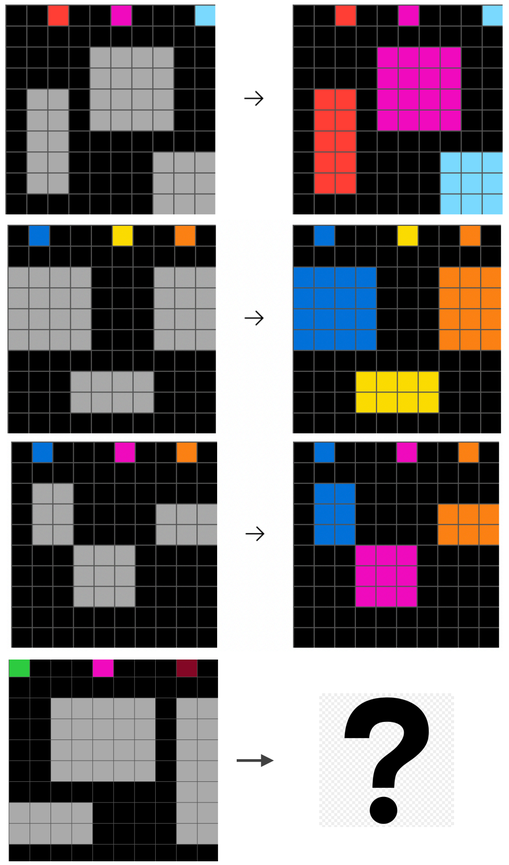
HySynth: Context-free LLM Approximation for Guiding Program Synthesis

NeurIPS 2024

UC San Diego

Shraddha Barke, Emmanuel Anaya-Gonzalez, Saketh Kasibatla,
Taylor Berg-Kirkpatrick, Nadia Polikarpova

Programming By Example



ARC Domain

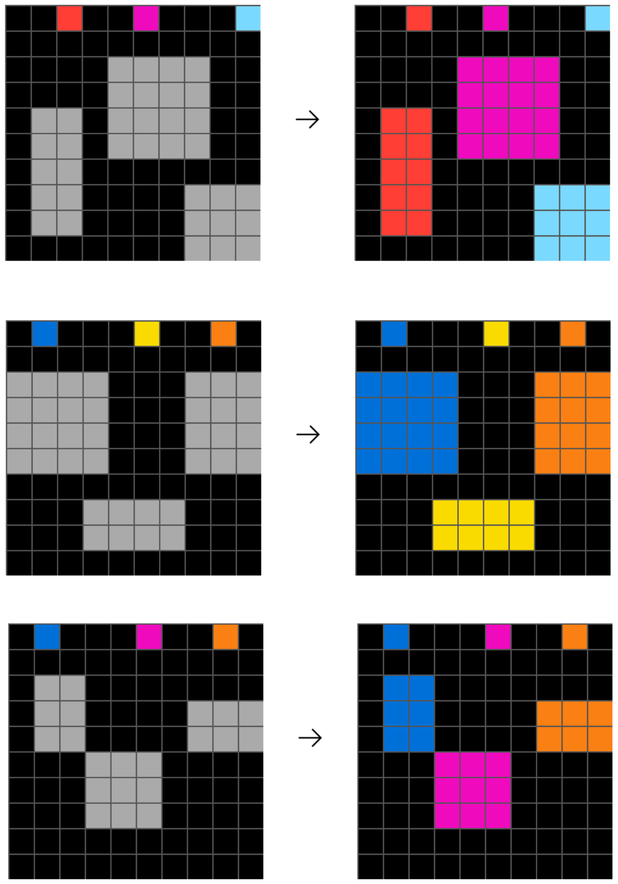
```
Input Tensor:
in = [[0, 1, 0, 0],
      [0, 1, 1, 0],
      [1, 1, 1, 1]]

Output Tensor:
out = [[0.0, 1.0, 0.0, 0.0],
       [0.0, 0.5, 0.5, 0.0],
       [0.25, 0.25, 0.25, 0.25]]

Natural language description
(optional):

"Normalize the rows of a tensor"
```

Tensor Domain

	A	B
1	Name and ID	First name and last name
2	Thomas, Rhonda 82132	Rhonda Thomas
3	Emmett, Keara 34231	Keara Emmett
4	Vogel, James 32493	James Vogel
5	Jelen, Bill 23911	Bill Jelen
6	Miller, Sylvia 78356	Sylvia Miller
7	Lambert, Bobby 25900	Bobby Lambert

String Domain

ARC Example



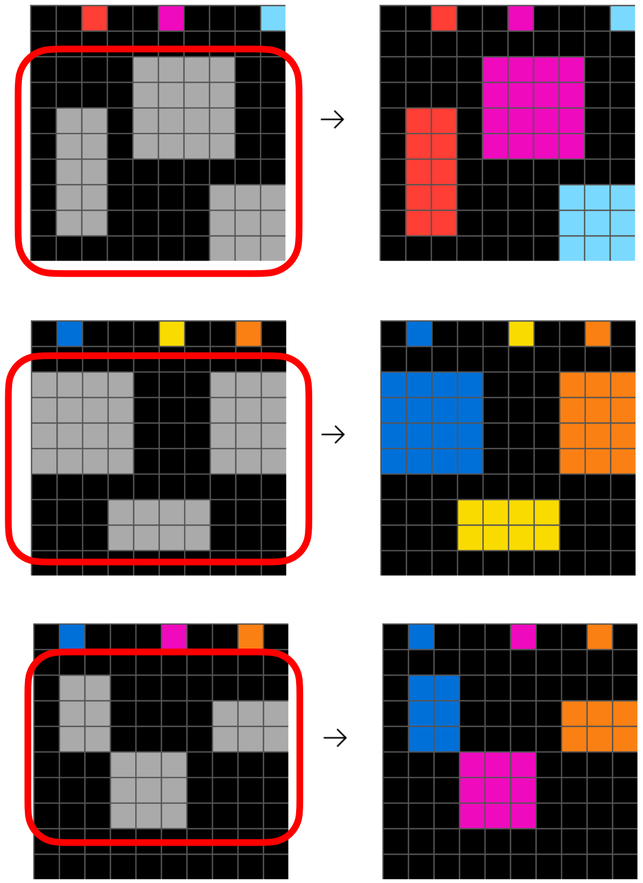
Filter

if color_of(self) = GREY ∧ is_neighbor(self, other) ∧ size_of(other) = MIN
then update_color(color_of(other))

Transformation

[Khalil 2023] Graphs, Constraints, and Search for the Abstraction and Reasoning Corpus

ARC Example



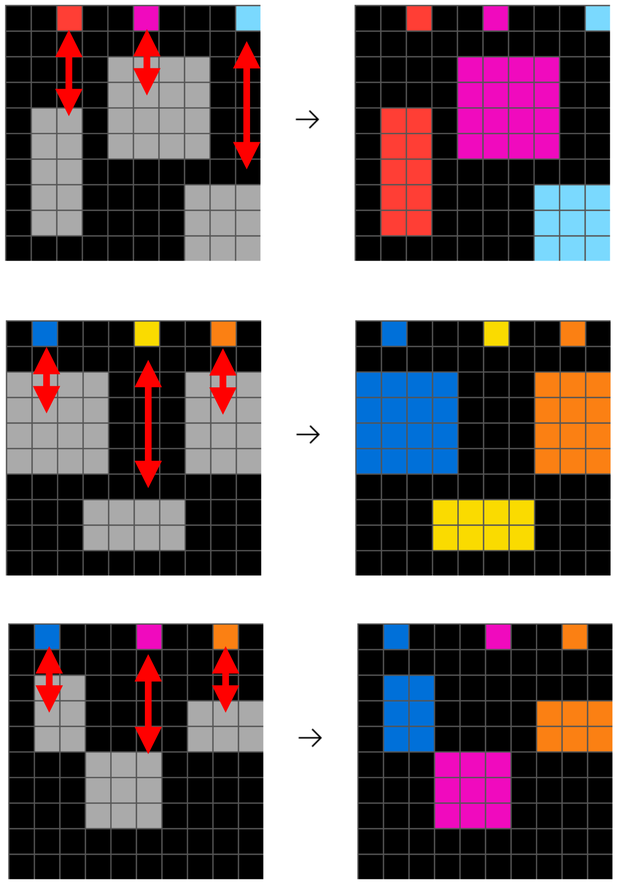
Filter

if color_of(self) = GREY ∧ is_neighbor(self, other) ∧ size_of(other) = MIN
then update_color(color_of(other))

