2024. 04. 30. Paper Presentation Talk

Optimal Program Synthesis via Abstract Interpretation

Geon Park
KAIST Programming Systems Laboratory

Motivation

- FlashFill

	A	B
1	Name	First Name
2	Homer Simpson	Homer
3	Marge Simpson	Marge
4	Bart Simpson	

Examples

	A	B
1	Name	First Name
2	Homer Simpson	Homer
3	Marge Simpson	Marge
4	Bart Simpson	Bart

Fill in blanks

Motivation

- FlashFill

	A	B
1	Name	First Name
2	Homer Simpson	Homer
3	Marge Simpson	Marge
4	Bart Simpson	

Examples

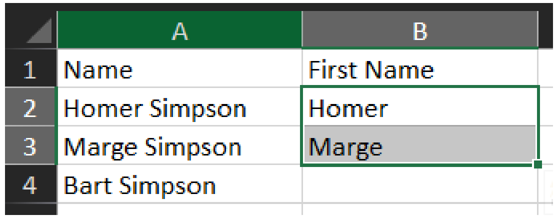
	A	B
1	Name	First Name
2	Homer Simpson	Homer
3	Marge Simpson	Marge
4	Bart Simpson	Bart

Fill in blanks

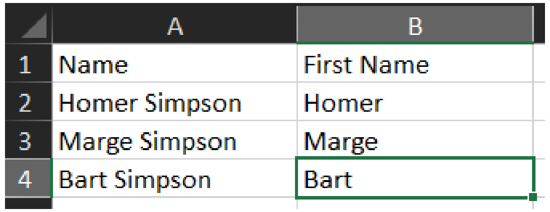
= LEFT(A\$1, FIND(" ", A\$1)-1)

Motivation

- FlashFill is program synthesis and it’s useful



	A	B
1	Name	First Name
2	Homer Simpson	Homer
3	Marge Simpson	Marge
4	Bart Simpson	



	A	B
1	Name	First Name
2	Homer Simpson	Homer
3	Marge Simpson	Marge
4	Bart Simpson	Bart

Examples → Synthesize program → Fill in blanks

= LEFT(A\$1, FIND(" ", A\$1)-1)

Program Synthesis

- Program synthesis is robust and interpretable (v.s. machine learning)

Program synthesis

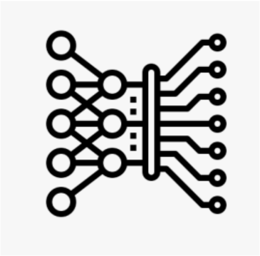
Data	
x	out
0	1
1	2
2	3
3	4
4	5

= x + 1

Data	
x	out
0	1
1	2
2	3
3	4
4	5
5	6
6	7

Machine learning

Data	
x	out
0	1
1	2
2	3
3	4
4	5



Data	
x	out
0	1
1	2
2	3
3	4
4	5
5	6
6	7

Program Synthesis

- Program synthesis is robust and interpretable (v.s. machine learning)

Program synthesis

Data	
x	out
0	1
1	2
2	3
3	4
4	5

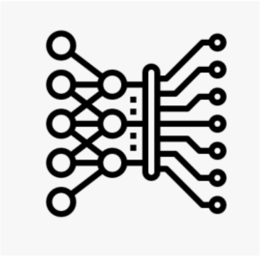
= x + 1

Data	
x	out
0	1
1	2
2	3
3	4
4	5
5	6
6	7

How?

Machine learning

Data	
x	out
0	1
1	2
2	3
3	4
4	5



Data	
x	out
0	1
1	2
2	3
3	4
4	5
5	6
6	7

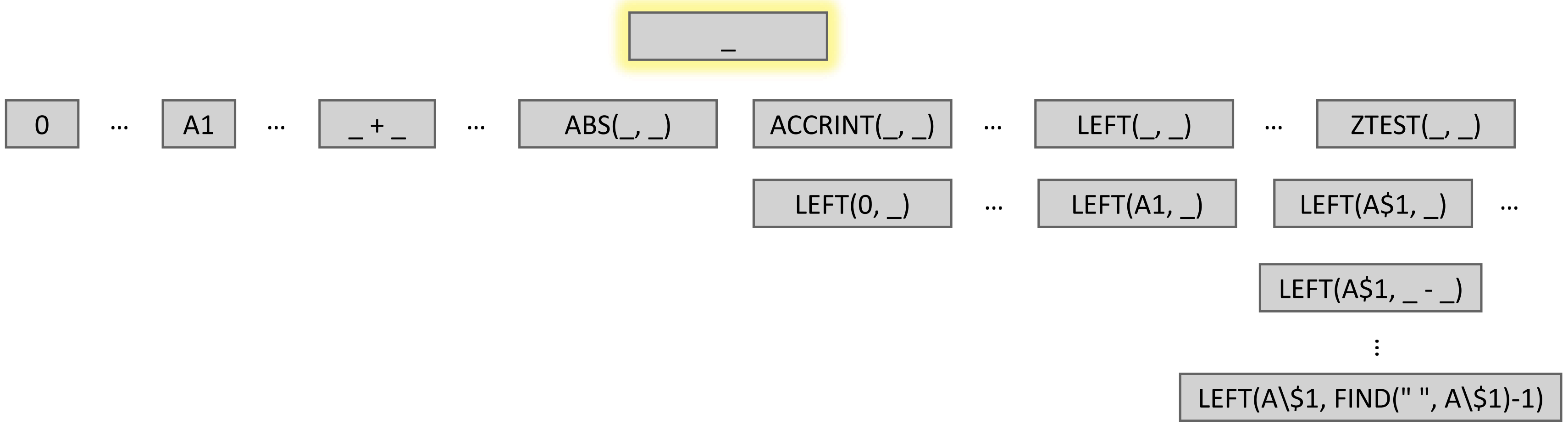
Program Synthesis enumeration method

- Many program synthesis, like FlashFill, uses top-down enumeration
  - Filling in the blanks, starting from initial blank

_

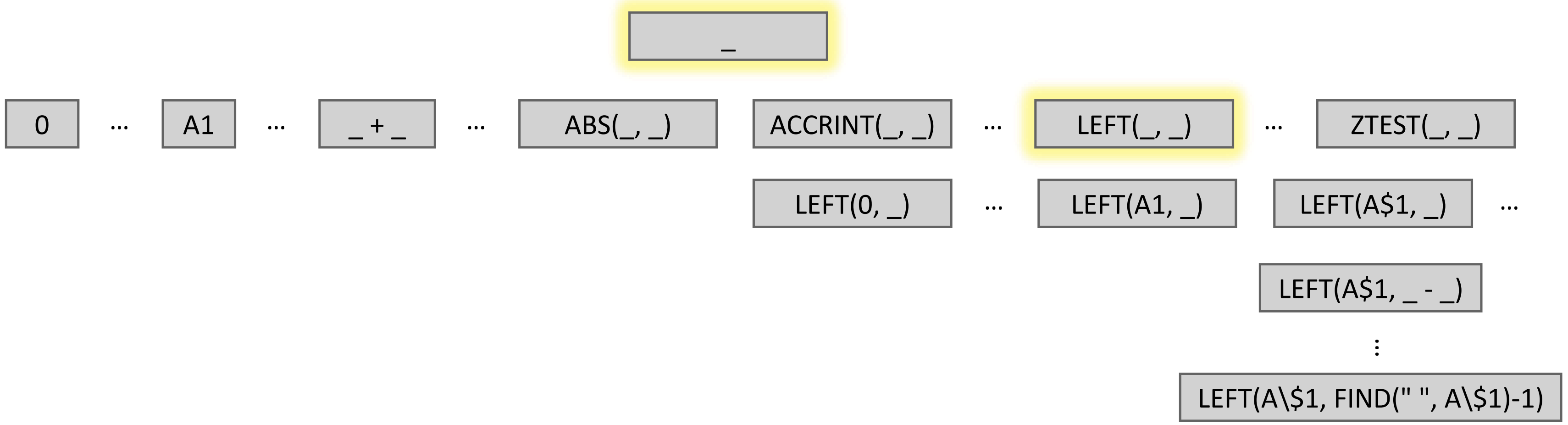
Program Synthesis enumeration method

- Many program synthesis, like FlashFill, uses top-down enumeration
  - Filling in the blanks, starting from initial blank



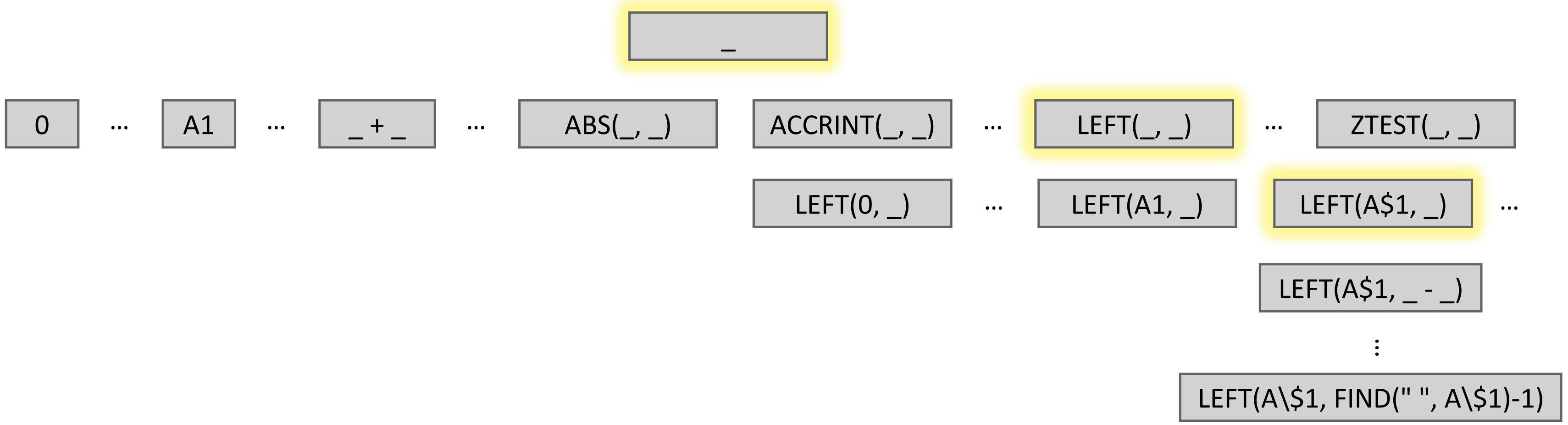
Program Synthesis enumeration method

- Many program synthesis, like FlashFill, uses top-down enumeration
  - Filling in the blanks, starting from initial blank



