Neural Task Synthesis for Visual Programming

Victor-Alexandru Pădurean, Georgios Tzannetos, Adish Singla

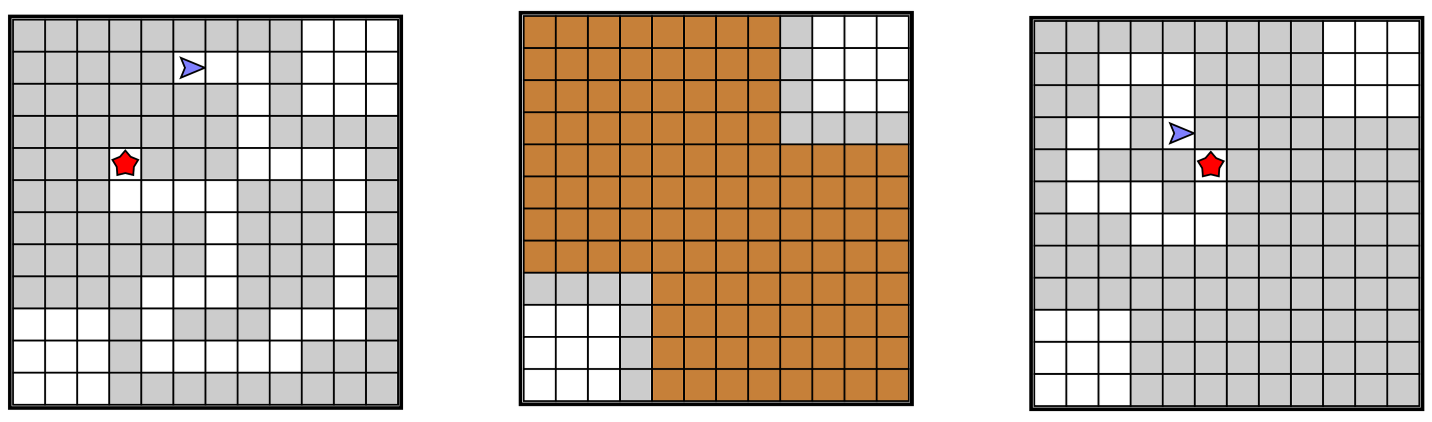
Max Planck Institute for Software Systems, Germany

Motivation and Overview

- We explore the role of generative AI in visual programming domains such as *Hour of Code:Maze Challenge* by Code.org and Karel.
- The available set of tasks on existing platforms is very limited, posing a major hurdle for novice students in mastering the missing concepts.
- We develop a novel technique to synthesize tasks for a given specification.
- Our technique can enable AI systems to provide personalized feedback to students as new simpler tasks and worked examples for scaffolding.

Student is working on task T^{ref}, struggling with obtaining a solution code C^{ref}. → AI system extracts the specifications ψ^{in} for a simpler task. → AI system generates a task T^{out} based on ψ^{in}, and delivers it to student.

(a) An interaction between student and AI system



(b)	(c)	(d)
Allowed blocks: {move, turnLeft, turnRight, RepeatUntil, IfElse}	a blocks is a body of basic action blocks from the set {move, turnLeft, turnRight}	Allowed blocks: {move, turnLeft, turnRight, RepeatUntil, IfElse}
	b is a boolean condition from {pathAhead, pathLeft, pathRight}	
Maximum size: 7	Number of blocks should be ≤ 7	Maximum size: 7

```
def Run(){
  RepeatUntil(goal){
    If(pathAhead){
      move
    }
    Else{
      If(pathRight){
        turnRight
      }
      Else{
        turnLeft
      }
    }
  }
}
```

(b) T^{ref} and solution C^{ref}

```
def Run(){
  a blocks
  RepeatUntil(goal){
    a blocks
    If(b){
      a blocks
    }
    Else{
      a blocks
    }
    a blocks
  }
}
```

(c) Specifications ψ^{in}

```
def Run(){
  RepeatUntil(goal){
    If(pathRight){
      turnRight
    }
    Else{
      turnLeft
      move
    }
    move
  }
}
```

(d) T^{out} and C^{out}

Figure 1: Illustration of an AI system providing a new simpler task to a student

Problem Setup

Visual Programming Task $T := (T_{IO}, T_{code})$

- T_{IO} denotes the visual puzzle
- T_{code} denotes additional constraints on a solution code

Solution Code C **of a Task** T

- C successfully solves T_{IO}
- C respects T_{code}

Task Synthesis Specification $\psi := (\psi_{IO}, \psi_{sketch}, \psi_{\Delta})$

- ψ_{IO} is a partially initialized visual puzzle of the task to be synthesized
- ψ_{sketch} and ψ_{Δ} capture the constraints that should be followed by solution codes of the task to be synthesized

Synthesis Objective for T^{out} **given** ψ^{in}

- **O1:Validity.** T^{out} respects ψ^{in} and there exists one solution C for T^{out}
- **O2:Concepts.** T^{out} conceptually captures specification ψ^{in}, i.e., its solution codes respect the concepts and complexity of ψ^{in}_{sketch}
- **O3:Trace.** The quality of execution trace of solution codes on T^{out}
- **O4:Overall.** All the above objectives (O1, O2, O3) are satisfied
- **O5:Human.** The quality of T^{out} from a human expert's point of view

Our Technique NeurTaskSyn

Purely Neural Techniques

- The generative process is highly brittle and the output task could be incorrect w.r.t. the input specification.