Transformation

[Khalil 2023] Graphs, Constraints, and Search for the Abstraction and Reasoning Corpus

ARC Example



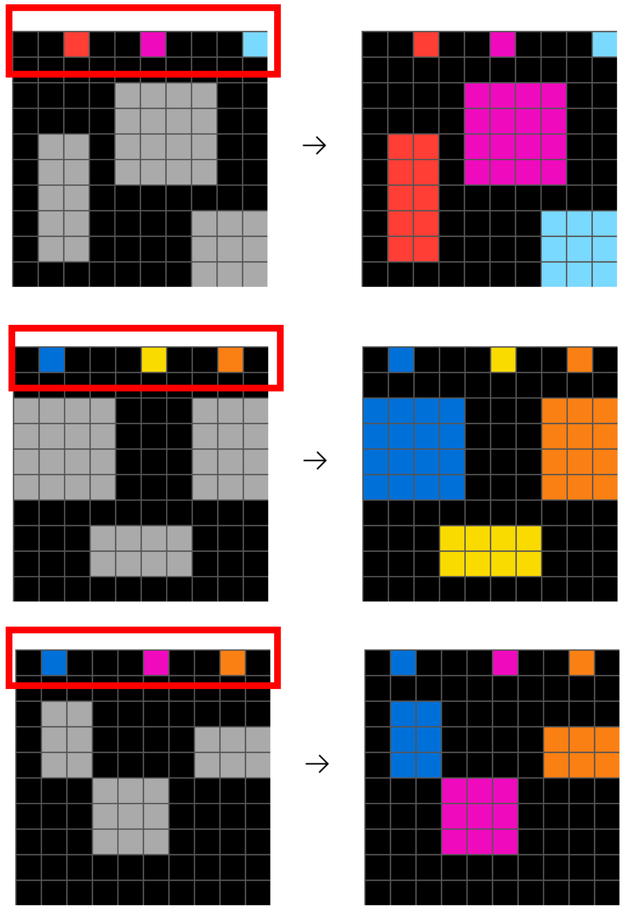
Filter

if color_of(self) = GREY ∧ is_neighbor(self, other) ∧ size_of(other) = MIN
then update_color(color_of(other))

Transformation

[Khalil 2023] Graphs, Constraints, and Search for the Abstraction and Reasoning Corpus

ARC Example



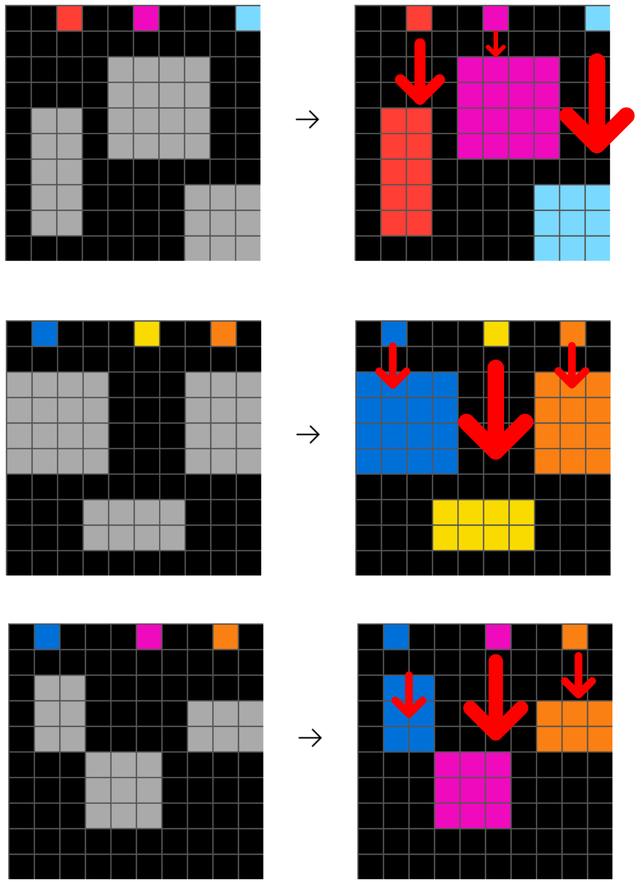
Filter

if color_of(self) = GREY ∧ is_neighbor(self, other) ∧ size_of(other) = MIN
then update_color(color_of(other))

Transformation

[Khalil 2023] Graphs, Constraints, and Search for the Abstraction and Reasoning Corpus

ARC Example



Filter

if color_of(self) = GREY ∧ is_neighbor(self, other) ∧ size_of(other) = MIN
then update_color(color_of(other))

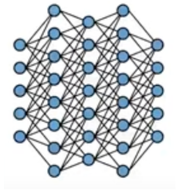
Transformation

[Khalil 2023] Graphs, Constraints, and Search for the Abstraction and Reasoning Corpus

LLM Performance on PBE

LLM and domain	% of valid completions	Problems Solved
ARC - GPT4o	78.4%	3 / 160
String - GPT4o	37.5%	14 / 70
Tensor - GPT4o	99.9%	0 / 69

Hybrid Synthesis



Neural Models

+

Program Search λ

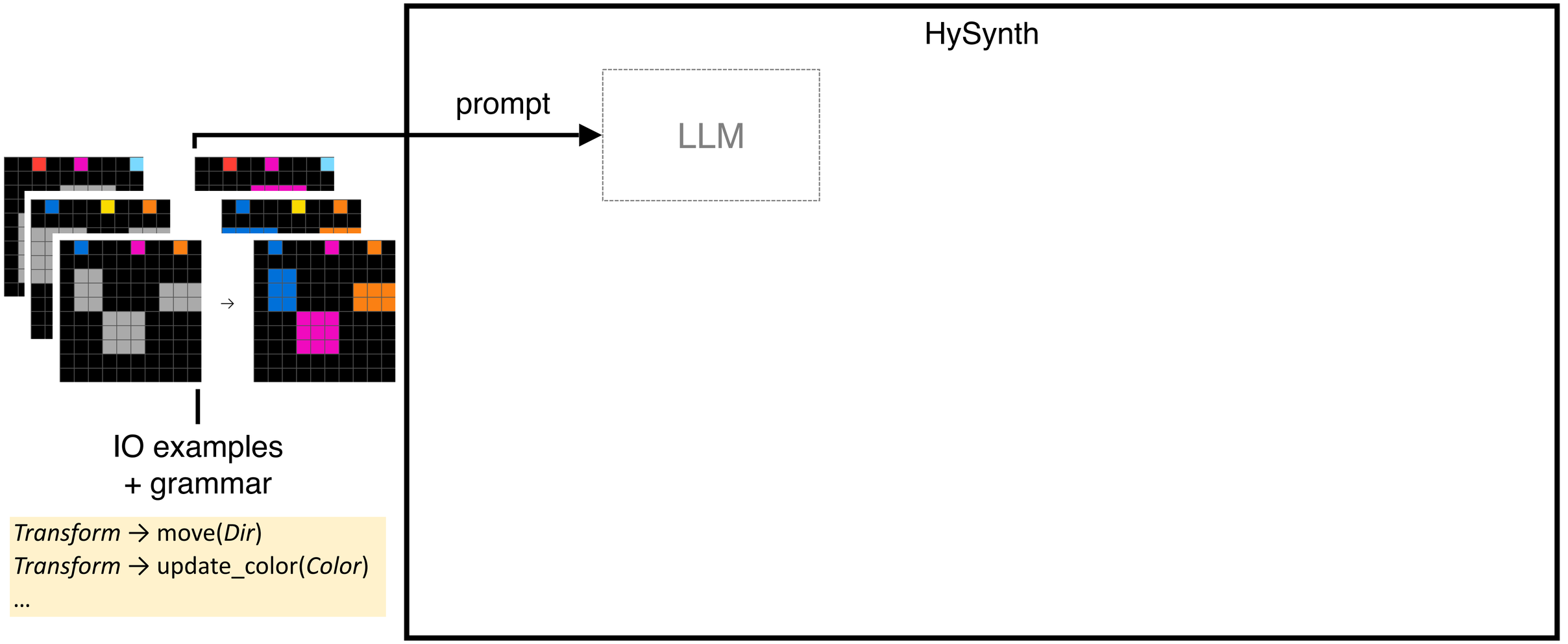
- Learning efficient: gradient descent
- Data Inefficient: dense sampling of data
- Good at intuition and pattern recognition