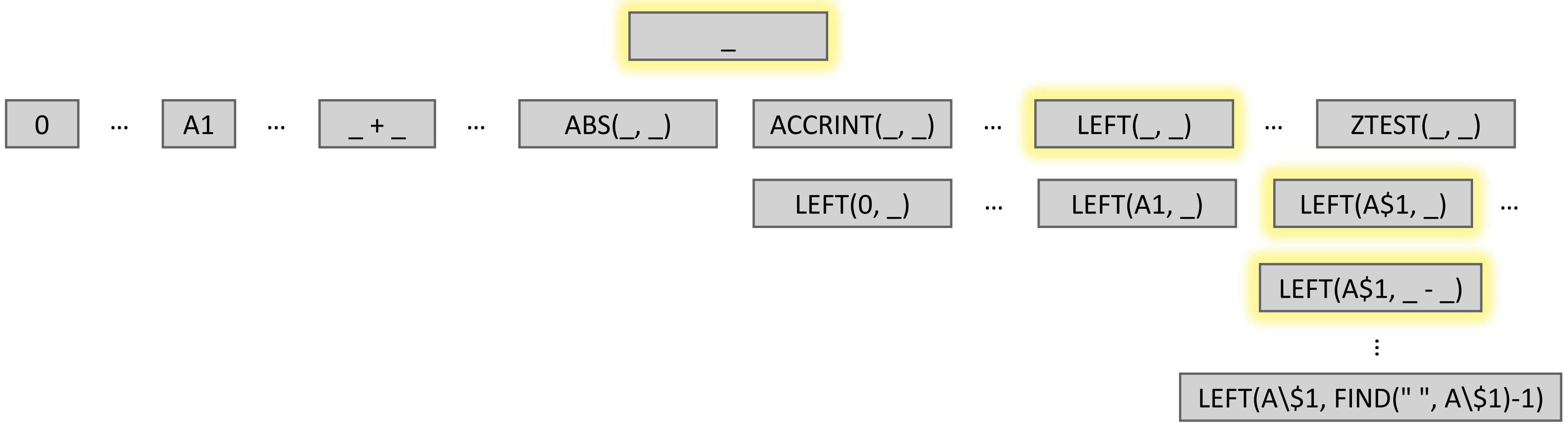
Program Synthesis enumeration method

- Many program synthesis, like FlashFill, uses top-down enumeration
  - Filling in the blanks, starting from initial blank



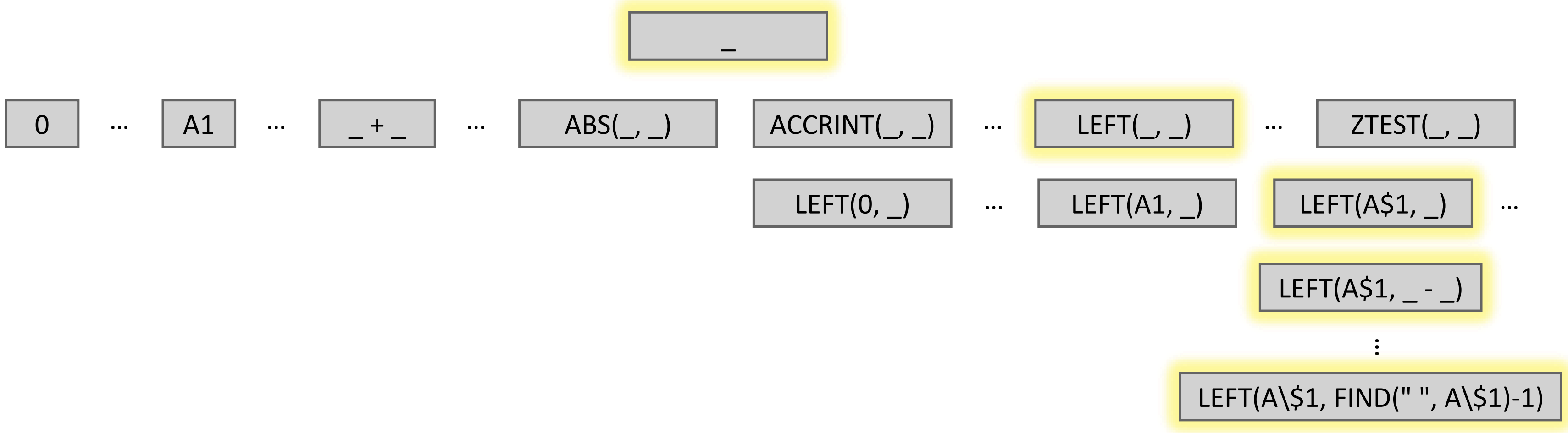
Program Synthesis enumeration method

- Many program synthesis, like FlashFill, uses top-down enumeration
  - Filling in the blanks, starting from initial blank



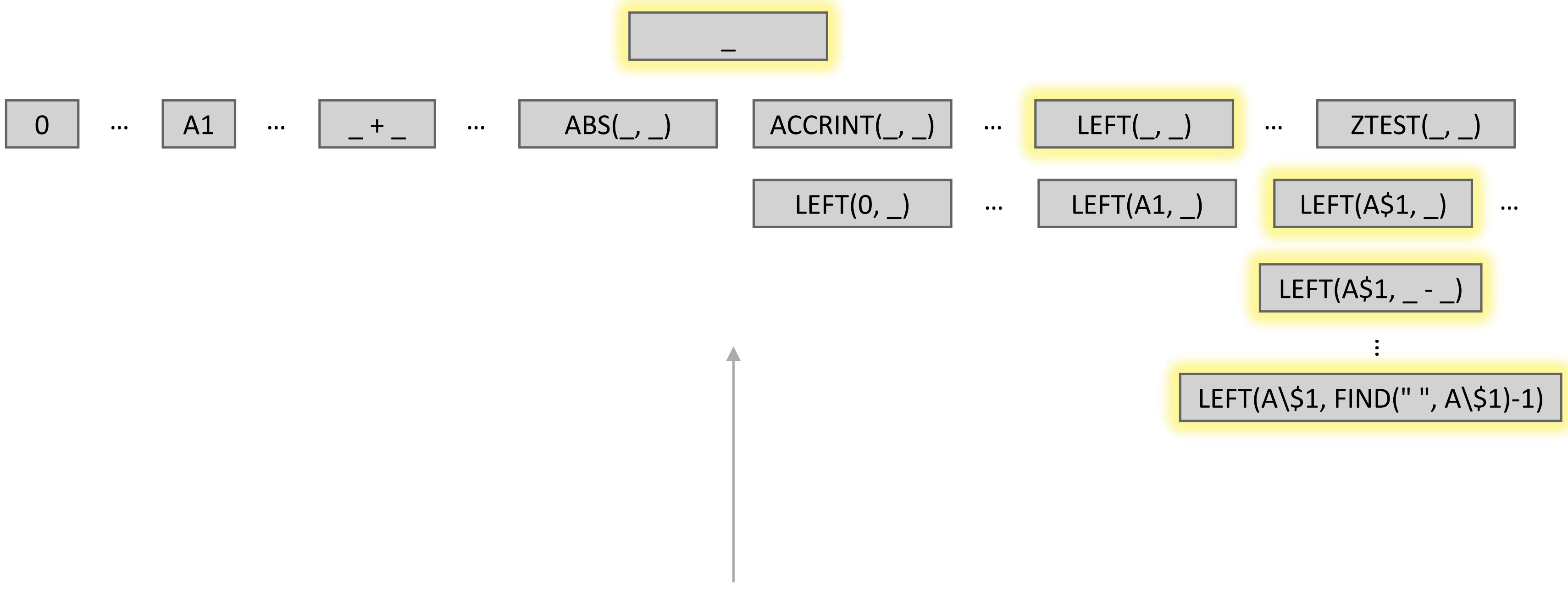
Program Synthesis enumeration method

- Many program synthesis, like FlashFill, uses top-down enumeration
  - Filling in the blanks, starting from initial blank



Program Synthesis enumeration method

- Many program synthesis, like FlashFill, uses top-down enumeration
  - Filling in the blanks, starting from initial blank



How to traverse this whole (infinite) set?