Purely Symbolic Techniques

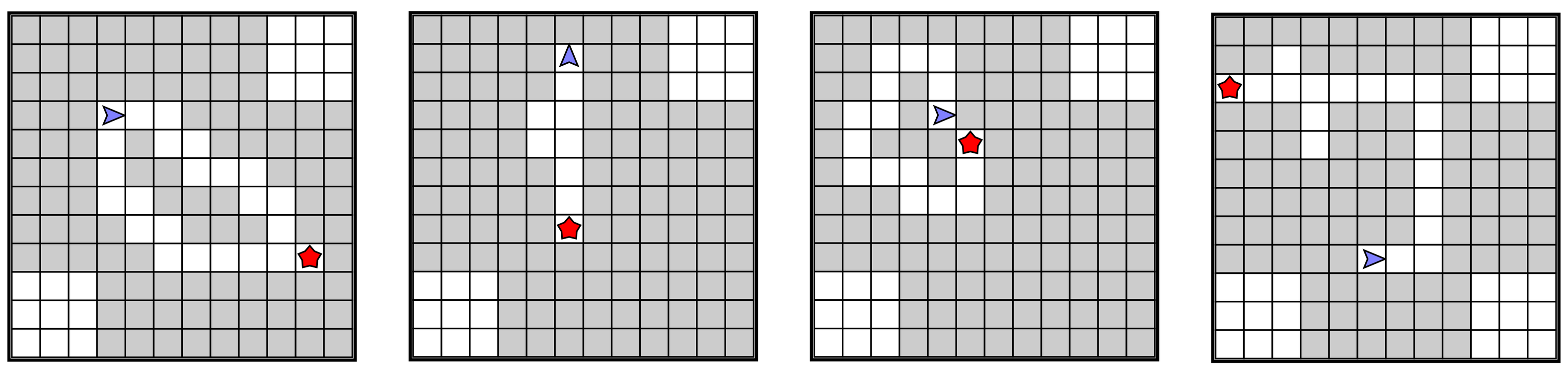
- The generative process is time-inefficient and not suitable for applications that require real-time synthesis.

Neurally-guided Symbolic Engine

- We develop NeurTaskSyn that can synthesize high-quality tasks while being robust and efficient.

Symbolic Engine ⟷ Neural Model

Illustrative Example



```
def Run(){
  RepeatUntil(goal){
    If(pathLeft){
      turnLeft
      move
    }
    Else{
      move
    }
  }
}
```

(a) T^{out}_{IO}, C^{out} by GPT4TaskSyn

```
def Run(){
  turnLeft
  turnLeft
  RepeatUntil(goal){
    If(pathRight){
      move
    }
    Else{
      move
    }
  }
}
```

(b) T^{out}_{IO}, C^{out} by BaseTaskSyn

```
def Run(){
  RepeatUntil(goal){
    If(pathRight){
      turnRight
    }
    Else{
      turnLeft
      move
    }
    move
  }
}
```

(c) T^{out}_{IO}, C^{out} by NeurTaskSyn

```
def Run(){
  RepeatUntil(goal){
    If(pathAhead){
      move
    }
    Else{
      turnLeft
    }
  }
}
```

(d) T^{out}_{IO}, C^{out} by Expert

Figure 2: Outputs for Specification ψ^{in} from Figure 1

Experimental Results

ψ^{in}	ψ^{in}_{sketch} structure	(depth, constructs)	ψ^{in}_{IO}	ψ^{in}_{Δ}	Source
ψ0	{Run {Repeat}}	(2, 1)	16x16 empty	HoCMaze, blocks $\leqslant 10$	HoC:Maze9
ψ1	{Run {RepeatUntil}}	(2, 1)	16x16 empty	HoCMaze, blocks $\leqslant 10$	HoC:Maze13
ψ2	{Run {Repeat; Repeat}}	(2, 2)	16x16 empty	HoCMaze, blocks $\leqslant 10$	HoC:Maze8
ψ3	{Run {RepeatUntil{IfElse}}}	(3, 2)	16x16 empty	HoCMaze, blocks $\leqslant 10$	HoC:Maze18
ψ4	{Run {RepeatUntil{If; If}}}	(3, 3)	16x16 empty	HoCMaze, blocks $\leqslant 10$	HoC:Maze20
ψ5	{Run}	(1, 0)	16x16 empty	Karel, blocks $\leqslant 10$	Karel:OurFirst
ψ6	{Run {While}}	(2, 1)	16x16 empty	Karel, blocks $\leqslant 10$	Karel:Diagonal
ψ7	{Run {While; While}}	(2, 2)	16x16 empty	Karel, blocks $\leqslant 10$	Karel:RowBack
ψ8	{Run {While{If}}}	(3, 2)	16x16 empty	Karel, blocks $\leqslant 10$	Karel:Stairway
ψ9	{Run {While{Repeat}}}	(3, 2)	16x16 empty	Karel, blocks $\leqslant 10$	Karel:CleanAll

Technique	O1:Validity	O2:Concepts	O3:Trace	O4:Overall	O5:Human
NeurTaskSyn$_{c:10,p:100}$	1.00	0.83	0.80	0.80	0.77
BaseTaskSyn$_{c:10,p:100}$	0.97	0.37	0.33	0.33	0.20
GPT4TaskSyn-*converse*	0.97	0.57	0.60	0.43	0.27
GPT4TaskSyn-*fewshot*	0.80	0.37	0.57	0.33	0.13
Expert	1.00	1.00	1.00	1.00	1.00

Conclusions and Future Work

- We propose NeurTaskSyn, a novel neuro-symbolic technique for synthesizing visual programming tasks.
- Our results show that GPT-4 faces challenges in code execution, symbolic operations, and visual planning.
- Visual programming domains can serve as benchmarks for assessing the capabilities of generative models.
- It would be interesting to fine-tune LLMs to improve their capabilities for visual programming domains.

NeurIPS'23 Workshop on Generative AI for Education (GAIED)