System 1 thinking

- Learning inefficient: combinatorial search
- Data efficient: few examples to generalize
- Good at planning and reasoning tasks

System 2 thinking

Our Solution: Hybrid Synthesis



Prompting the LLM

Problem Context

+

DSL Grammar

+

In-Context Examples +
Test task

Prompting the LLM (ARC)

Problem Context

+

DSL Grammar

+

In-Context Examples + Test task

You are an efficient assistant for logical reasoning and code generation. You will help me solve a visual perception and reasoning task. I will first provide you with the definition of a Domain Specific Language you will use for writing a solution for the task. I will then present you with the description of the task that you will be tested in. You will then respond the queries I make regarding the solution of the task.

This is the definition of the DSL you will use to solve the task. It is given as a context-free grammar in the EBNF format used by the Lark parser generator, with some informative comments about the semantics. You will return a string that is parseable by the `program` non-terminal of the grammar.

```
 library: "(" program* ")"
// Rules are executed one after another, in the order they appear.
// There could be no rules, in which case the program does nothing.
program: "(" "do" rule* ")"
...
```

Now we continue with the visual perception and reasoning task. The input for the task is a small number of pairs of grids. The value of each of the cells in the grids correspond to colors defined in the DSL. Each pair of grids correspond to an input-output example for an unknown program P.
For each pair, the program P is evaluated on the image grid and operates on the objects that appear in it. The output of the program is then the output image. .. Now I will show you some demonstration tasks along with the output you would be expected to produce for each of them.

Prompting the LLM (ARC)

Problem Context

+

DSL Grammar

+

In-Context Examples + Test task

```
            // select all objects with the given color.
filter_op: "(" FL_COLOR color_expr ")"
            // select all objects that contain the given number of pixels.
          | "(" FL_SIZE int_expr ")"
            // select the objects of min/max size
          | "(" FL_SIZE ORD_STAT ")"
            // select all objects that can see the given number of neighbors.
          | "(" FL_DEGREE int_expr ")"
        ...
// transformations are applied to the objects in the subset sequentially.
transform_list: "(" ")"
              | "(" transform_plus_ ")"
transform_plus_: transform
               | transform transform_plus_
// a single transformation is applied to the objects in the subset simultaneously.
// so we can think of the transformation as applying to a single object.
            // change the color the object to the given color.
transform: "(" TR_UPDATE_COLOR color_expr ")"
            // move the object in the given direction.
          | "(" TR_MOVE_NODE DIRECTION ")"
            // extend (move while leaving a trail) the object in the given direction.
          | "(" TR_EXTEND_NODE DIRECTION tr_extend_node_params*")"
            // move the object in the given direction until it hits another object.
        ...
        | "(" TR_MIRROR mirror_axis ")"
         | "(" TR_FLIP SYMM_AXIS ")"
……
```

Prompting the LLM (ARC)

Problem Context

+

DSL Grammar

+

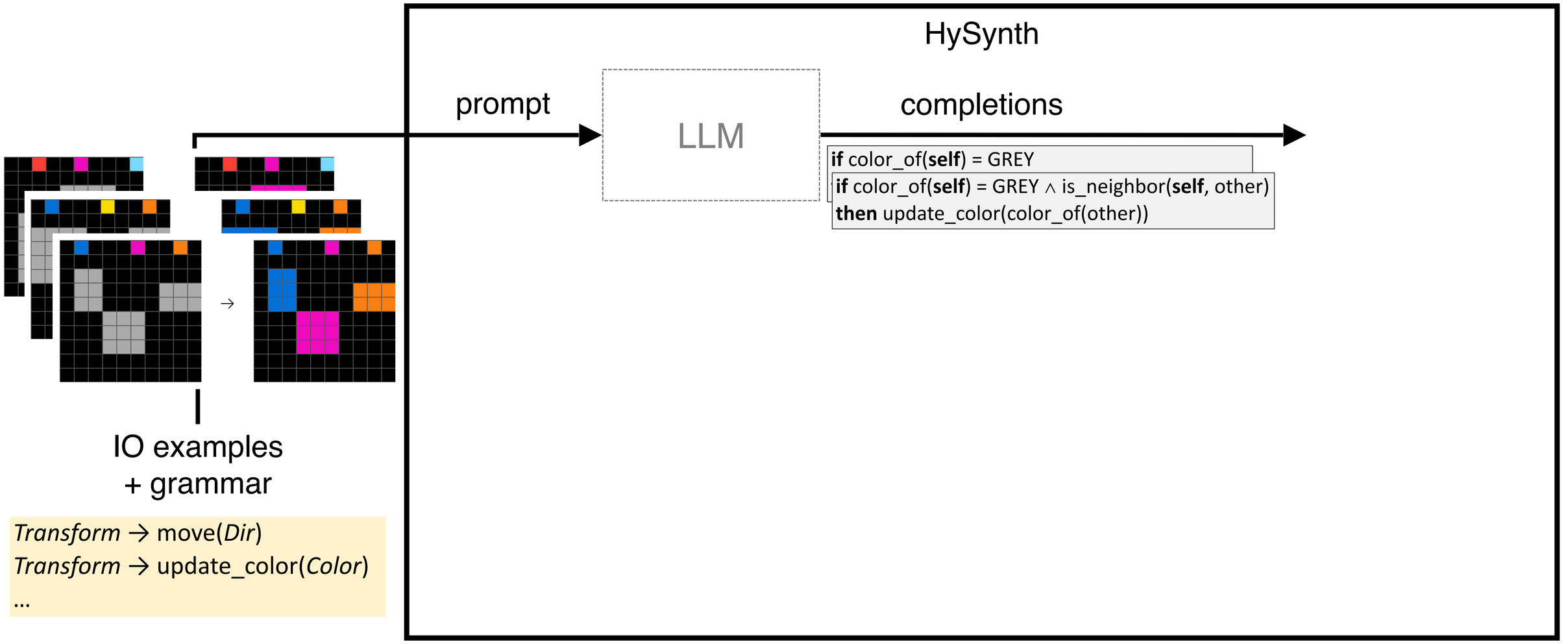
In-Context Examples + Test task

```
## DEMONSTRATION TASK 1
### INPUT
PAIR 1
INPUT GRID:
O O O O O O O O
O O O O O R O O
OUTPUT GRID:
O O O O O O O O
O O O O O Y O O

### EXPECTED OUTPUT
{
    "nl_description": "Recolor all objects to color Y",
    "code": <SOLUTION>
}
```

Now follows task you will be evaluated on. Output the solution as a JSON object, which should contain both a natural language description of the solution and the solution written in the DSL. The code should be parseable by the DSL grammar. The JSON must have the following structure:

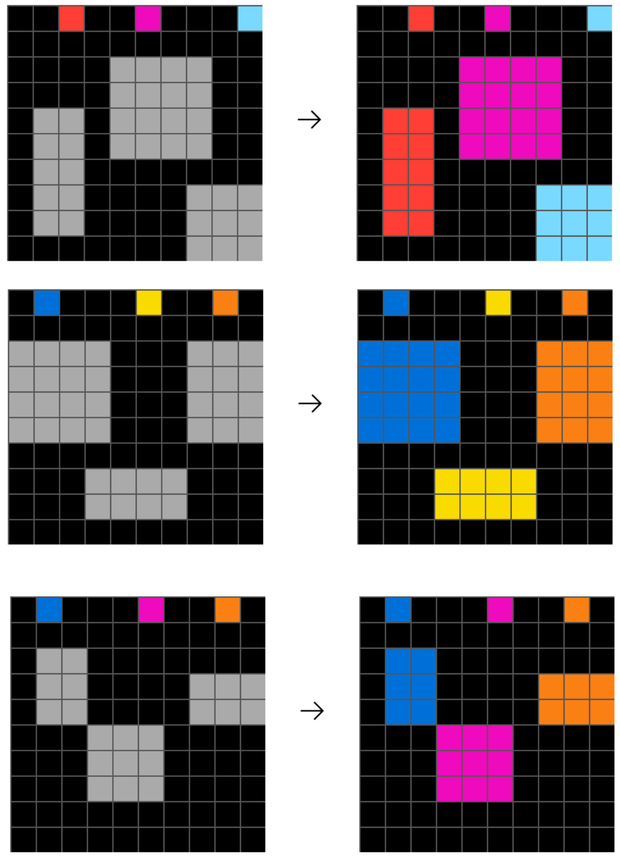
```
    "nl_description": "TO_BE_FILLED",
    "code": "TO_BE_FILLED" ...
```

Our Solution: Hybrid Synthesis



Sampling LLM completions

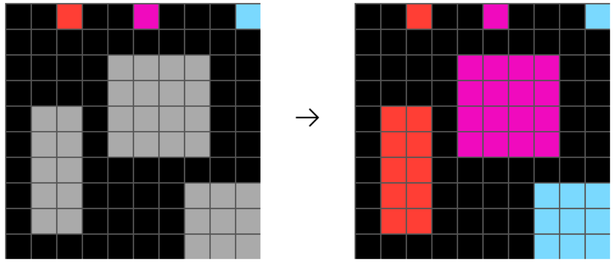
Expected Solution → **if** color_of(**self**) = GREY ∧ is_neighbor(**self**, other) ∧ size_of(other) = MIN
then update_color(color_of(other))



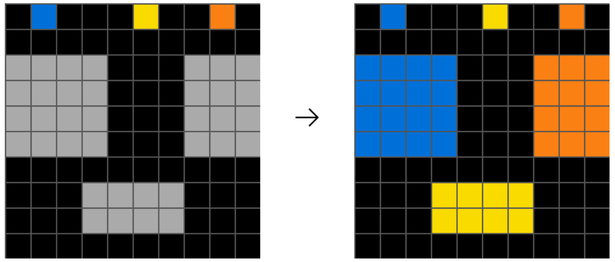
Solution 1 - **if** color_of(**self**) = GREY ∧ is_neighbor(**self**, other)
then update_color(color_of(other))

Sampling LLM completions

Expected Solution → **if** color_of(**self**) = GREY ∧ is_neighbor(**self**, other) ∧ size_of(other) = MIN
then update_color(color_of(other))

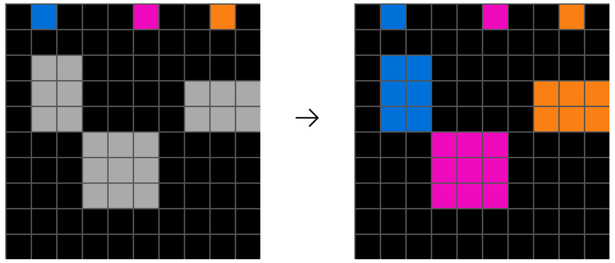


Solution 1 - **if** color_of(**self**) = GREY ∧ is_neighbor(**self**, other)
then update_color(color_of(other))



Solution 2 - **if** is_neighbor(**self**, other) ∧ color_of(other) = GREY
then update_color(color_of(other))

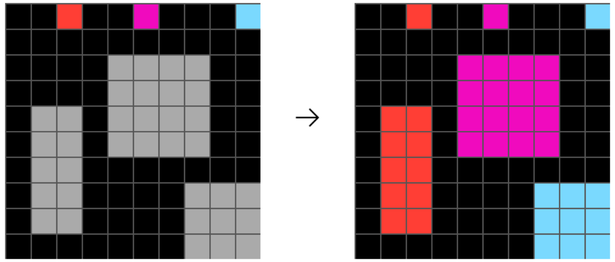
Solution 3 - **if** color_of(**self**) = GREY
then update_color(color_of(other))



Solution 4 - **if** not (color_of(**self**) = GREY) ∧ is_neighbor(**self**, other) ∧ color_of(other) = GREY
then update_color(FUCHSIA)

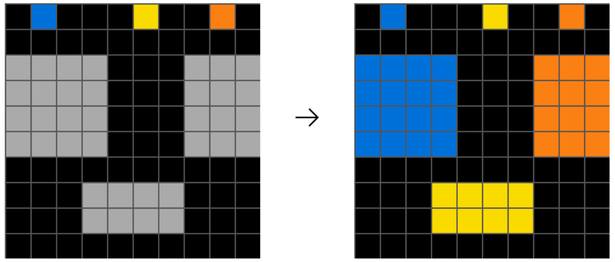
Sampling LLM completions

Expected Solution → **if** color_of(**self**) = GREY ∧ is_neighbor(**self**, other) ∧ size_of(other) = MIN
then update_color(color_of(other))



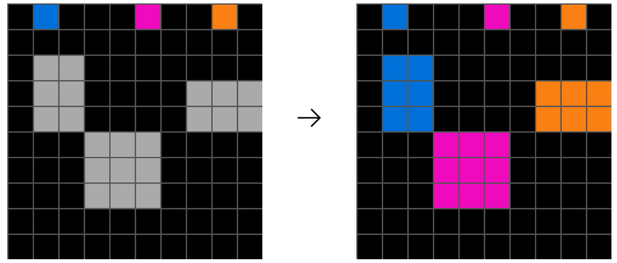
Idea: LLM Solutions have correct components so can be used to guide search

Solution 1 - **if** color_of(**self**) = GREY ∧ is_neighbor(**self**, other)
then update_color(color_of(other))



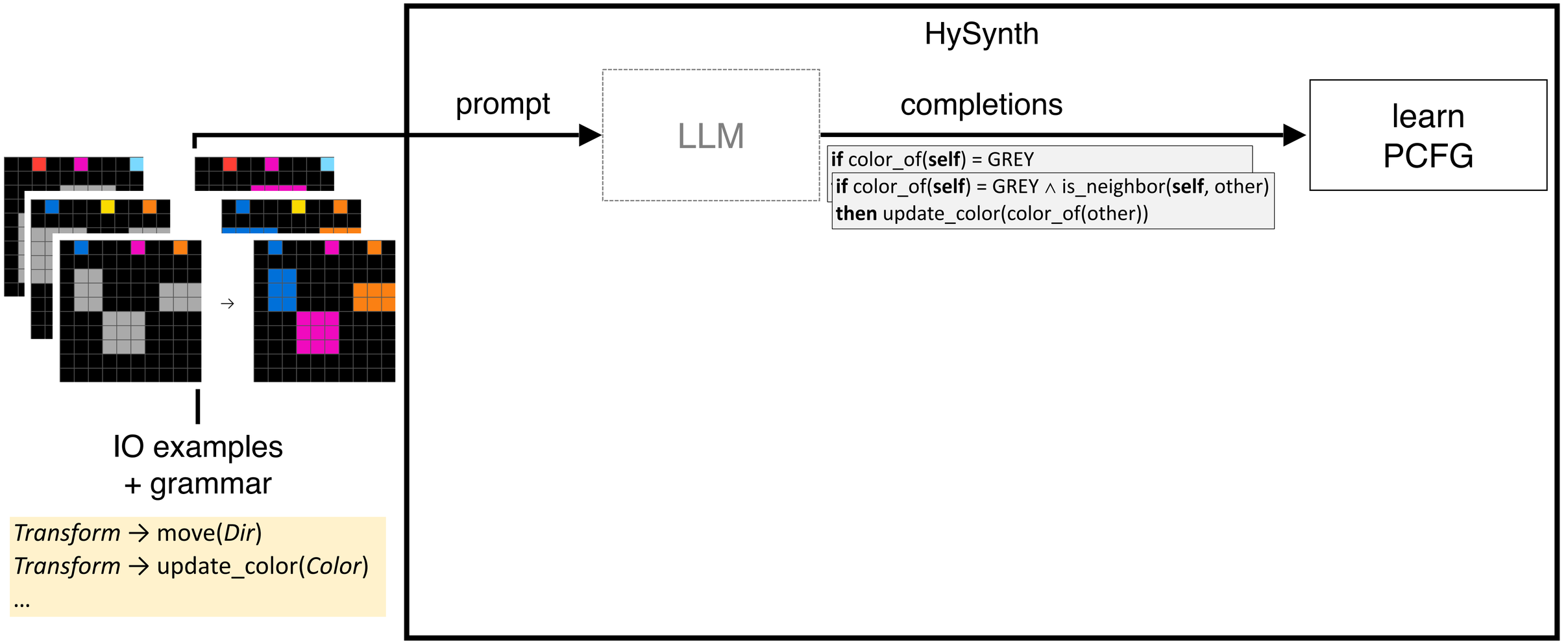
Solution 2 - **if** is_neighbor(**self**, other) ∧ color_of(other) = GREY
then update_color(color_of(other))