Program Synthesis enumeration method

- Pruning is usually used

  - Traverse the tree in own standard
  - If the node does not fit data, prune out

Data	
x	out
0	1
1	2
2	3
3	4
4	5

_

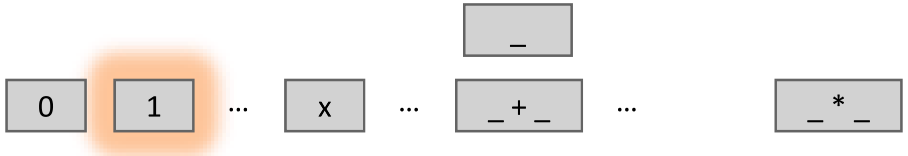
0 1 … x … _ + _ … _ * _

Program Synthesis enumeration method

- Pruning is usually used

- Traverse the tree in own standard
- If the node does not fit data, prune out

Data	
x	out
0	1
1	2
2	3
3	4
4	5



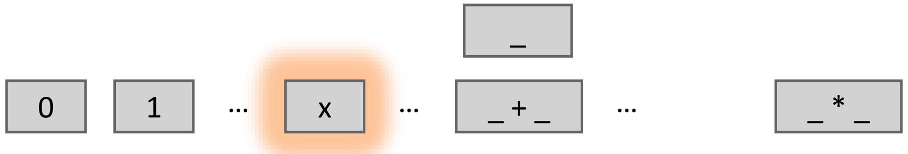
Program Synthesis enumeration method

- Pruning is usually used

x	out
0	1
1	2
2	3
3	4
4	5

Data

  - Traverse the tree in own standard
  - If the node does not fit data, prune out

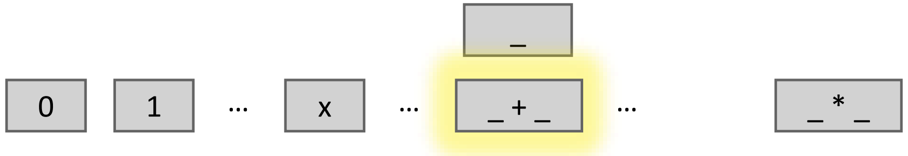


Program Synthesis enumeration method

- Pruning is usually used

  - Traverse the tree in own standard
  - If the node does not fit data, prune out

Data	
x	out
0	1
1	2
2	3
3	4
4	5



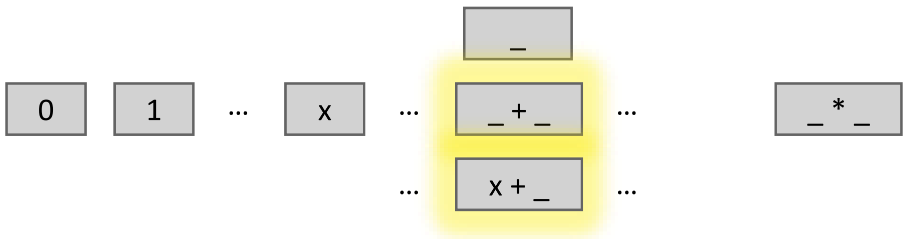
Program Synthesis enumeration method

- Pruning is usually used

  - Traverse the tree in own standard
  - If the node does not fit data, prune out

Data

x	out
0	1
1	2
2	3
3	4
4	5

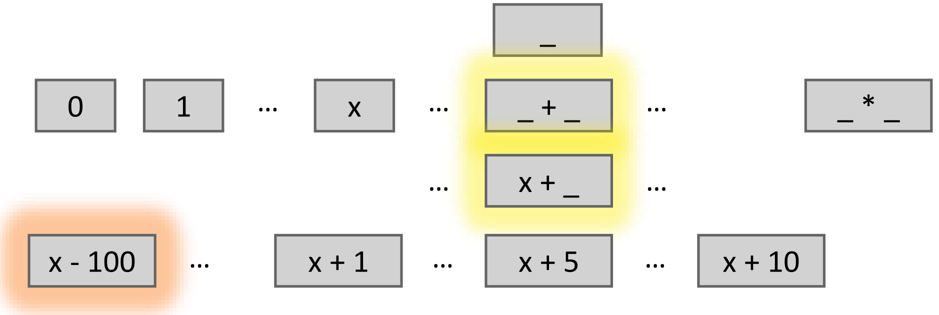


Program Synthesis enumeration method

- Pruning is usually used

  - Traverse the tree in own standard
  - If the node does not fit data, prune out

Data	
x	out
0	1
1	2
2	3
3	4
4	5

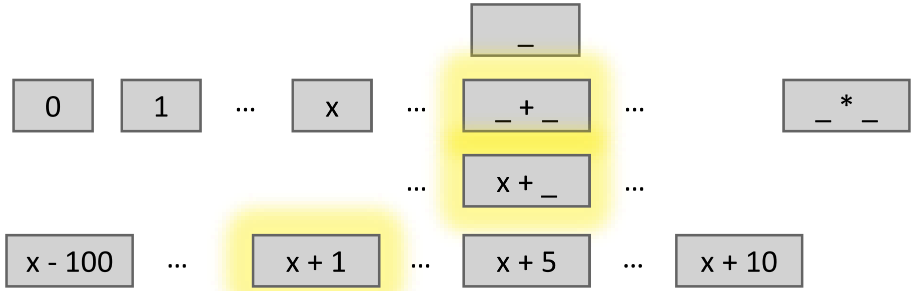


Program Synthesis enumeration method

- Pruning is usually used

- Traverse the tree in own standard
- If the node does not fit data, prune out

Data	
x	out
0	1
1	2
2	3
3	4
4	5

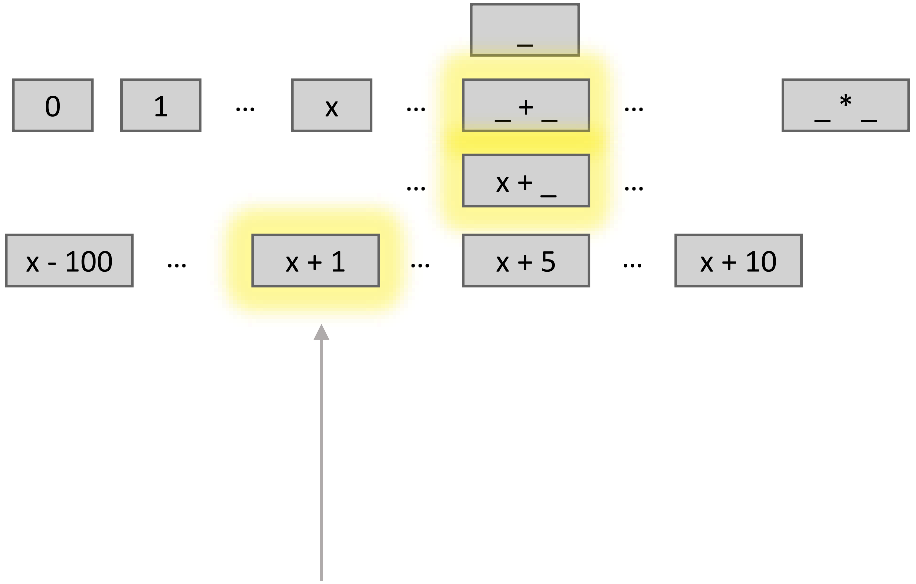


Program Synthesis enumeration method

- Pruning is usually used

  - Traverse the tree in own standard
  - If the node does not fit data, prune out

Data	
x	out
0	1
1	2
2	3
3	4
4	5



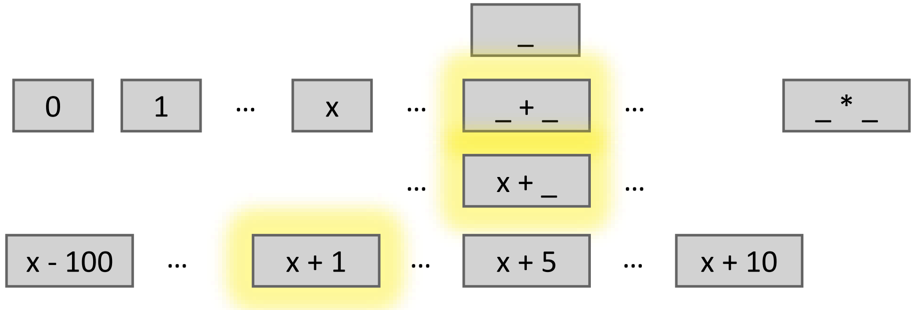
How do we “efficiently” traverse highly likely path first?

Program Synthesis enumeration method

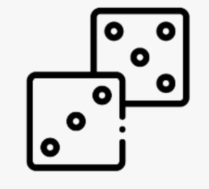
- Pruning is usually used

  - Traverse the tree in own standard
  - If the node does not fit data, prune out

Data	
x	out
0	1
1	2
2	3
3	4
4	5



- For traversing order, probabilistic model is well used



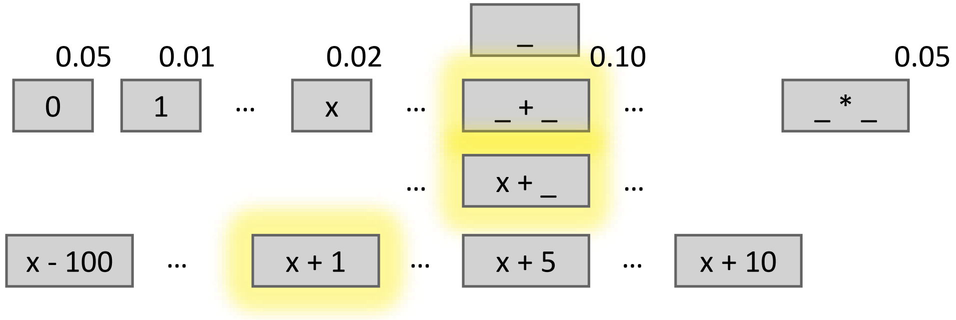
  - Provide probability for each production rule
  - Search first the path with highest likelihood

Program Synthesis enumeration method

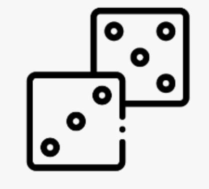
- Pruning is usually used

- Traverse the tree in own standard
- If the node does not fit data, prune out

Data	
x	out
0	1
1	2
2	3
3	4
4	5



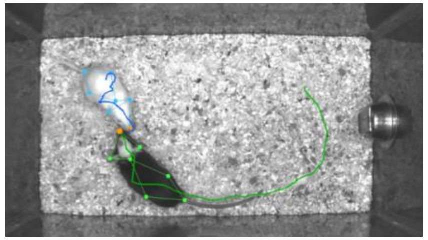
- For traversing order, probabilistic model is well used



- Provide probability for each production rule
- Search first the path with highest likelihood

Problem: Synthesis for fuzzy, new data

1. Biologist want to know mice' behavior



2. Videos mice, extract features (like distance)

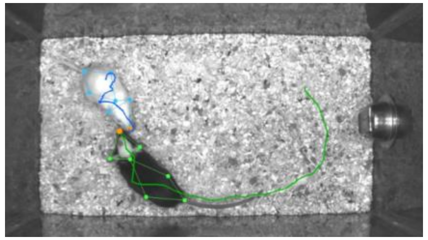
Scene 1			Scene 2		
Time	Dist.	Label	Time	Dist.	Label
0	100	N	0	20	Y
1	80	N	1	13	Y
2	2	Y	2	60	N
3	33	Y	3	94	N
4	60	N	4	100	N

3. Labels some of them for training set

*Jennifer J Sun et al., The Multi-Agent Behavior Dataset: Mouse Dyadic Social Interactions, arXiv preprint, 2021.

Problem: Synthesis for fuzzy, new data

1. Biologist want to know mice' behavior

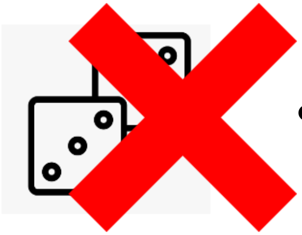


2. Videos mice, extract features (like distance)

Scene 1			Scene 2		
Time	Dist.	Label	Time	Dist.	Label
0	100	N	0	20	Y
1	80	N	1	13	Y
2	2	Y	2	60	N
3	33	Y	3	94	N
4	60	N	4	100	N

3. Labels some of them for training set

- Goal to make formula that **best (not perfectly)** describes the behavior
- Still can prune out formula that is worse than temporal best

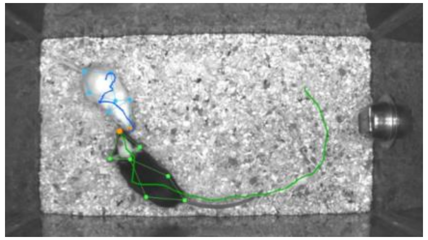


- New data **cannot give any probability** for production rule

*Jennifer J Sun et al., The Multi-Agent Behavior Dataset: Mouse Dyadic Social Interactions, arXiv preprint, 2021.

Problem: Synthesis for fuzzy, new data

1. Biologist want to know mice' behavior

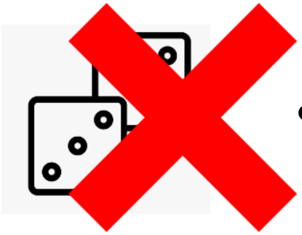


2. Videos mice, extract features (like distance)

Scene 1			Scene 2		
Time	Dist.	Label	Time	Dist.	Label
0	100	N	0	20	Y
1	80	N	1	13	Y
2	2	Y	2	60	N
3	33	Y	3	94	N
4	60	N	4	100	N

3. Labels some of them for training set

- Goal to make formula that **best (not perfectly)** describes the behavior
- Still can prune out formula that is worse than temporal best



- New data **cannot give any probability** for production rule

- Need a **novel approach**

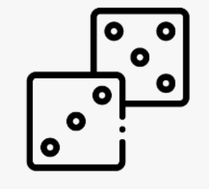
*Jennifer J Sun et al., The Multi-Agent Behavior Dataset: Mouse Dyadic Social Interactions, arXiv preprint, 2021.

