Tsinghua University

Parametric Visual Program Induction

A Function Modularization Solution and Monto-Carlo Tree-Search Learning

Authors: **Xuguang Duan**, Xin Wang, Ziwei Zhang, Wenwu Zhu

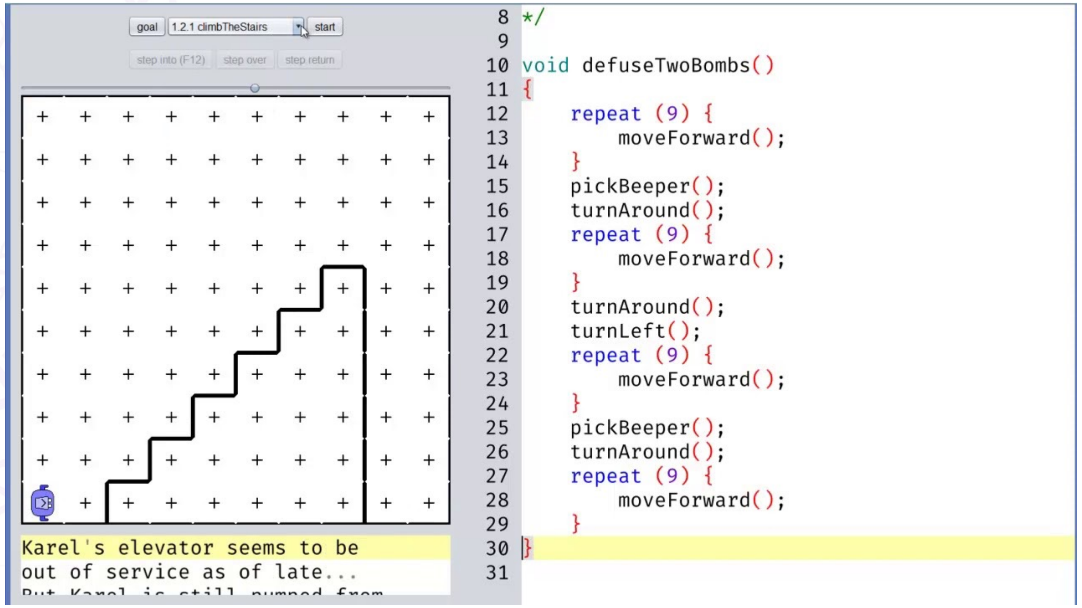
Department of Computer Science and Technology, Tsinghua University

Contact: duan_xg@outlook.com; xin_wang@tsinghua.edu.cn;

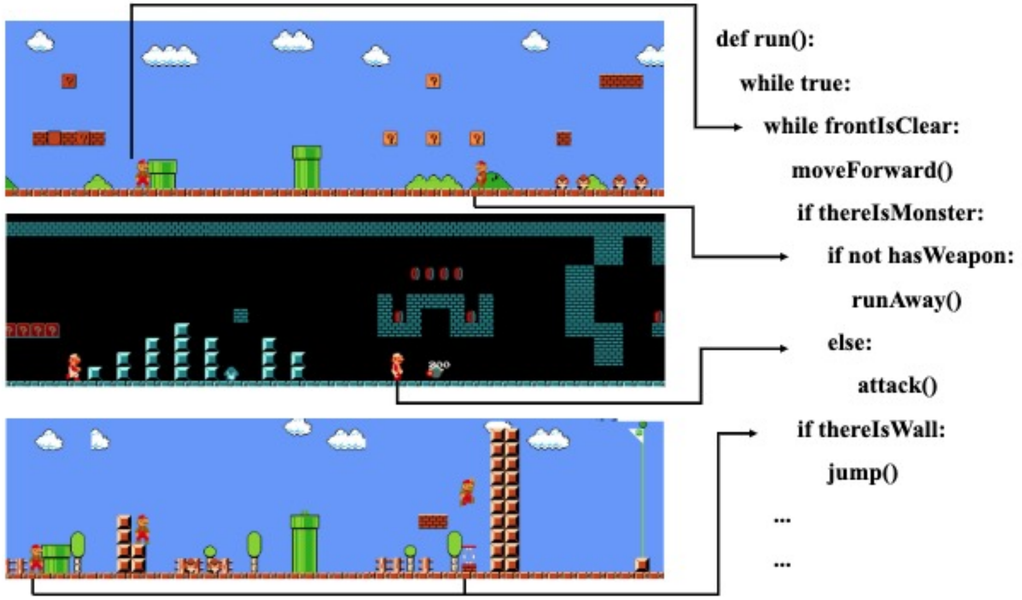
Introduction

Program Induction, or Program Learning..

aims to generate a program to describe the underlying logic or patterns:



(a) A program to control the ***Karel*** "robot".