Solution 3 - **if** color_of(**self**) = GREY
then update_color(color_of(other))



Solution 4 - **if** not (color_of(**self**) = GREY) ∧ is_neighbor(**self**, other) ∧ color_of(other) = GREY
then update_color(FUCHSIA)

Our Solution: Hybrid Synthesis



Learning a PCFG

Solution 1 - **if** color_of(**self**) = GREY ∧ is_neighbor(**self**, other)
then update_color(color_of(other))

Solution 2 - **if** is_neighbor(**self**, other) ∧ color_of(other) = GREY
then update_color(color_of(other))

Solution 3 - **if** color_of(**self**) = GREY
then update_color(color_of(other))

Solution 4 - **if** not (color_of(**self**) = GREY) ∧ is_neighbor(**self**, other)
∧ color_of(other) = GREY **then** update_color(FUCHSIA)

……

LLM Solutions

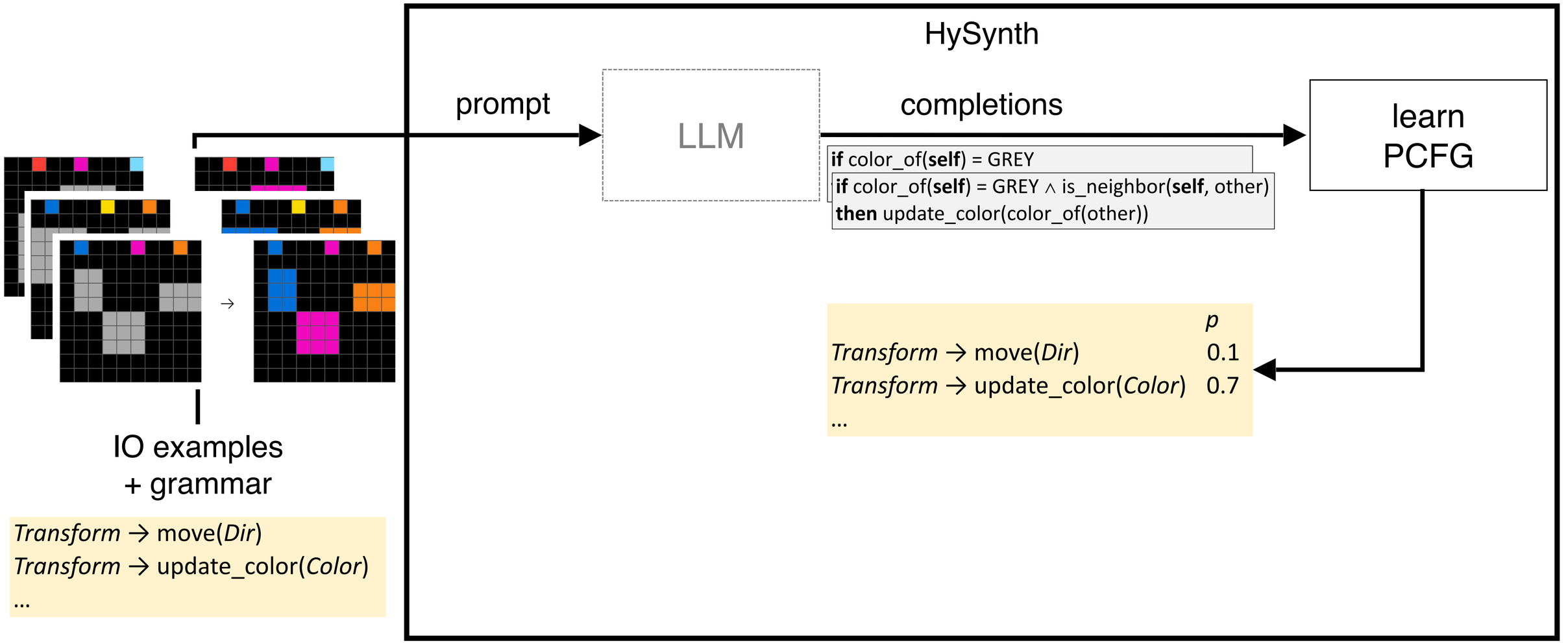
Maximum likelihood estimation

	p
Transform → move(*Dir*)	0.1
Transform → update_color(*Color*)	0.7
…	

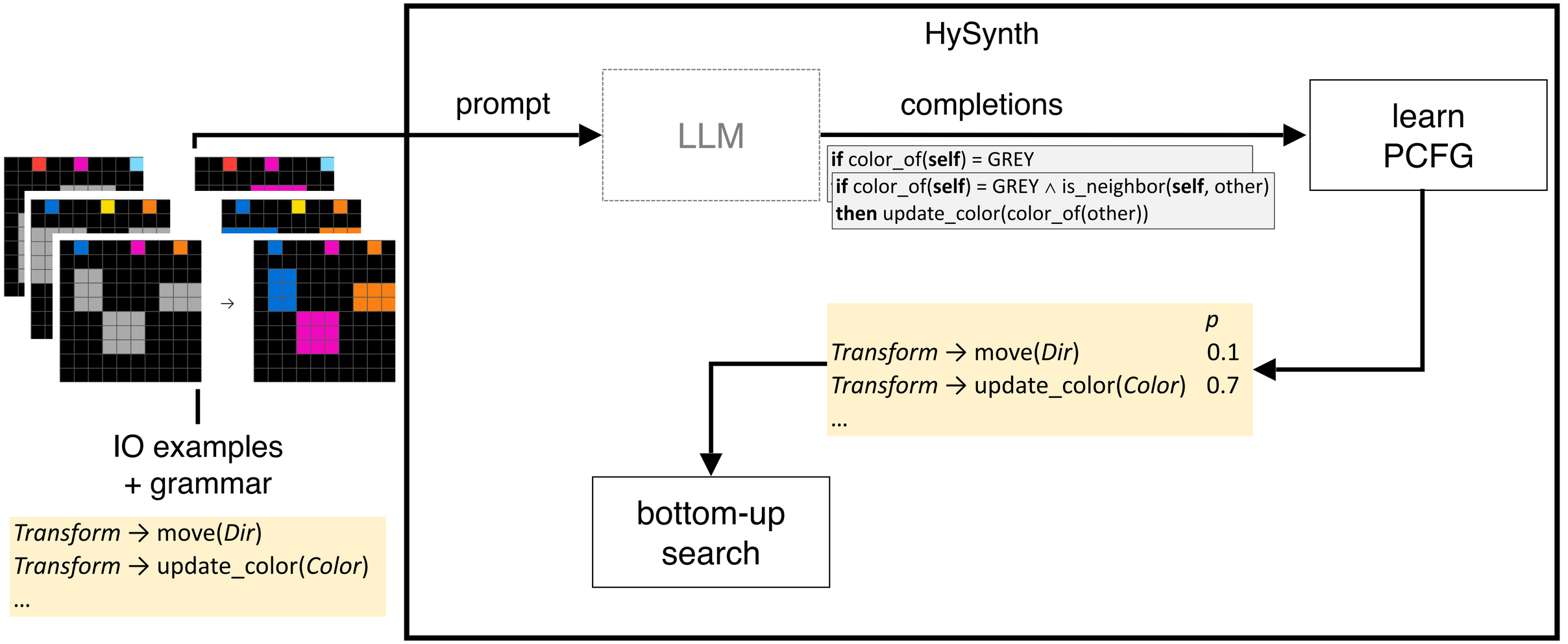
PCFG

More frequent operators have a higher probability in the PCFG

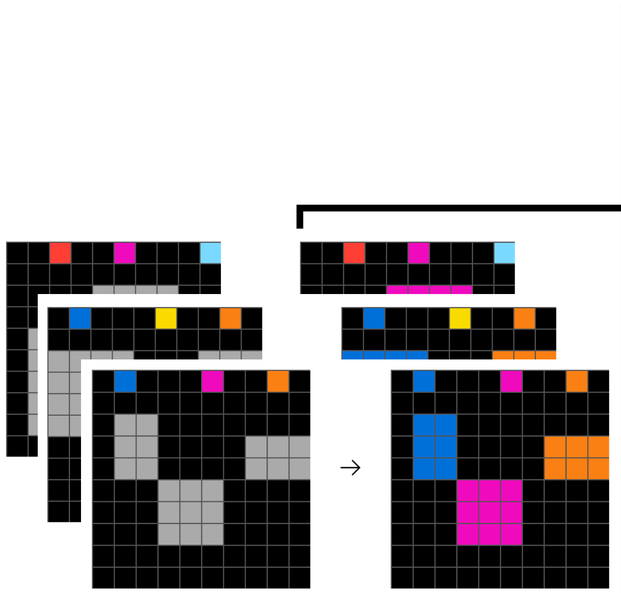
Our Solution: Hybrid Synthesis



Our Solution: Hybrid Synthesis



Our Solution: Hybrid Synthesis



IO examples