Idea

- Pruning by heuristic

  - For each node, calculate highest accuracy it and its child can get
  - If another node has accuracy higher than that, prune this out

- For traversing order,



  - Traverse the tree in order given by A* search
  - Search first the node that is likely to have optimal node as child
  - Makes faster to find optimal node

Killer example

- Pruning by heuristic

  - For each node, calculate highest accuracy it and its child can get
  - If another node has accuracy higher than that, prune this out

Killer example

- Pruning by heuristic

- For each node, calculate highest accuracy it and its child can get
- If another node has accuracy higher than that, prune this out

Data	
x	out
1	10
2	20
3	-10

Killer example

- Pruning by heuristic

  - For each node, calculate highest accuracy it and its child can get
  - If another node has accuracy higher than that, prune this out

Data	
x	out
1	10
2	20
3	-10

_

10 * x

c * x   (c < 0)

Killer example

- Pruning by heuristic

- For each node, calculate highest accuracy it and its child can get
- If another node has accuracy higher than that, prune this out

Data	
x	out
1	10
2	20
3	-10

_

10 * x

accuracy = 0.67

c * x   (c < 0)

accuracy at most 0.33

Killer example

- Pruning by heuristic

  - For each node, calculate highest accuracy it and its child can get
  - If another node has accuracy higher than that, prune this out

Data	
x	out
1	10
2	20
3	-10

_

10 * x

accuracy = 0.67

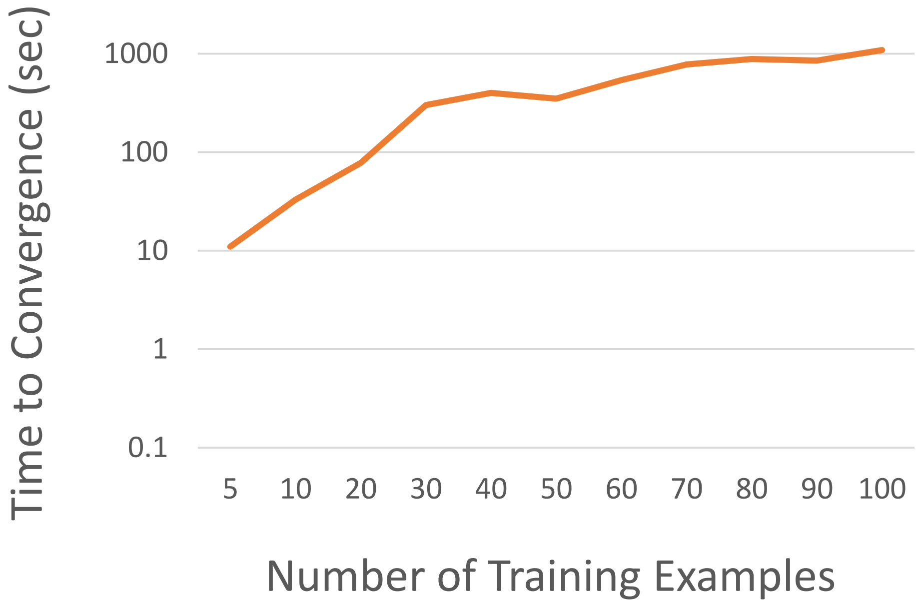
c * x

acc … most 0.33

Appealing result: A new best attempt

Novel approach finds optimal program significantly faster than Metasketches.

Time to identify the optimal program and prove its optimality



Metasketches (SMT solver)

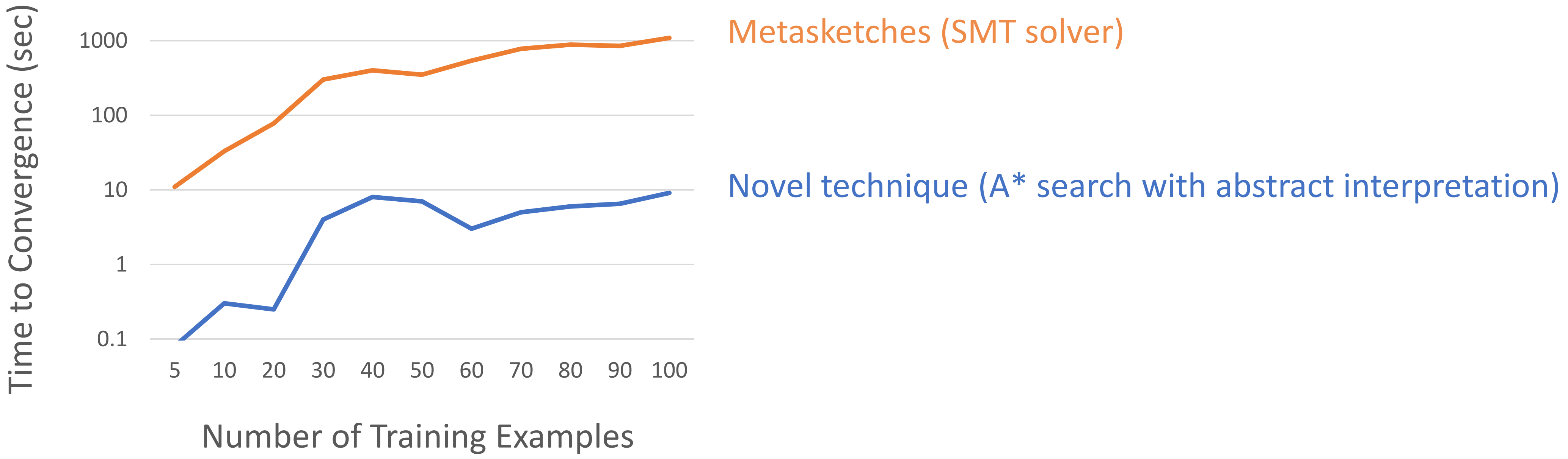
- Adds SMT constraint that programs' score > temporal best
- Change temporal best score if "SAT" made

*Stephen Mell et al., Optimal Program Synthesis via Abstract Interpretation, POPL, 2024.

Appealing result: A new best attempt

Novel approach finds optimal program significantly faster than Metasketches.

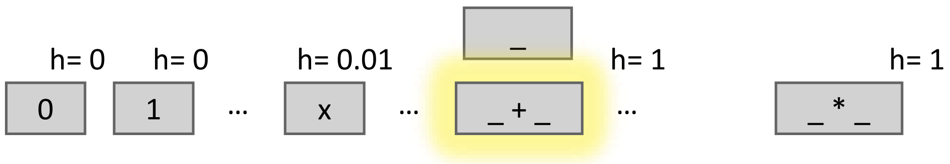
Time to identify the optimal program and prove its optimality



*Stephen Mell et al., Optimal Program Synthesis via Abstract Interpretation, POPL, 2024.

A* search

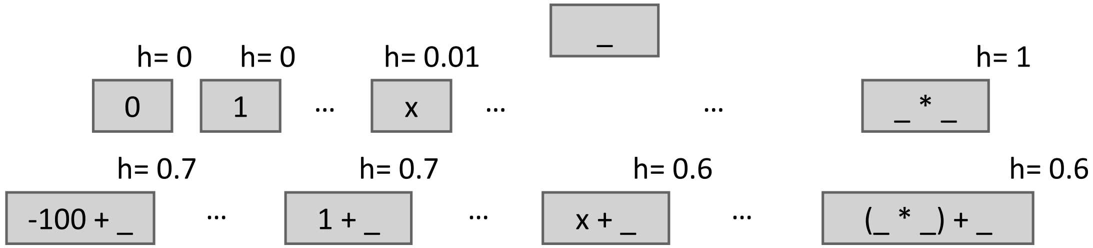
- Search first the path with highest heuristic



- Heuristic is a function that **overapproximates** the accuracy
  - Maximum accuracy **possible** for any program filling [_ + _] ≤ **h**([_ + _])

A* search

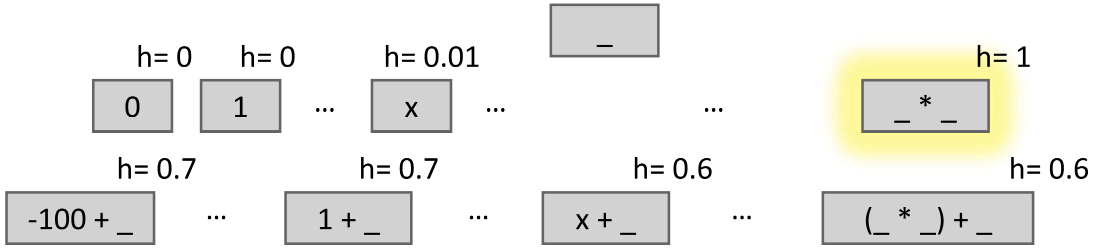
- Search first the path with highest heuristic



- Heuristic is a function that **overapproximates** the accuracy
  - Maximum accuracy **possible** for any program filling _ + _ ≤ **h**(_ + _)

A* search

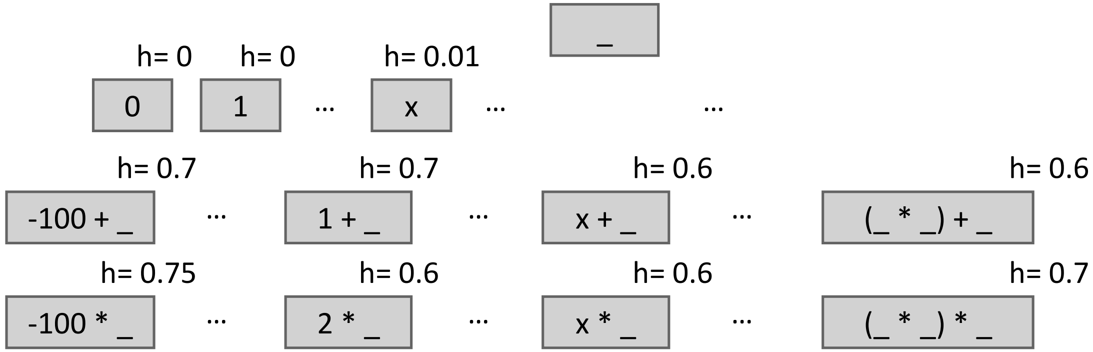
- Search first the path with highest heuristic



