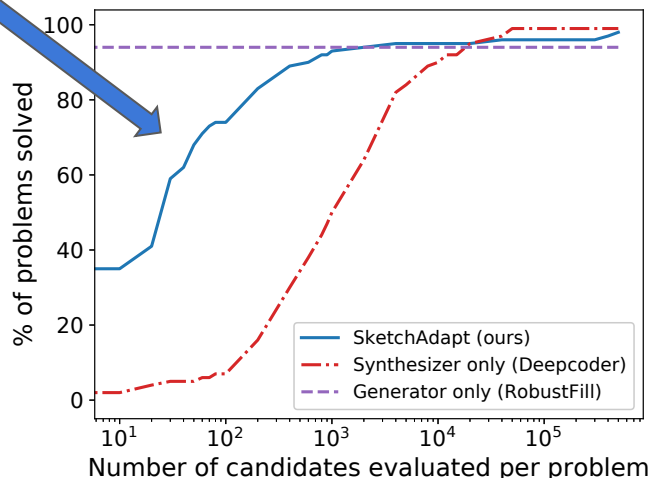
Ours
Pattern recognition only (neural network)
Reasoning only (symbolic enumeration)

SketchAdapt can recognize familiar problems and generalize to unfamiliar problems

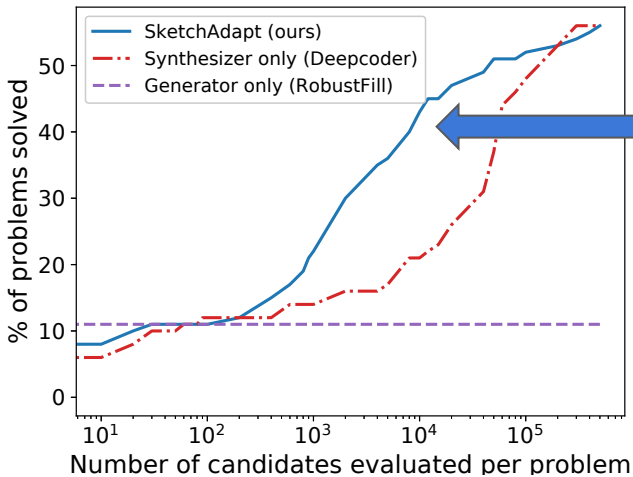
Trained on length 3 programs

SketchAdapt

Length 3 test programs:



Length 4 test programs:



SketchAdapt

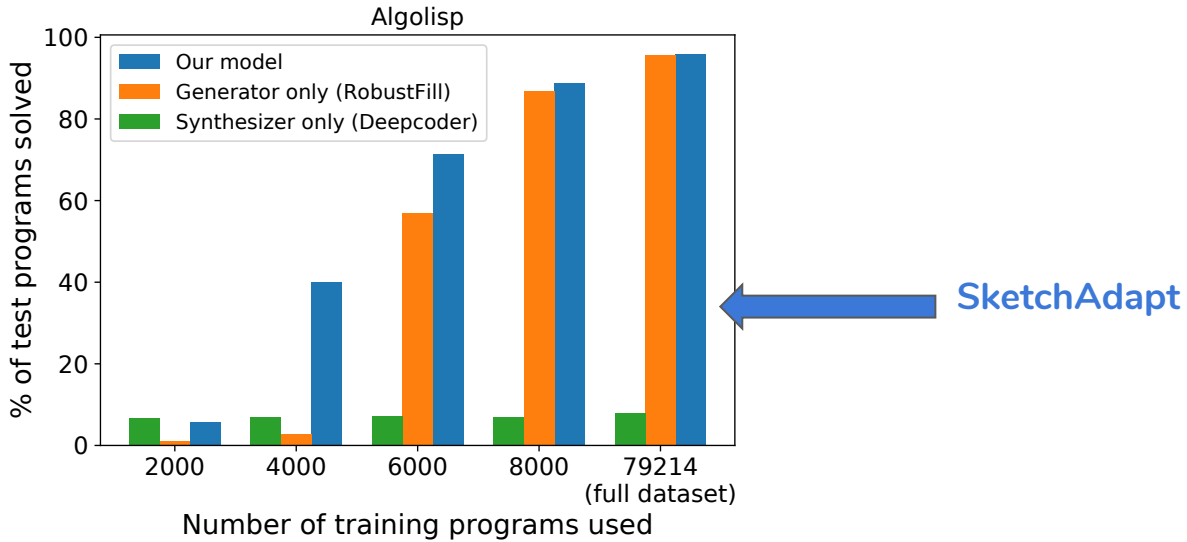
Natural language + IO examples → Code

Spec	Program
Consider an array of numbers, find elements in the given array not divisible by two	`( filter a ( lambda1 ( == (` `% arg1 2 ) 1)))`

You are given an array of numbers,
your task is to compute median
in the given array with its digits reversed
	`(reduce(reverse(digits(deref (sort a)`
	`(/ (len a) 2)))) 0`
	`(lambda2 (+(* arg1 10) arg2)))`

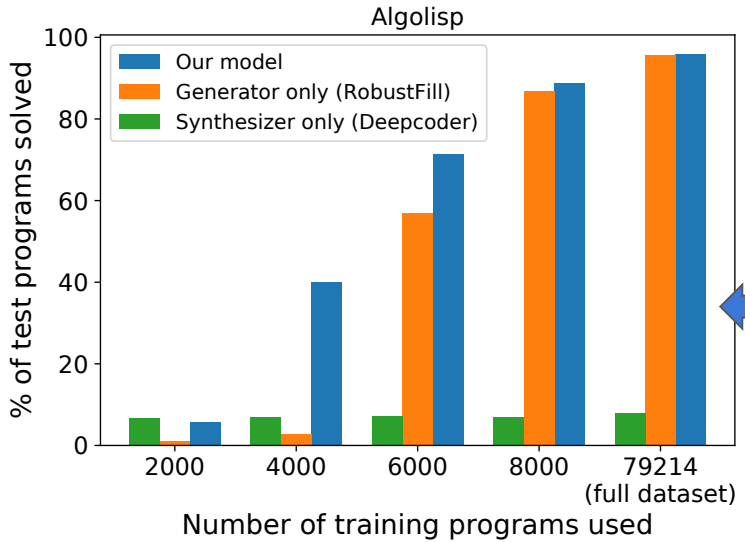
Natural language + IO examples → Code

Requires less data than pure neural approaches:



Natural language + IO examples → Code

Requires less data than pure neural approaches:



Generalizes to unseen concepts:

Table 5. Algolisp generalization results: Trained on 8000 programs, excluding 'Odd' concept:

Model	Even	Odd
SKETCHADAPT (Ours)	**34.4**	**29.8**
Synthesizer only	23.7	0.0
Generator only	4.5	1.1

SketchAdapt

Learning to Infer Program Sketches

Maxwell Nye, Luke Hewitt, Josh Tenenbaum, Armando Solar-Lezama

Come see our poster:

Today (Thurs) 06:30 - 09:00 PM @ Pacific Ballroom #182