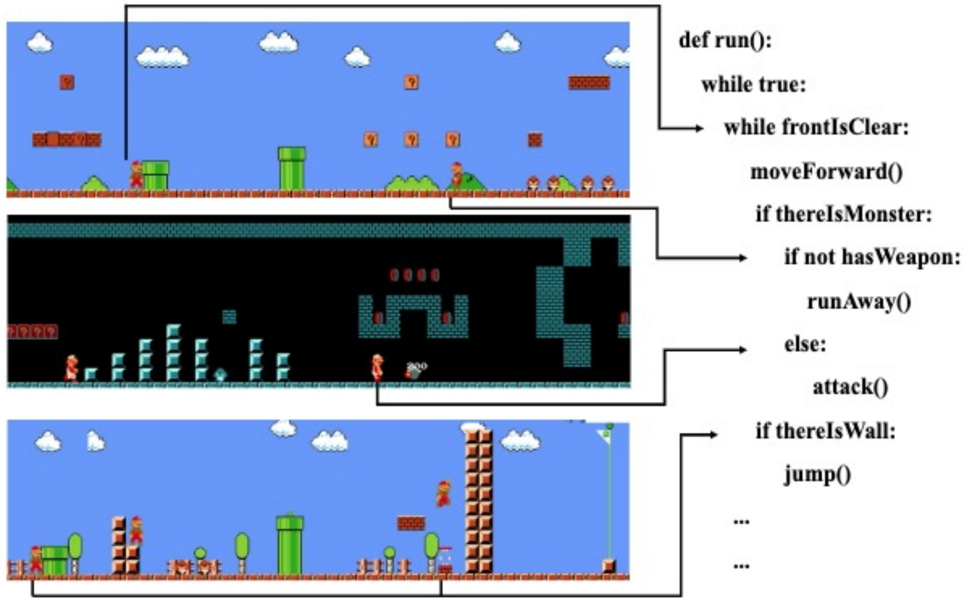
(b) A program to describe the logic under ***Super Mario***.

Introduction

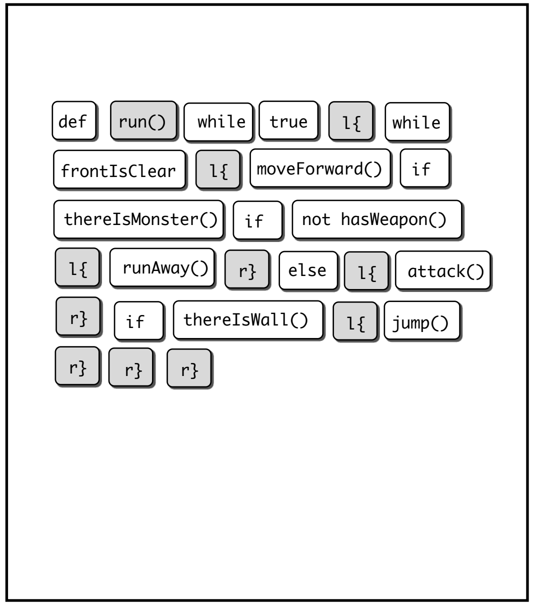
Program Induction, or Program Learning..

is usually solved via program tokneization and then Learning or/and Searching.