- Heuristic is a function that **overapproximates** the accuracy
  - Maximum accuracy **possible** for any program filling _ + _ ≤ **h**(_ + _)

A* search

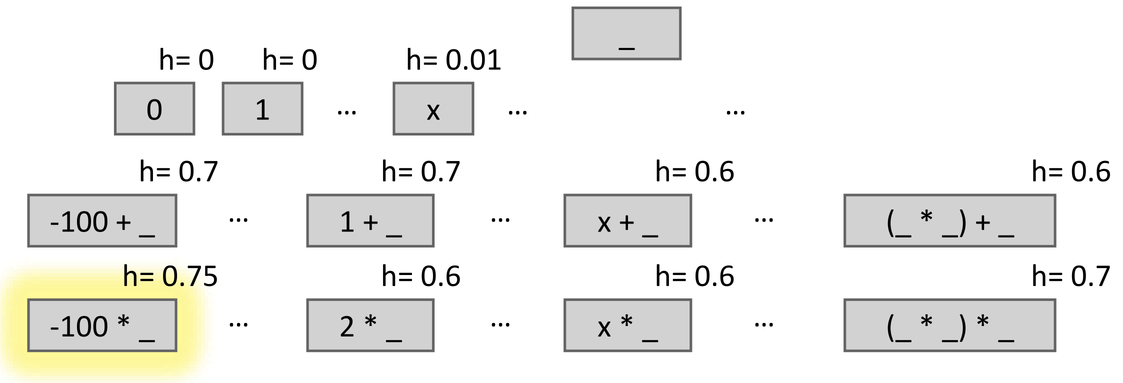
- Search first the path with highest heuristic



- Heuristic is a function that **overapproximates** the accuracy
  - Maximum accuracy **possible** for any program filling _ + _ ≤ **h**(_ + _)

A* search

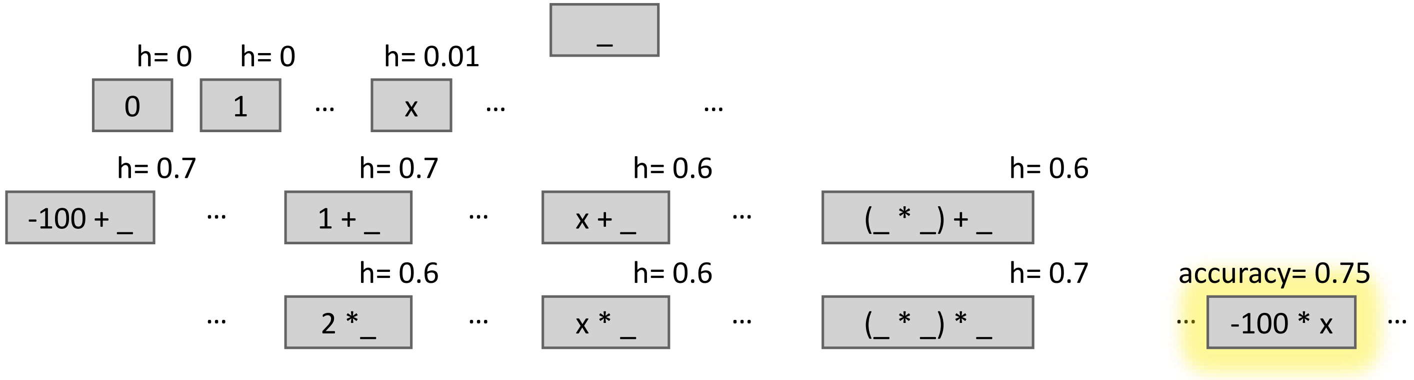
- Search first the path with highest heuristic



- Heuristic is a function that **overapproximates** the accuracy
    - Maximum accuracy **possible** for any program filling _ + _ ≤ **h**(_ + _)

A* search

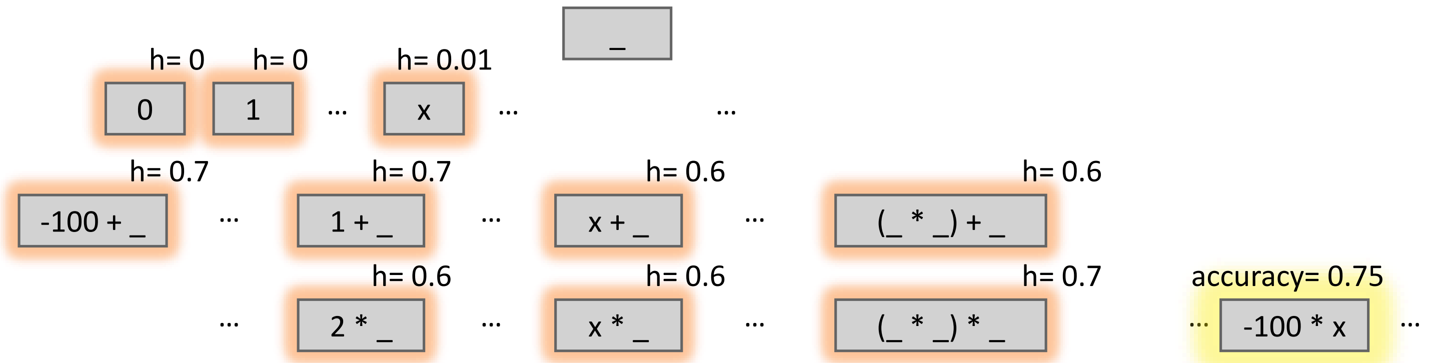
- Search first the path with highest heuristic



- Heuristic is a function that **overapproximates** the accuracy
  - Maximum accuracy **possible** for any program filling [_ + _] ≤ **h**([_ + _])
- When formula is complete, can prune other nodes with low heuristic

A* search

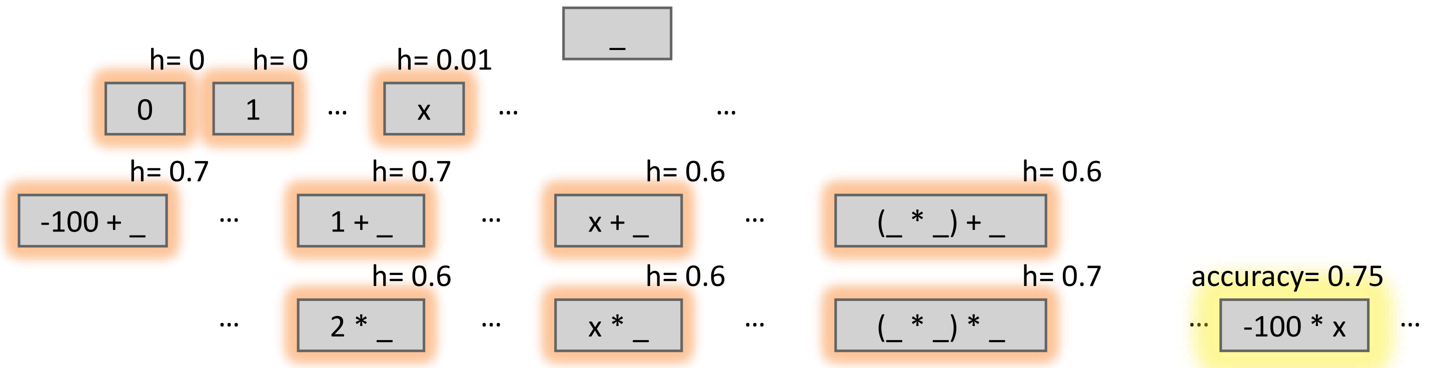
- Search first the path with highest heuristic



- Heuristic is a function that **overapproximates** the accuracy
  - Maximum accuracy **possible** for any program filling [_ + _] ≤ **h**([_ + _])
- When formula is complete, can prune other nodes with low heuristic

A* search

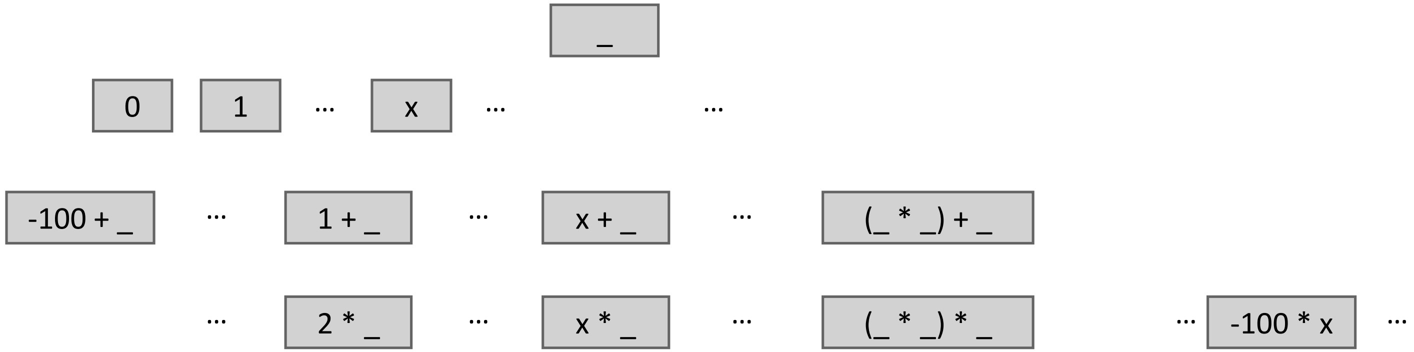
- Search first the path with highest heuristic



- Heuristic is a function that **overapproximates** the accuracy
  - Maximum accuracy **possible** for any program filling _ + _ ≤ **h(** _ + _)
- When formula is complete, can prune other nodes with low heuristic
- If completed formula's accuracy is higher than any other node's heuristic, it's optimal

A* search's heuristic for our approach

- Define heuristic to be the highest accuracy possible for any completion



Data	
x	out
0	1
1	2
2	3
3	4
4	5

A* search's heuristic for our approach

- Define heuristic to be the highest accuracy possible for any completion
  - Example 1: completed node -100 * x

-100 * x

Data	
x	out
0	1
1	2
2	3
3	4
4	5

A* search's heuristic for our approach

- Define heuristic to be the highest accuracy possible for any completion
  - Example 1: completed node -100 * x

-100 * x

Data	
x	out
0	1
1	2
2	3
3	4
4	5

Evaluation	
x	out
0	0
1	-100
2	-200
3	-300
4	-400

A* search's heuristic for our approach

- Define heuristic to be the highest accuracy possible for any completion
  - Example 1: completed node -100 * x
  - h(-100 * x) = 0

-100 * x

Data	
x	out
0	1
1	2
2	3
3	4
4	5

Evaluation	
x	out
0	0
1	-100
2	-200
3	-300
4	-400

A* search's heuristic for our approach

- Define heuristic to be the highest accuracy possible for any completion
  - Example 1: completed node -100 * x
  - h(-100 * x) = 0
- Problem : need to evaluate for a lot number of constants
  - -100, -99, -98, …, 100