+ grammar

Transform → move(*Dir*)
Transform → update_color(*Color*)
…

HySynth

prompt

LLM

completions

if color_of(**self**) = GREY
if color_of(**self**) = GREY ∧ is_neighbor(**self**, other)
then update_color(color_of(other))

learn
PCFG

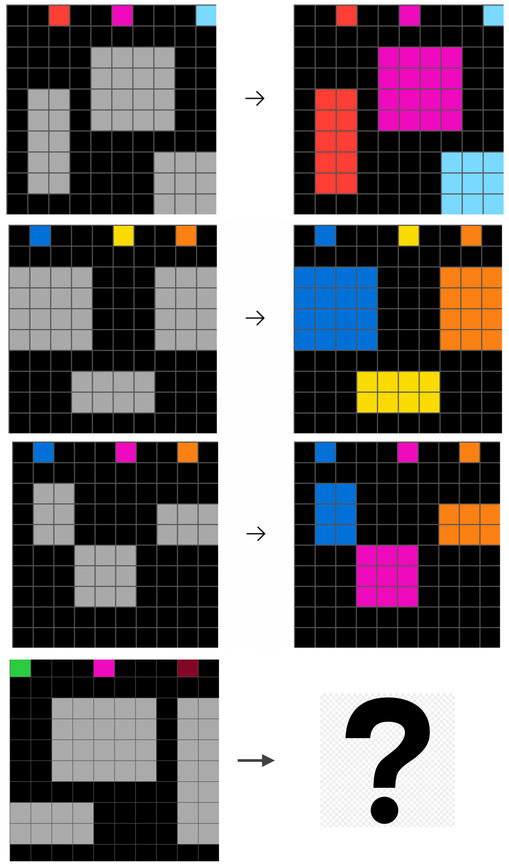
	p
Transform → move(*Dir*)	0.1
Transform → update_color(*Color*)	0.7
…	

bottom-up
search

if color_of(**self**) = GREY ∧ is_neighbor(**self**, other)
∧ size_of(other) = MIN
then update_color(color_of(other))

solution

Experimental Set-up: Benchmarks



ARC Domain (160 tasks)

```
Input Tensor:
in = [[0, 1, 0, 0],
      [0, 1, 1, 0],
      [1, 1, 1, 1]]

Output Tensor:
out = [[0.0, 1.0, 0.0, 0.0],
       [0.0, 0.5, 0.5, 0.0],
       [0.25, 0.25, 0.25, 0.25]]

Natural language description
(optional):

"Normalize the rows of a tensor"
```

Tensor Domain (69 tasks)

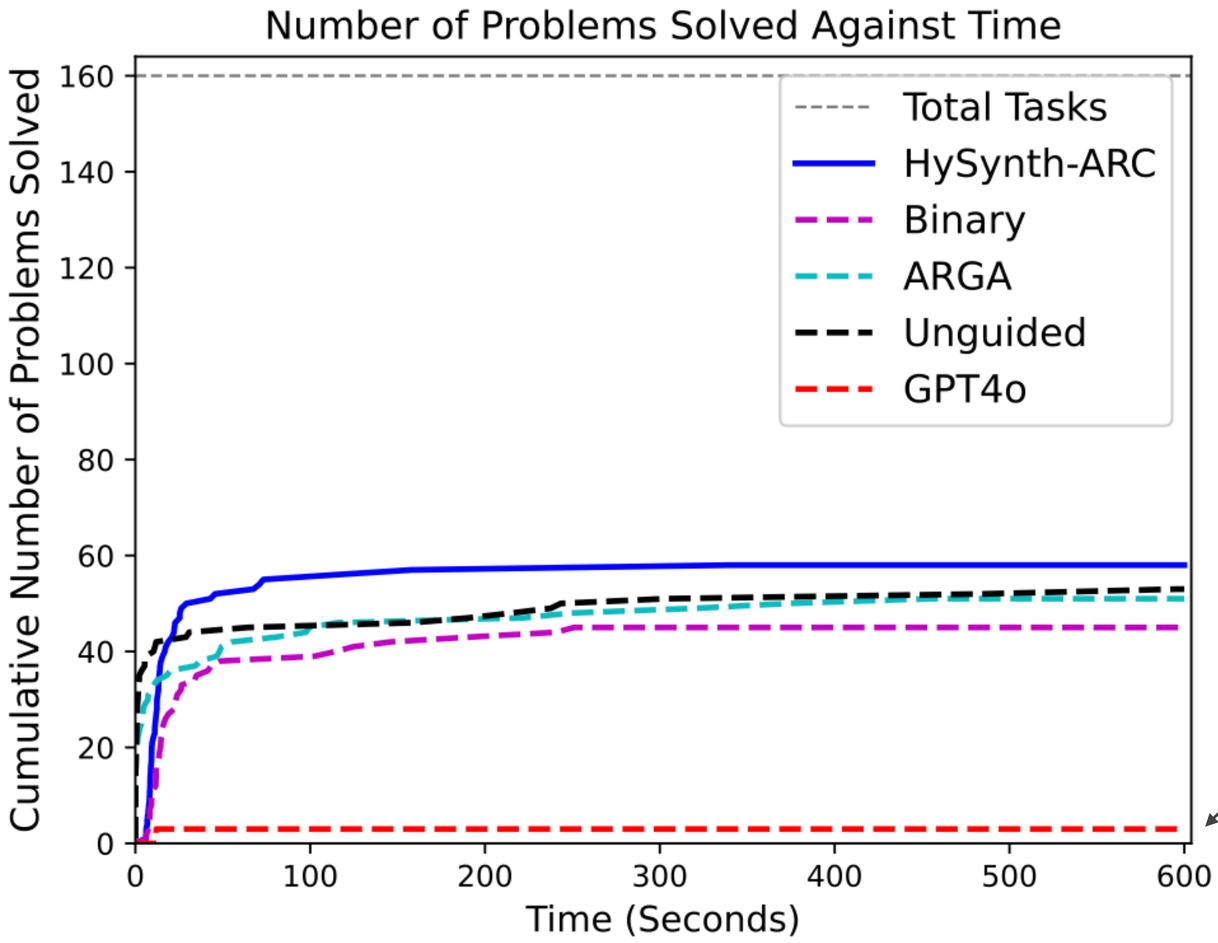
	A	B
1	Name and ID	First name and last name
2	Thomas, Rhonda 82132	Rhonda Thomas
3	Emmett, Keara 34231	Keara Emmett
4	Vogel, James 32493	James Vogel
5	Jelen, Bill 23911	Bill Jelen
6	Miller, Sylvia 78356	Sylvia Miller
7	Lambert, Bobby 25900	Bobby Lambert