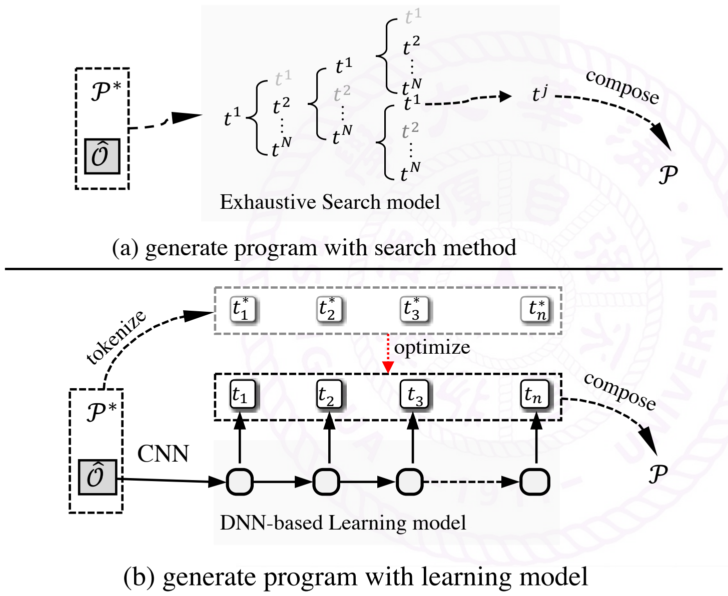
program tokneization: <u>split a program into symbolic tokens:</u>



(I) Super Mario and its game logic;



(II) tokenized program;

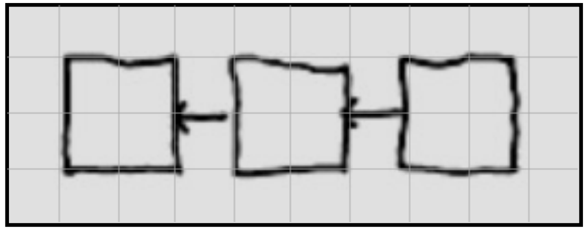


(III) how to generate a proper program.

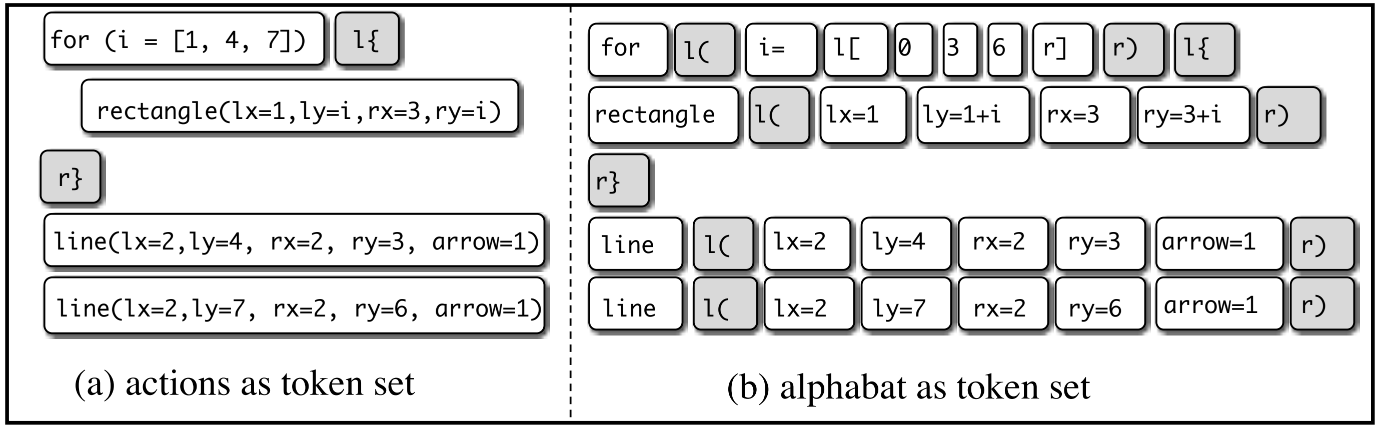
Introduction

However, When it comes to Visual Program Induction:

two challenges raises due to the huge program space...
Firstly, the modeling of functions:



```
for (i = [0, 3, 6]){
  rectangle(lx=1,ly=1+i,rx=3,ry=3+i);
}
line(lx=2,ly=4, rx=2, ry=3, arrow=1);
line(lx=2,ly=7, rx=2, ry=6, arrow=1);
```



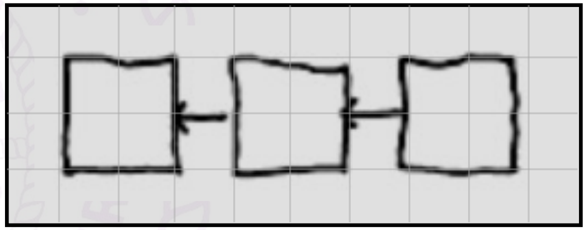
(a) actions as token set
(b) alphabat as token set