-100 * x

Data	
x	out
0	1
1	2
2	3
3	4
4	5

Evaluation	
x	out
0	0
1	-100
2	-200
3	-300
4	-400

A* search's heuristic for our approach

- Define heuristic to be the highest accuracy possible for any completion
  - Example 1: completed node -100 * x
  - h(-100 * x) = 0
- Problem : need to evaluate for a lot number of constants
  - -100, -99, -98, …, 100
- Solution : use abstract interpretation

-100 * x

Data	
x	out
0	1
1	2
2	3
3	4
4	5

Evaluation	
x	out
0	0
1	-100
2	-200
3	-300
4	-400

A* search's heuristic for our approach

- Define heuristic to be the highest accuracy possible for any completion
  - Example 1: completed node -100 * x
  - h(-100 * x) = 0
- Problem : need to evaluate for a lot number of constants
  - -100, -99, -98, …, 100
- Solution : use abstract interpretation
  - c * x (c is constant)

c * x

Data	
x	out
0	1
1	2
2	3
3	4
4	5

A* search's heuristic for our approach

- Define heuristic to be the highest accuracy possible for any completion
  - Example 1: completed node -100 * x
  - h(-100 * x) = 0
- Problem : need to evaluate for a lot number of constants
  - -100, -99, -98, …, 100
- Solution : use abstract interpretation
  - c * x (c is constant)
  - Split the interval for c

c * x

Data	
x	out
0	1
1	2
2	3
3	4
4	5

A* search's heuristic for our approach

- Define heuristic to be the highest accuracy possible for any completion
  - Example 1: completed node -100 * x
  - h(-100 * x) = 0
- Problem : need to evaluate for a lot number of constants
  - -100, -99, -98, …, 100
- Solution : use abstract interpretation
  - c * x (c is constant)
  - Split the interval for c
  - Example 2: incomplete node $c_{[0,\ 100]}$ * x

$c_{[0,\ 100]}$ * x

Data	
x	out
0	1
1	2
2	3
3	4
4	5

Evaluation	
x	out
0	0
1	[0, 100]
2	[0, 200]
3	[0, 300]
4	[0, 400]

A* search's heuristic for our approach

- Define heuristic to be the highest accuracy possible for any completion
  - Example 1: completed node -100 * x
  - h(-100 * x) = 0
- Problem : need to evaluate for a lot number of constants
  - -100, -99, -98, …, 100
- Solution : use abstract interpretation
  - c * x (c is constant)
  - Split the interval for c
  - Example 2: incomplete node $c_{[0, 100]}$ * x
  - h($c_{[0, 100]}$ * x) = max([0, 0.8]) = 0.8

$c_{[0, 100]}$ * x

Data	
x	out
0	1
1	2
2	3
3	4
4	5

Evaluation	
x	out
0	0
1	[0, 100]
2	[0, 200]
3	[0, 300]
4	[0, 400]

A* search's heuristic for our approach

- Define heuristic to be the highest accuracy possible for any completion
  - Example 1: completed node -100 * x
  - h(-100 * x) = 0
- Problem : need to evaluate for a lot number of constants
  - -100, -99, -98, …, 100
- Solution : use abstract interpretation
  - c * x (c is constant)
  - Split the interval for c
  - Example 2: incomplete node $c_{[0,\ 100]}$ * x
  - h($c_{[0,\ 100]}$ * x) = max([0, 0.8]) = 0.8

$c_{[0,\ 100]}$ * x

Data	
x	out
0	1
1	2
2	3
3	4
4	5

Evaluation	
x	out
0	0
1	[0, 100]
2	[0, 200]
3	[0, 300]
4	[0, 400]

A* search's heuristic for our approach

- Define heuristic to be the highest accuracy possible for any completion
  - Example 1: completed node -100 * x
  - h(-100 * x) = 0
- Problem : need to evaluate for a lot number of constants
  - -100, -99, -98, …, 100
- Solution : use abstract interpretation
  - c * x (c is constant)
  - Split the interval for c
  - Example 2: incomplete node $c_{[0, 100]}$ * x
  - h($c_{[0, 100]}$ * x) = max([0, 0.8]) = 0.8
  - Example 3: incomplete note $c_{[-100, 0]}$ * x

$c_{[-100, 0]}$ * x

Data	
x	out
0	1
1	2
2	3
3	4
4	5

Evaluation	
x	out
0	0
1	[-100, 0]
2	[-200, 0]
3	[-300, 0]
4	[-400, 0]

A* search's heuristic for our approach

- Define heuristic to be the highest accuracy possible for any completion
  - Example 1: completed node -100 * x
  - h(-100 * x) = 0
- Problem : need to evaluate for a lot number of constants
  - -100, -99, -98, …, 100
- Solution : use abstract interpretation
  - c * x (c is constant)
  - Split the interval for c
  - Example 2: incomplete node $c_{[0, 100]}$ * x
  - h($c_{[0, 100]}$ * x) = max([0, 0.8]) = 0.8
  - Example 3: incomplete note $c_{[-100, 0]}$ * x
  - h($c_{[-100, 0]}$ * x) = max([0, 0]) = 0

$c_{[-100, 0]}$ * x

Data	
x	out
0	1
1	2
2	3
3	4
4	5

Evaluation	
x	out
0	0
1	[-100, 0]
2	[-200, 0]
3	[-300, 0]
4	[-400, 0]

A* search's heuristic for our approach

- Define heuristic to be the highest accuracy possible for any completion
  - Example 1: completed node -100 * x
  - h(-100 * x) = 0
- Problem : need to evaluate for a lot number of constants
  - -100, -99, -98, …, 100
- Solution : use abstract interpretation
  - c * x (c is constant)
  - Split the interval for c
  - Example 2: incomplete node $c_{[0, 100]}$ * x
  - h($c_{[0, 100]}$ * x) = max([0, 0.8]) = 0.8
  - Example 3: incomplete note $c_{[-100, 0]}$ * x
  - h($c_{[-100, 0]}$ * x) = max([0, 0]) = 0
  - When heuristic 0.8 is the remaining highest, split [0, 100] to [0, 50], [50, 100] and repeat

$c_{[-100, 0]}$ * x

Data	
x	out
0	1
1	2
2	3
3	4
4	5

Evaluation	
x	out
0	0
1	[-100, 0]
2	[-200, 0]
3	[-300, 0]
4	[-400, 0]

A* search's heuristic for our approach

- Define heuristic to be the highest accuracy possible for any completion
  - Example 4: incomplete node

x * _

Data	
x	out
0	1
1	2
2	3
3	4
4	5

A* search's heuristic for our approach

- Define heuristic to be the highest accuracy possible for any completion
  - Example 4: incomplete node
  - Annotate blank with (-∞, ∞)

x * _

Data	
x	out
0	1
1	2
2	3
3	4
4	5

A* search's heuristic for our approach

- Define heuristic to be the highest accuracy possible for any completion
  - Example 4: incomplete node
  - Annotate blank with (-∞, ∞)
  - h(x * _) = max([0, 0.8]) = 0.8

x * _

Data	
x	out
0	1
1	2
2	3
3	4
4	5

Evaluation	
x	out
0	0
1	(-∞, ∞)
2	(-∞, ∞)
3	(-∞, ∞)
4	(-∞, ∞)

A* search's heuristic for our approach

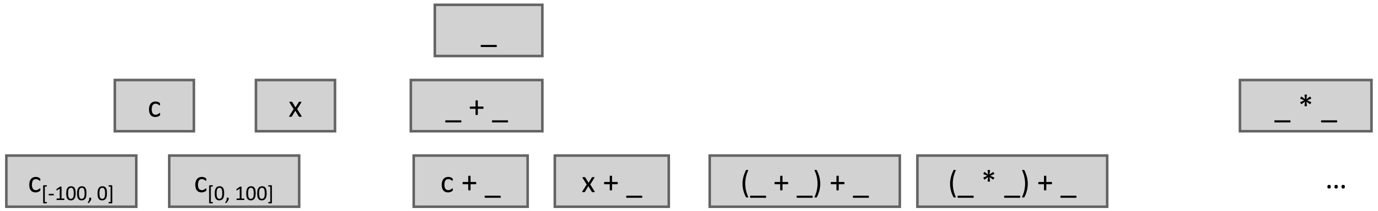
- In conclusion, the space for synthesis looks like below

A* search's heuristic for our approach

- In conclusion, the space for synthesis looks like below

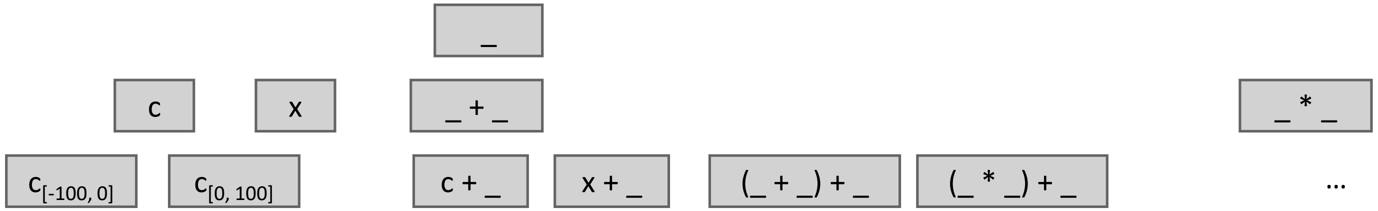
A* search's heuristic for our approach

- In conclusion, the space for synthesis looks like below



A* search's heuristic for our approach

- In conclusion, the space for synthesis looks like below



- Each node is evaluated, and the one with highest heuristic will get expanded

Domain for experiment

- Test on domain specific languages (which means, language with grammar)
- Each domain specific language is for describing specific situation

NEAR DSL (for trajectory-per-time $\mathbf{x} \in (\mathbb{R}^n)^*$)

(returns true if the value is nonnegative)

Quivr DSL (for trajectory-per-time $\mathbf{x} \in (\mathbb{R}^n)^*$)

(classify whole data to one true / false)

Domain for experiment

- Test on domain specific languages (which means, language with grammar)
- Each domain specific language is for describing specific situation

NEAR DSL (for trajectory-per-time $\mathbf{x} \in (R^n)^*$)

(returns true if the value is nonnegative)

map(z_i - 50)

z	out
(0, 100)	(f, t)

Quivr DSL (for trajectory-per-time $\mathbf{x} \in (R^n)^*$)

(classify whole data to one true / false)

Domain for experiment

- Test on domain specific languages (which means, language with grammar)
- Each domain specific language is for describing specific situation