String Domain (70 tasks)

Experimental Set-up: Ablations

1. Domain-specific Synthesizer
2. Unguided Search (uniform grammar)
3. No Search (Direct Sampling from LLM)
4. Binary Surrogate Model (Remove operators not occurring in LLM solutions)

Evaluation (HySynth vs Baselines)



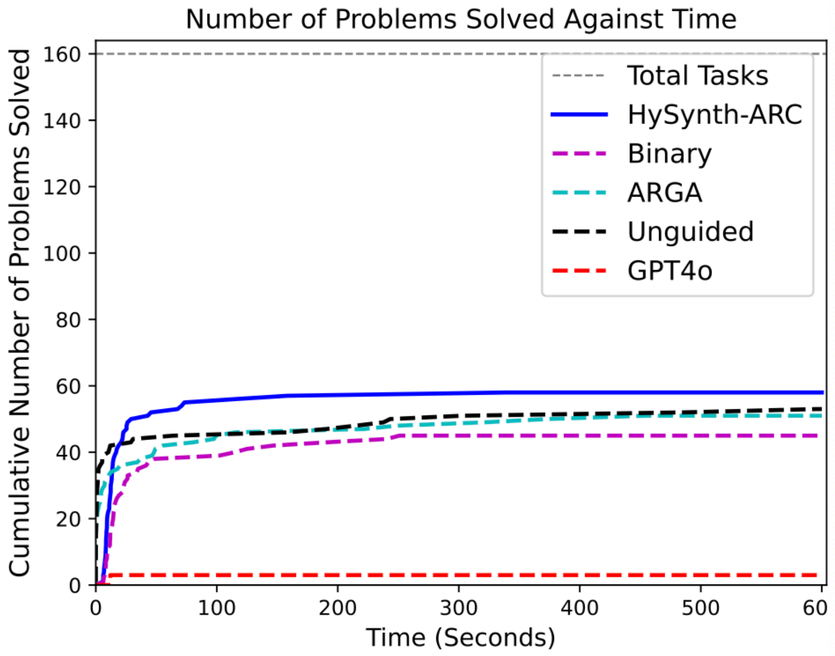
HySynth performs best

LLM solves least number of problems

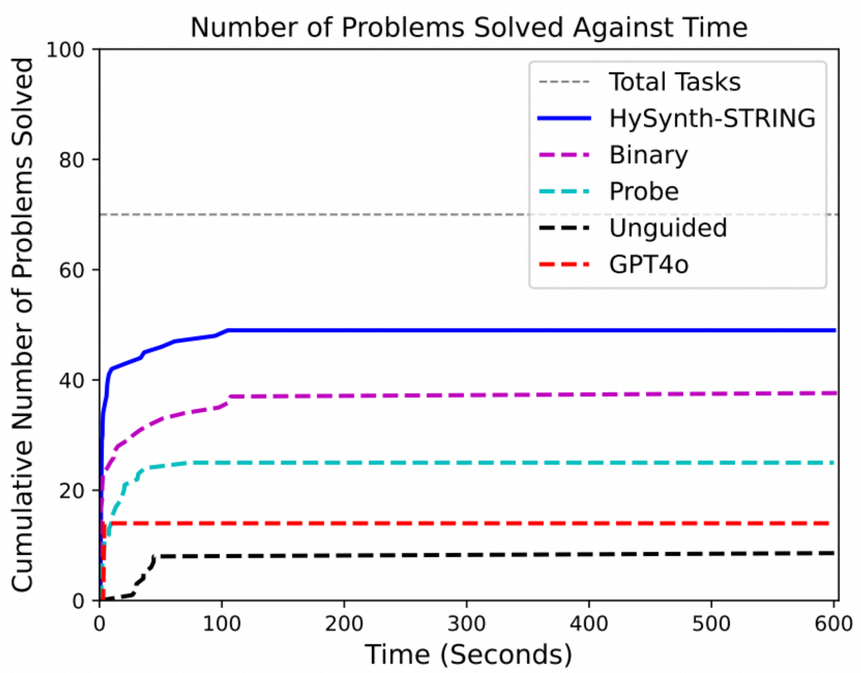
[Khalil 2023] Graphs, Constraints, and Search for the Abstraction and Reasoning Corpus

Evaluation (HySynth vs Baselines)

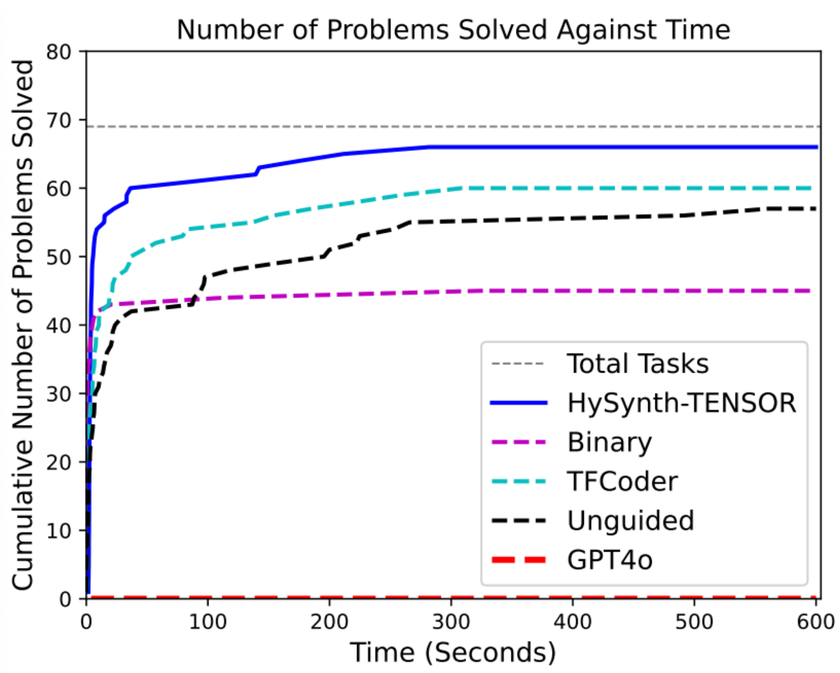
HySynth outperforms LLMs, unguided search and baseline synthesizers on all 3 domains