We may obtain a huge token set by **actions as token set** or we have to deal with the fragile program syntax with **alphabat as token set**

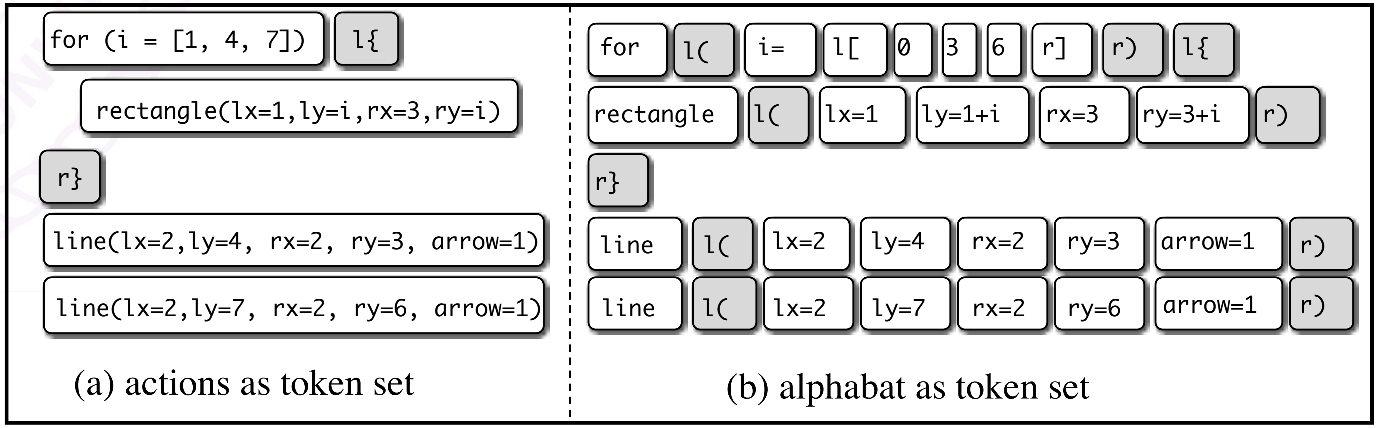
Introduction

However, When it comes to Visual Program Induction:

two challenges raises due to the huge program space...
Firstly, the modeling of functions:

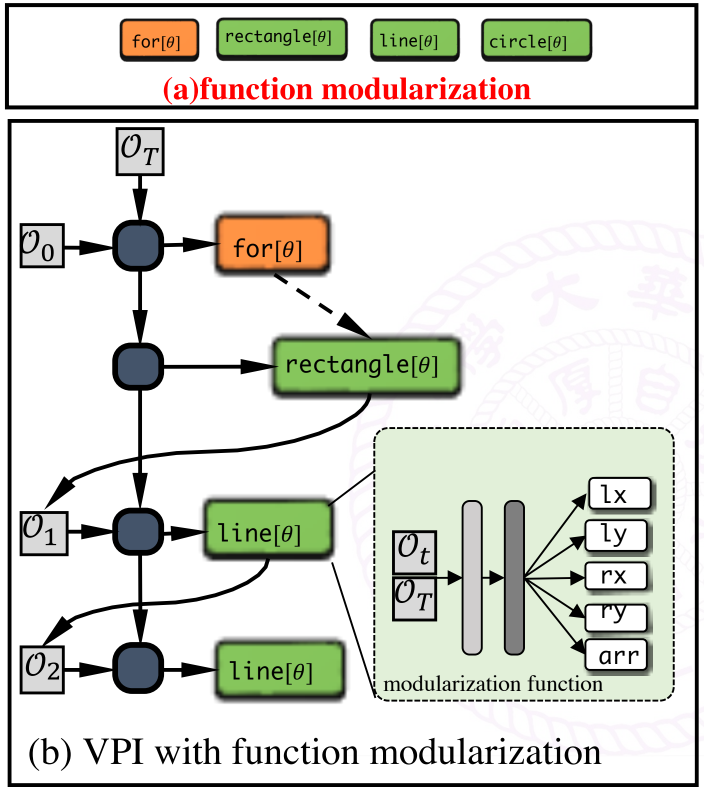


```
for (i = [0, 3, 6]){
  rectangle(lx=1,ly=1+i,rx=3,ry=3+i);
}
line(lx=2,ly=4, rx=2, ry=3, arrow=1);
line(lx=2,ly=7, rx=2, ry=6, arrow=1);
```



(a) actions as token set

(b) alphabat as token set



(b) VPI with function modularization

By using function modules; we could seperately learn the function dynamics and function parameters.