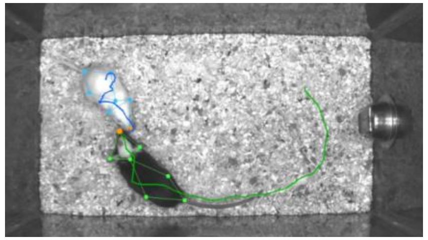
NEAR DSL (for trajectory-per-time $\mathbf{x}\in(R^n)^*$)

(returns true if the value is nonnegative)

map(z_i - 50)

z	out
(0, 100)	(f, t)

(useful for labeling animal behavior per time)



Scene 1			Scene 2		
Time	Dist.	Label	Time	Dist.	Label
0	100	N	0	20	Y
1	80	N	1	13	Y
2	2	Y	2	60	N
3	33	Y	3	94	N
4	60	N	4	100	N

Quivr DSL (for trajectory-per-time $\mathbf{x}\in(R^n)^*$)

(classify whole data to one true / false)

Domain for experiment

- Test on domain specific languages (which means, language with grammar)
- Each domain specific language is for describing specific situation

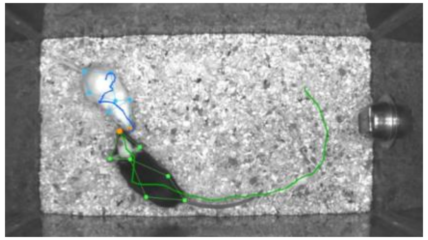
NEAR DSL (for trajectory-per-time $\mathbf{x} \in (R^n)^*$)

(returns true if the value is nonnegative)

map(z_i - 50)

z	out
(0, 100)	(f, t)

(useful for labeling animal behavior per time)



Scene 1			Scene 2		
Time	Dist.	Label	Time	Dist.	Label
0	100	N	0	20	Y
1	80	N	1	13	Y
2	2	Y	2	60	N
3	33	Y	3	94	N
4	60	N	4	100	N

Quivr DSL (for trajectory-per-time $\mathbf{x} \in (R^n)^*$)

(classify whole data to one true / false)

(fun x → x_2 > 0) ∧ (fun x → x_1 + x_2 ≥ 100)

z	out
(0, 100)	t

Domain for experiment

- Test on domain specific languages (which means, language with grammar)
- Each domain specific language is for describing specific situation

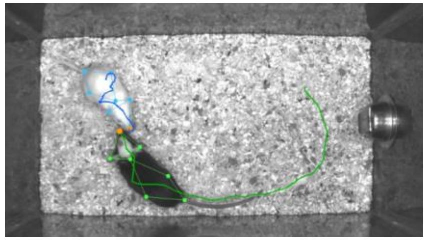
NEAR DSL (for trajectory-per-time $\mathbf{x} \in (R^n)^*$)

(returns true if the value is nonnegative)

map(z_i - 50)

z	out
(0, 100)	(f, t)

(useful for labeling animal behavior per time)



Scene 1			Scene 2		
Time	Dist.	Label	Time	Dist.	Label
0	100	N	0	20	Y
1	80	N	1	13	Y
2	2	Y	2	60	N
3	33	Y	3	94	N
4	60	N	4	100	N

Quivr DSL (for trajectory-per-time $\mathbf{x} \in (R^n)^*$)

(classify whole data to one true / false)

(fun x → x_2 > 0) ∧ (fun x → $x_1 + x_2 \geq 100$)

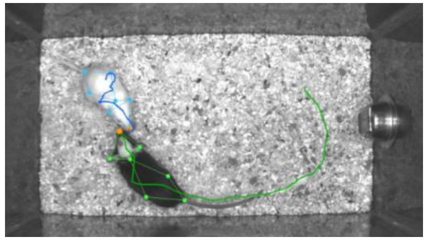
z	out
(0, 100)	t

(useful for clarifying if object satisfied certain condition)

Evaluation

- Neurosymbolic program synthesis benchmarks
  - features (numbers) are extracted from video by neural networks

NEAR DSL (for trajectory-per-time $\mathbf{x} \in (R^n)^*$)



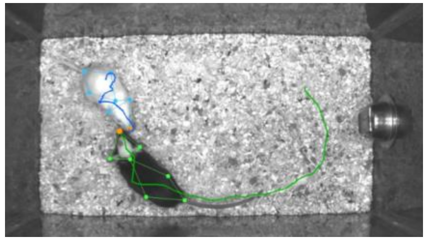
Quivr DSL (for trajectory-per-time $\mathbf{x} \in (R^n)^*$)

Evaluation

- Neurosymbolic program synthesis benchmarks
  - features (numbers) are extracted from video by neural networks

NEAR DSL (for trajectory-per-time $\mathbf{x} \in (R^n)^*$)

CRIM13 dataset
Goal: clarify when mice sniff each other



Quivr DSL (for trajectory-per-time $\mathbf{x} \in (R^n)^*$)

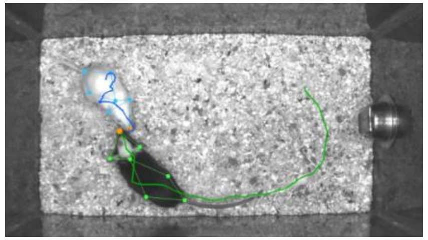
MABe22 dataset
Goal: clarify if three mice interact

YTStreams dataset
Goal: clarify if car caught in traffic camera makes a right turn

Evaluation

- Neurosymbolic program synthesis benchmarks
  - features (numbers) are extracted from video by neural networks
- Comparison: Metasketches (SMT solver-based synthesis tool), Breadth-First Search

NEAR DSL (for trajectory-per-time $\mathbf{x} \in (\mathrm{R}^n)^*$)	Quivr DSL (for trajectory-per-time $\mathbf{x} \in (\mathrm{R}^n)^*$)
CRIM13 dataset Goal: clarify when mice sniff each other	MABe22 dataset Goal: clarify if three mice interact YTStreams dataset Goal: clarify if car caught in traffic camera makes a right turn



Evaluation

- Neurosymbolic program synthesis benchmarks
  - features (numbers) are extracted from video by neural networks
- Comparison: Metasketches (SMT solver-based synthesis tool), Breadth-First Search
- Q1. How fast does approach find optimal solution, compared to Metasketches?
- Q2. In constrained time, how well does approach's answer evaluate, compared to BFS?

Result

Q1. How fast does approach find optimal solution, compared to Metasketches?

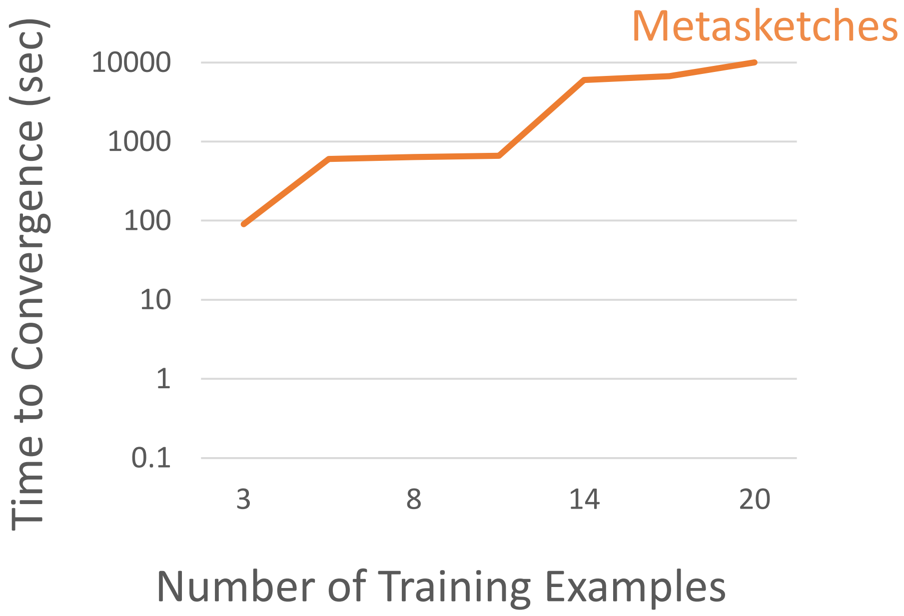
A1. Novel approach finds significantly faster than Metasketches.

Result

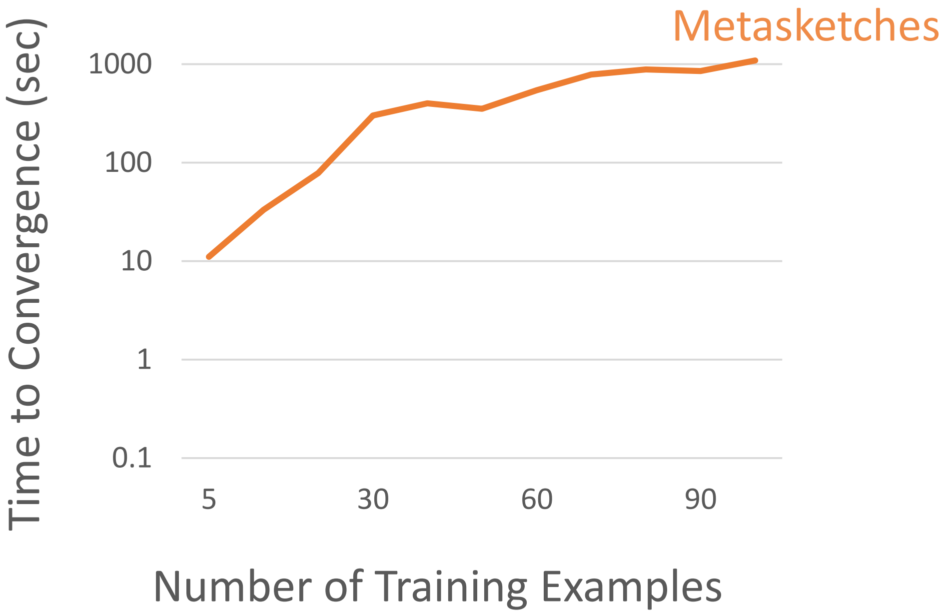
Q1. How fast does approach find optimal solution, compared to Metasketches?
A1. Novel approach finds significantly faster than Metasketches.