ARC Domain



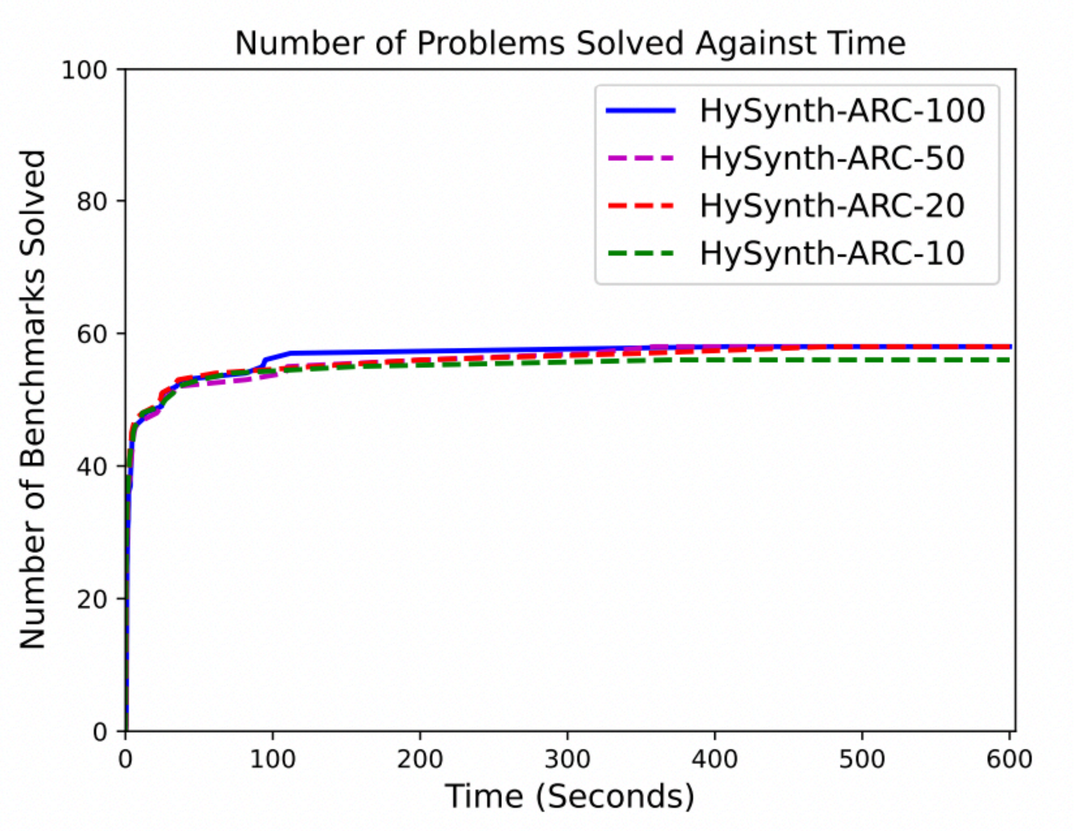
String Domain



Tensor Domain

[Shi 2022] TF-Coder: Program Synthesis for Tensor Manipulations

HySynth: Ablation of different sample sizes

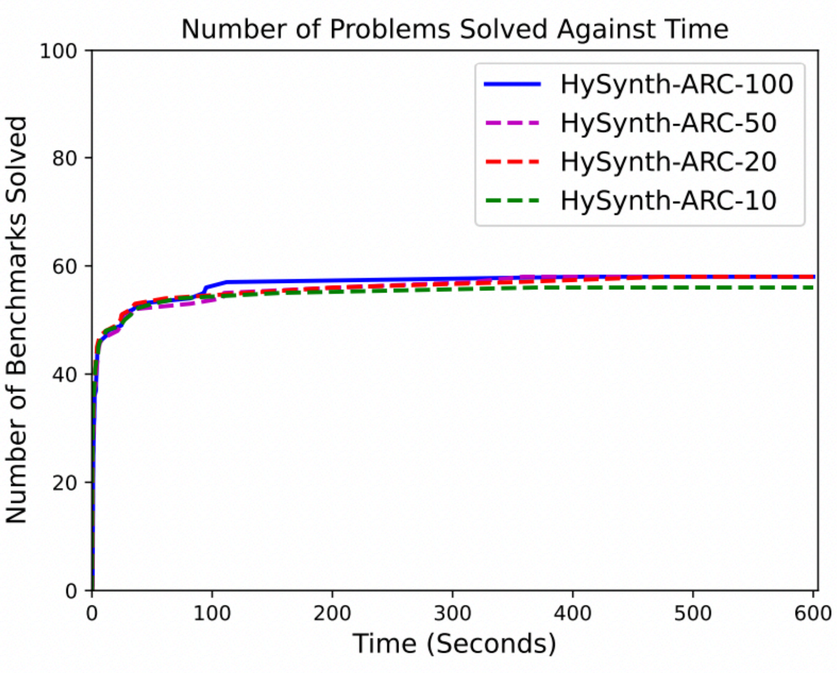


ARC Domain

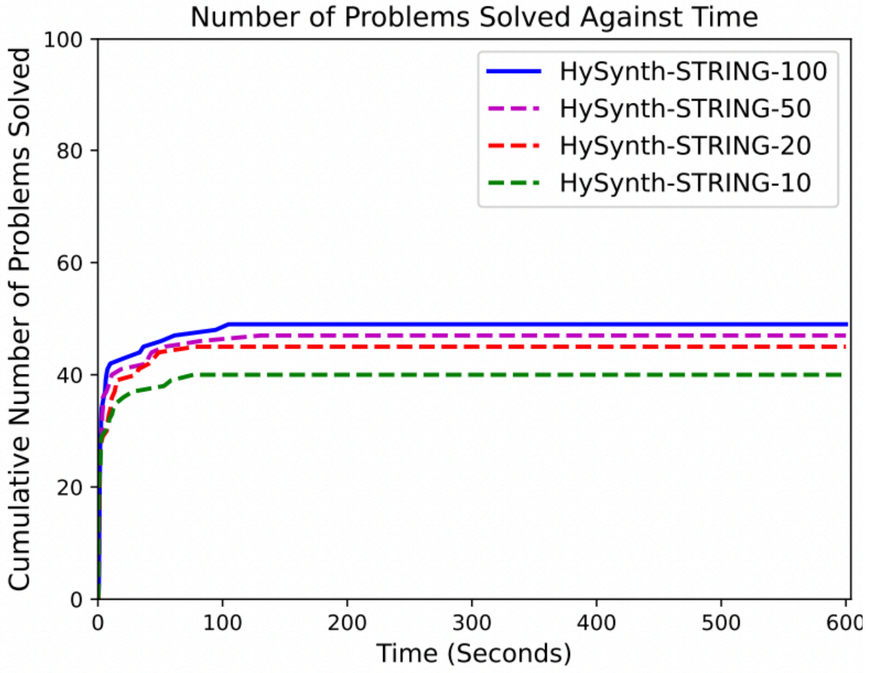
Sample sizes
N = 10, 20, 50, 100

HySynth: Ablation of different sample sizes

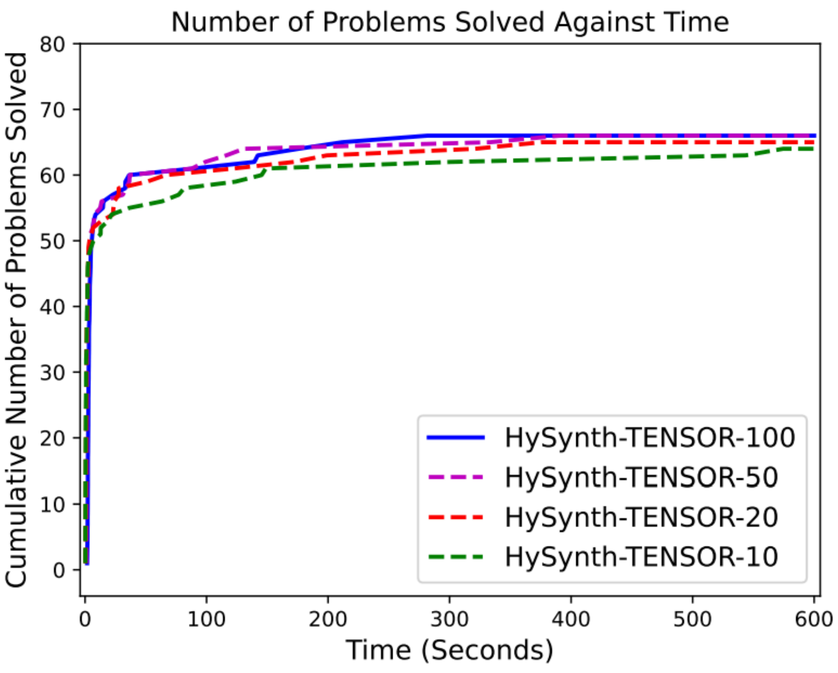
Few samples are sufficient to train a robust surrogate model



ARC Domain



String Domain



Tensor Domain

HySynth Conclusion

LLM Guidance + Symbolic Search

1. Learn a surrogate model from valid and invalid LLM completions
2. Guided search using PCFG, an alternate to large-scale LLM sampling
3. HySynth outperforms unguided, LLM and baseline across 3 domains