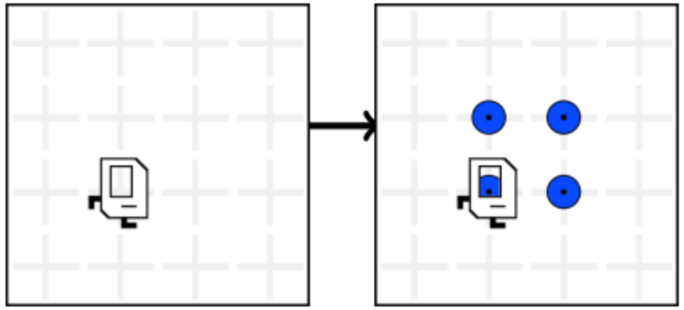
Tsinghua University

Method

However, When it comes to Visual Program Induction:

two challenges raises due to the huge program space...

Secondly, the learning of programs:

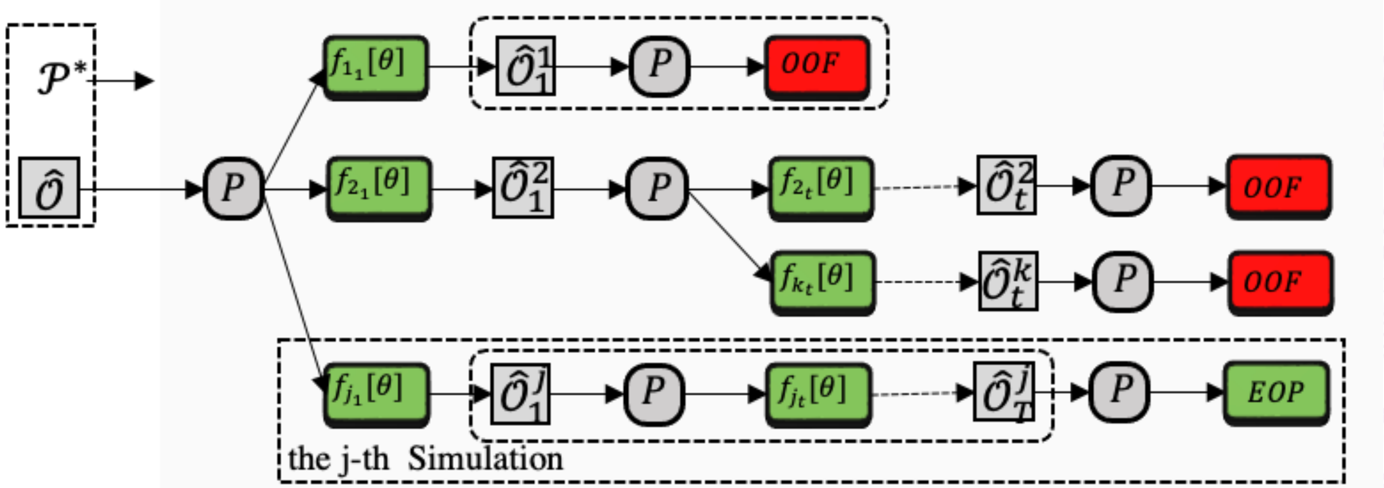


Program A

```
def run():
  repeat(4):
    putMarker()
    move()
    turnLeft()
```

Program B

```
def run():
  while(noMarkersPresent):
    putMarker()
    move()
    turnLeft()
```