Time to find optimal program and prove its optimality

Domain: NEAR DSL, Dataset: CRIM13-A



Domain: Quivr DSL, Dataset: YTStreams-G

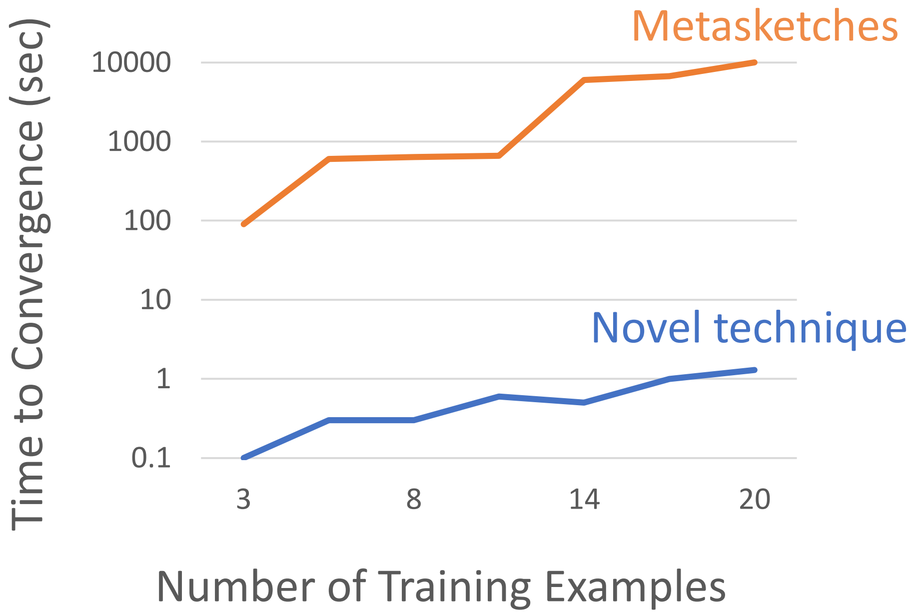


Result

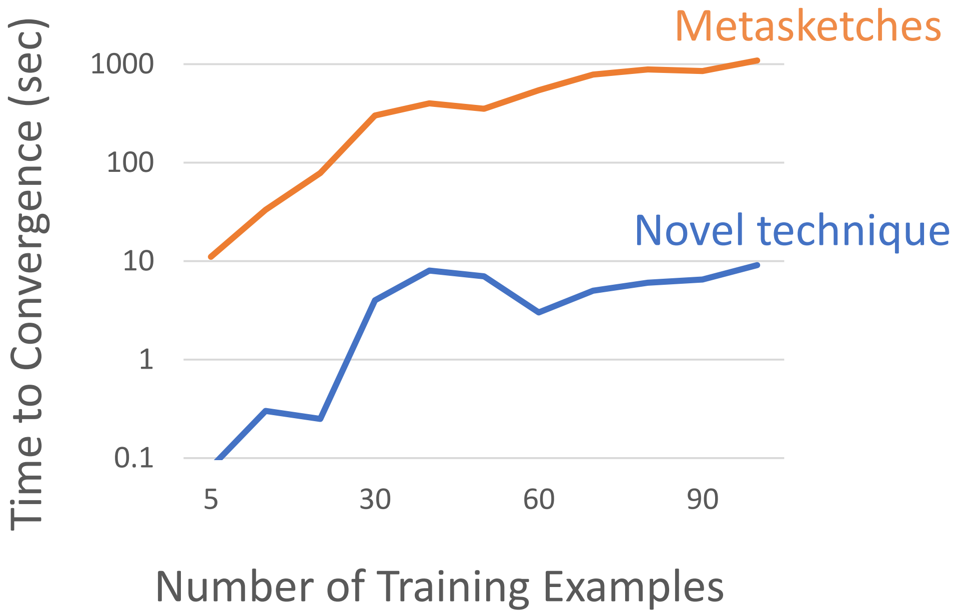
Q1. How fast does approach find optimal solution, compared to Metasketches?
A1. Novel approach finds significantly faster than Metasketches.

Time to find optimal program and prove its optimality

Domain: NEAR DSL, Dataset: CRIM13-A



Domain: Quivr DSL, Dataset: YTStreams-G



Result

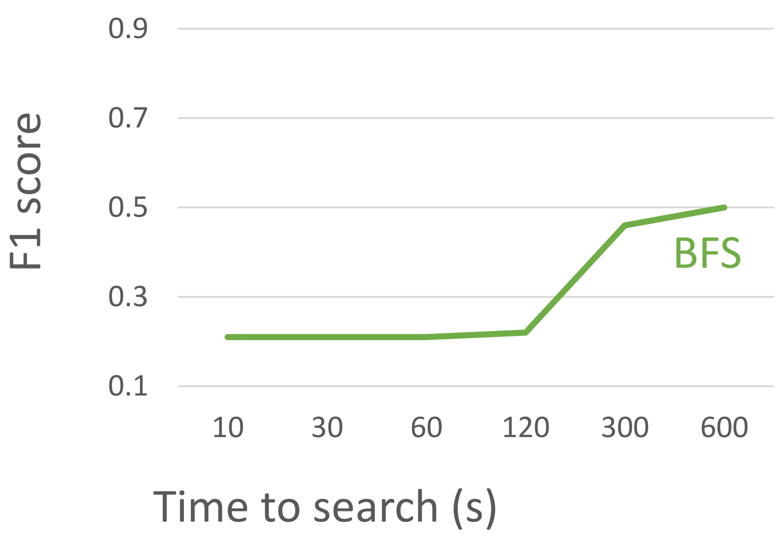
Q2. In constrained time, how well does approach's answer evaluate, compared to BFS?
A2. On most tasks, novel approach achieves higher F1 scores more quickly than BFS.

Result

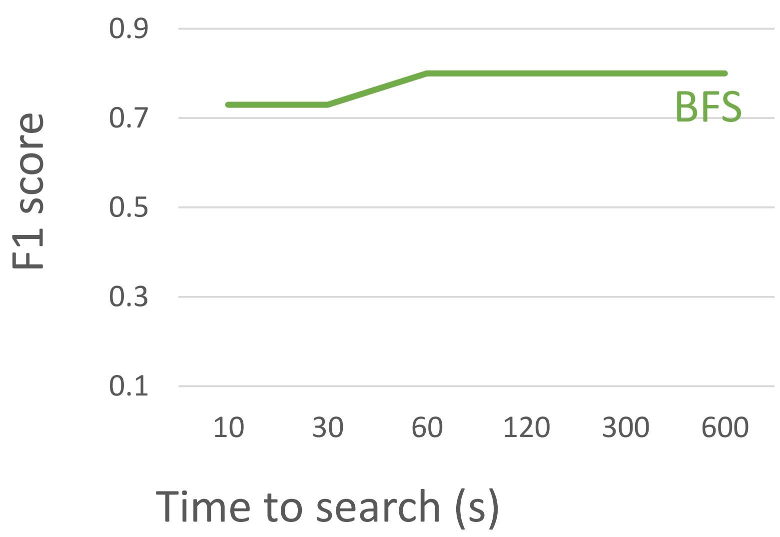
Q2. In constrained time, how well does approach's answer evaluate, compared to BFS?
A2. On most tasks, novel approach achieves higher F1 scores more quickly than BFS.

Best F1 score found

Domain: NEAR DSL, Dataset: CRIM13-A



Domain: Quivr DSL, Dataset: YTStreams-G

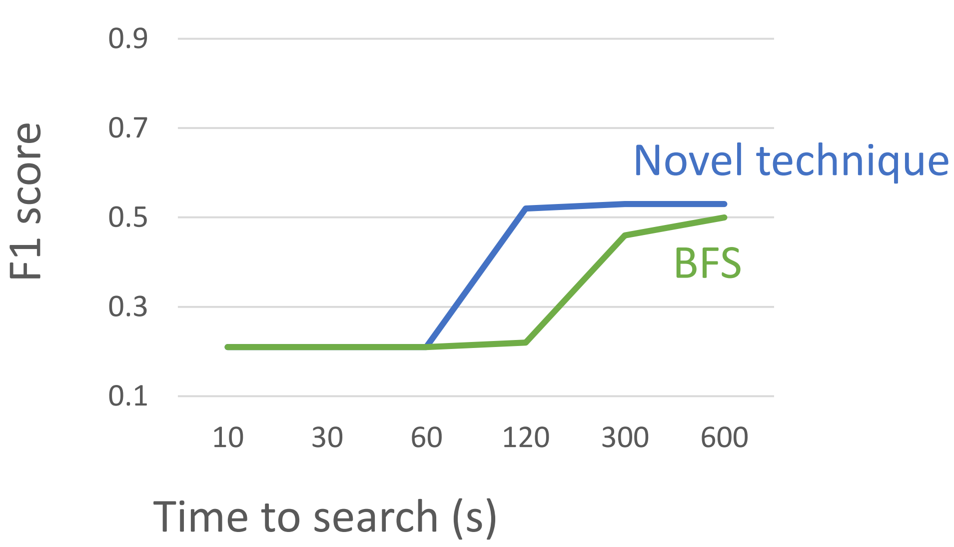


Result

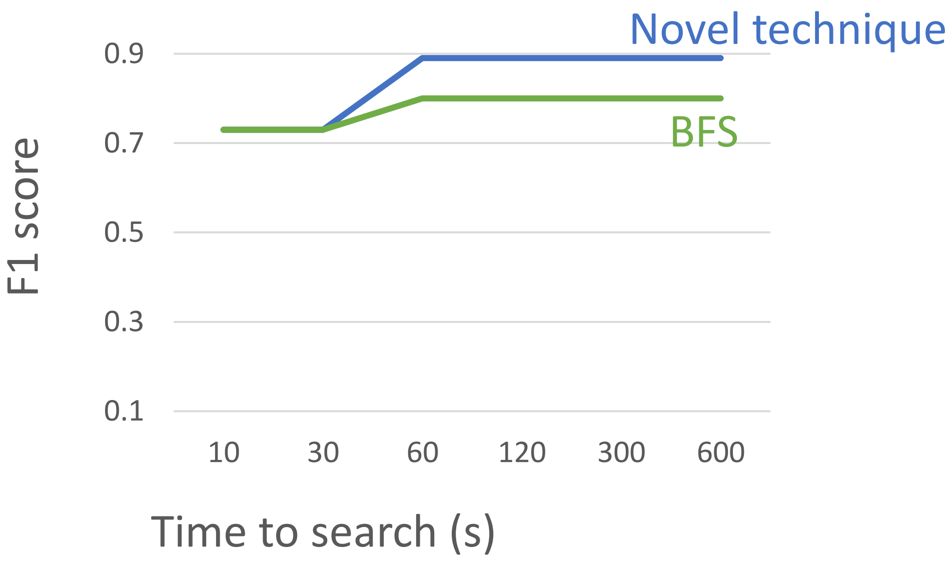
Q2. In constrained time, how well does approach's answer evaluate, compared to BFS?
A2. On most tasks, novel approach achieves higher F1 scores more quickly than BFS.

Best F1 score found

Domain: NEAR DSL, Dataset: CRIM13-A



Domain: Quivr DSL, Dataset: YTStreams-G



Conclusion

- Synthesizing program is a powerful, interpretable tool for classifying behaviors
- But for fuzzy and new data, traditional techniques are not available
- The novel approach utilizes A* search with abstract interpretation for scalability
- Experimental evaluation demonstrates superior results from other approaches