(Program Guided) Hierarchical Heterogeneous Monto-Carlo Tree-Search

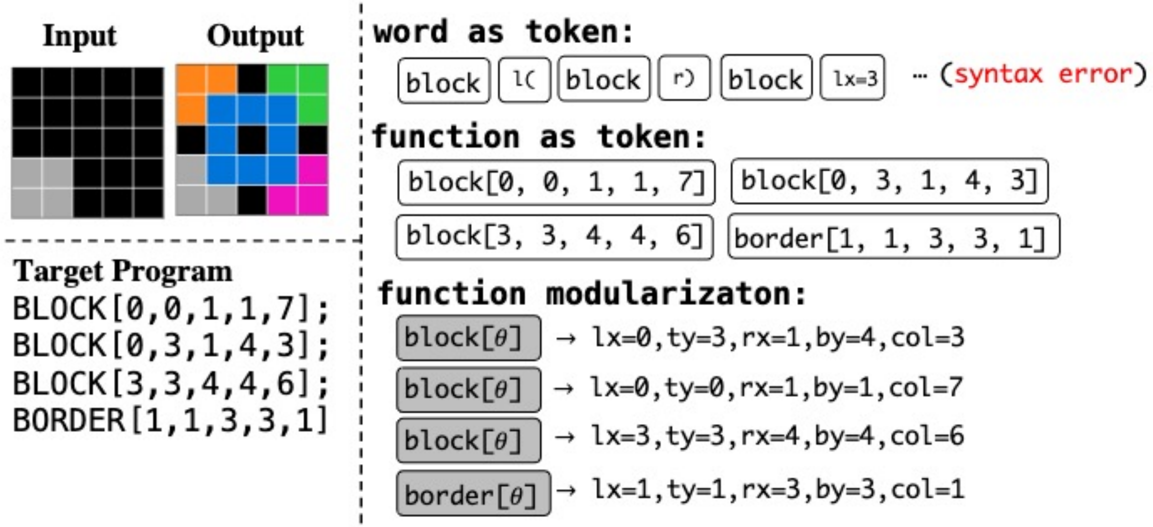
(I) The program alias;

(II) Our solution of Monto-Carlo Tree-Search solution.

Experiment

We test our methods on several datasets:

firstly, the Pixel-Grid dataset: function tokenization vs. function modules



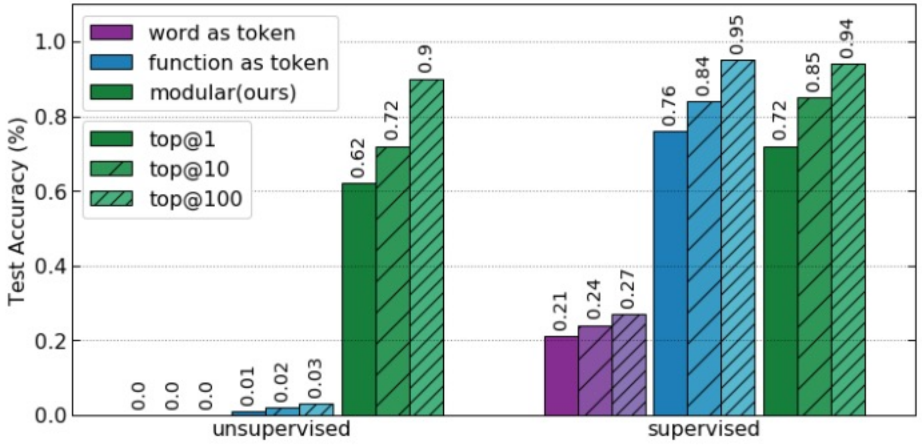
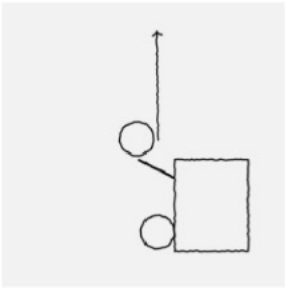


Figure 3. The testing accuracy (%) on the 5×5 Pixel Grid dataset.

Experiment

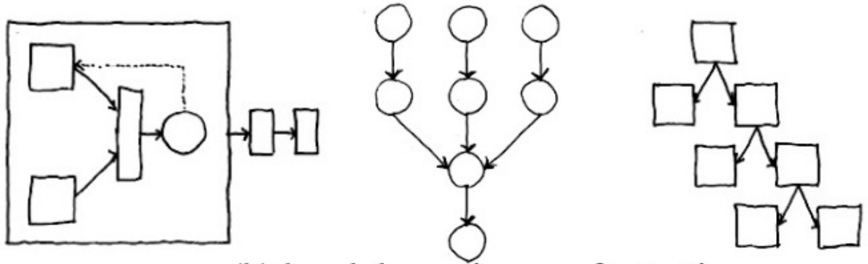
We test our methods on several datasets:

secondly, the Latex-Drawing dataset: Control-free Program Learning

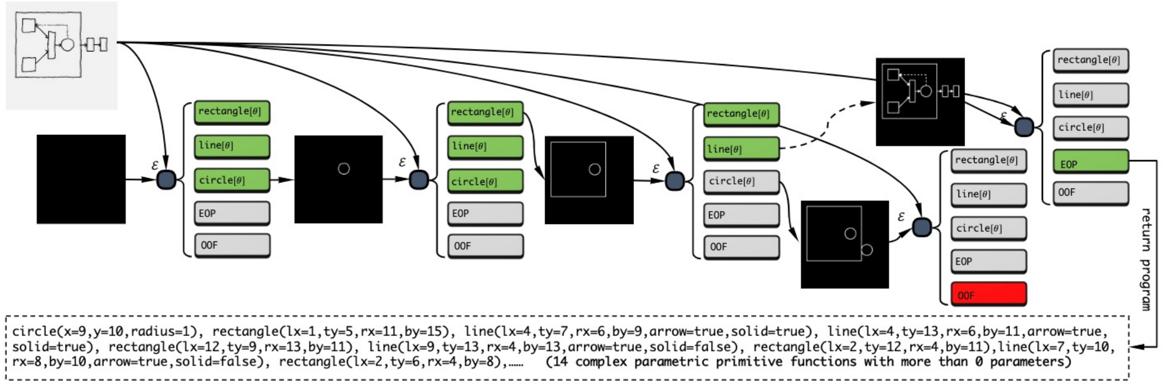


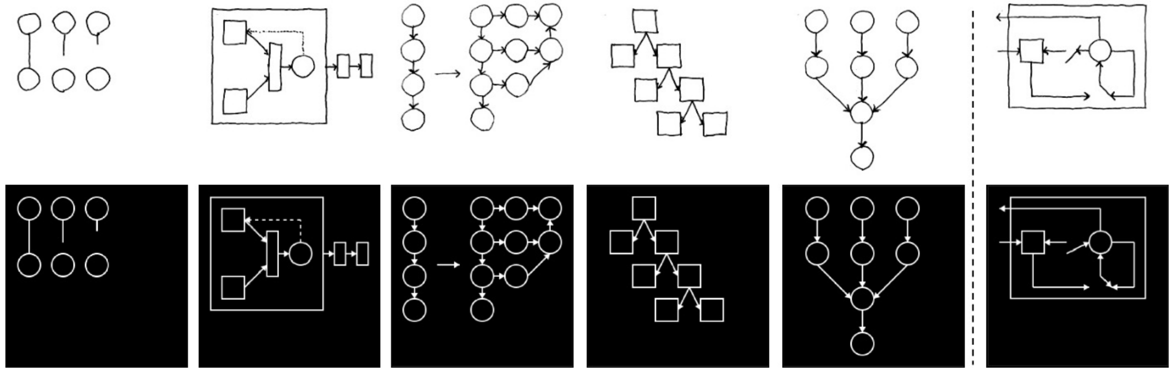
```
circle(x=8,y=8,radius=1),
circle(x=9,y=14, radius=1),
line(x1=9,y1=8,x2=9,y2=1,
      arrow=true,solid=true),
line(x1=8,y1=9,x2=10,y2=10,
      arrow=false,solid=true),
rectangle(x1=10,y1=9,x2=15,y2=15)
```

(a) synthesized observation and its ground truth program.



(b) hand drawn images for testing.



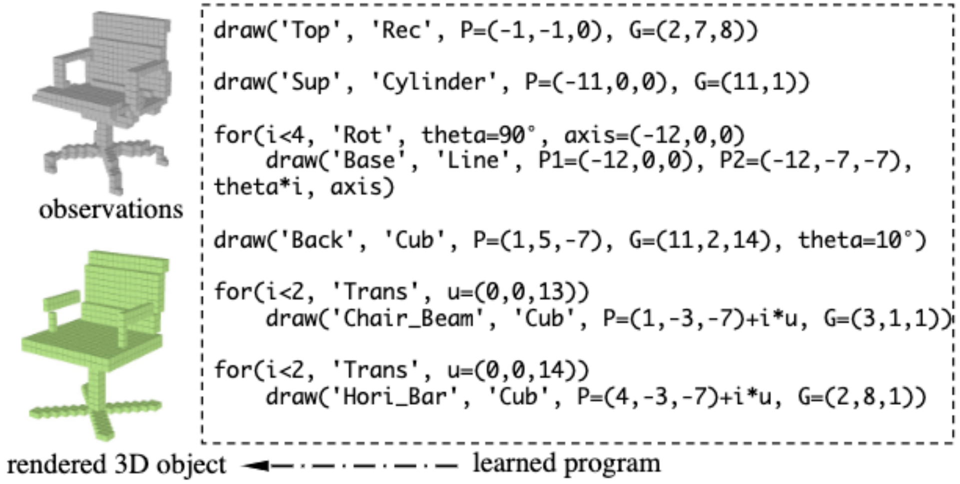


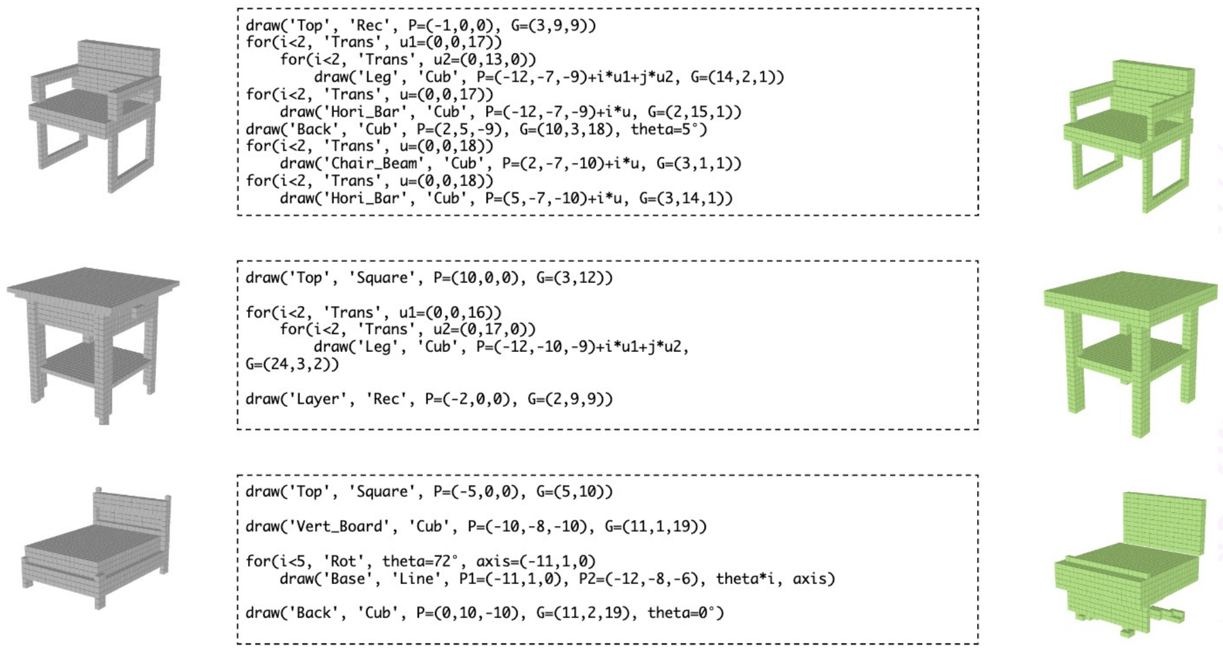
Experiment

We test our methods on several datasets:

third, the 3D Shape dataset: Control-based Program Learning

$$
\begin{aligned}
&\text{(Control Transition) } C &&: (\mathcal{O}_i, \mathcal{O}_O) \rightarrow C_i. \\
&\text{(Function Transition) } P &&: (\mathcal{O}_i, \mathcal{O}_O) \rightarrow f_i, \\
&\text{(Parameter Prediction) } Q_f &&: (\mathcal{O}_i, \mathcal{O}_O, C_i) \rightarrow \mathbf{\Theta}.
\end{aligned}
$$





Thanks