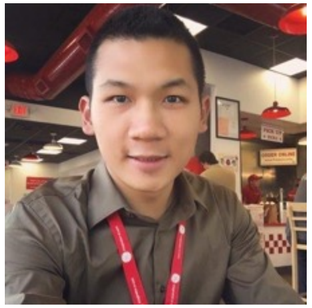
Neural Program Synthesis from Diverse Demonstration Videos



Shao-Hua Sun*[1]

Hyeonwoo Noh*[2]

Sriram Somasundaram[1]

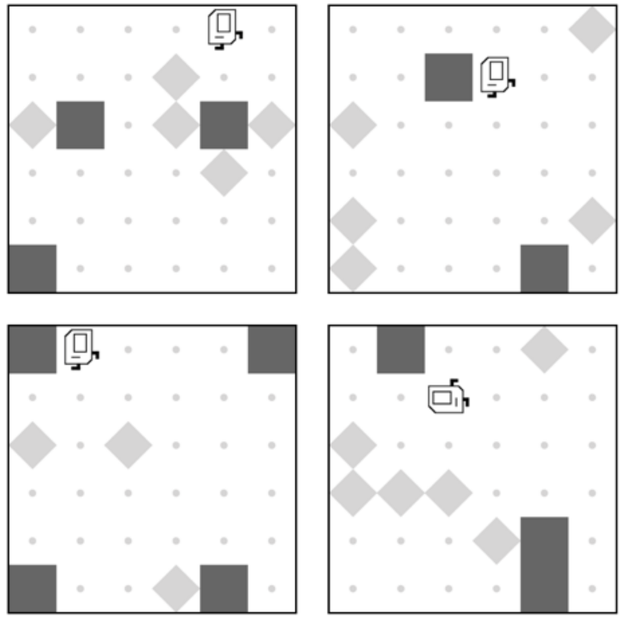
Joseph J. Lim[1]

[1]University of Southern California
[2]Pohang University of Science and Technology
*Equal contribution


youtube.com/croutoncrackerjacks

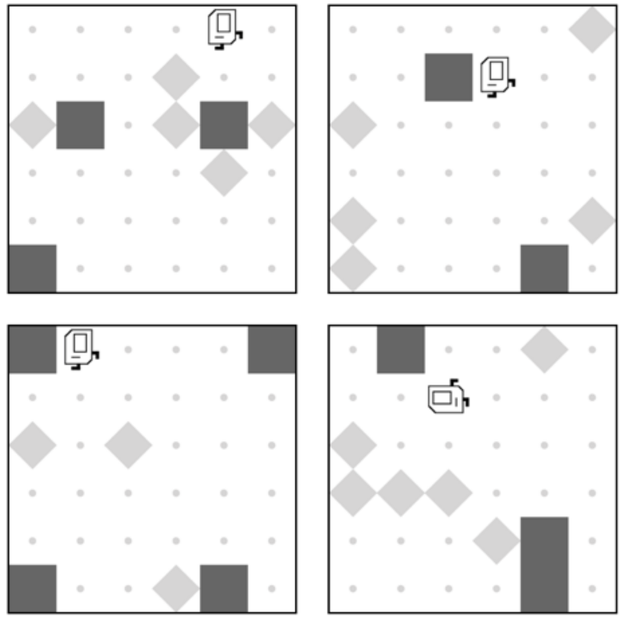

sesame oil
1 tbsp


Oil - 2 tbsp



```
def run()
  if frontIsClear():
    move()
  else:
    turnLeft()
  move()
  turnLeft()
  repeat(2):
    turnRight()
    putMarker()
```

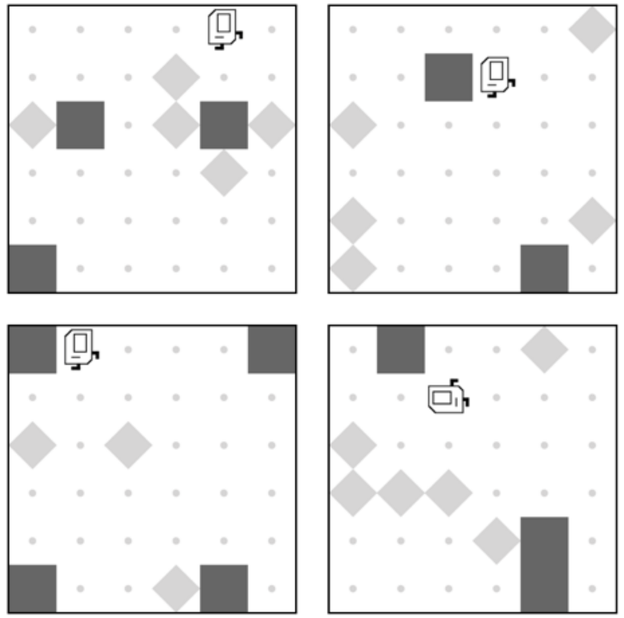
A **program** is an interpretable and executable way to describe behaviors



```
def run()
  if frontIsClear():
    move()
  else:
    turnLeft()
  move()
  turnLeft()
  repeat(2):
    turnRight()
    putMarker()
```

Human

A **program** is an interpretable and executable way to describe behaviors



```
def run()
  if frontIsClear():
    move()
  else:
    turnLeft()
  move()
  turnLeft()
  repeat(2):
    turnRight()
    putMarker()
```

Human **Robot**

A **program** is an interpretable and executable way to describe behaviors

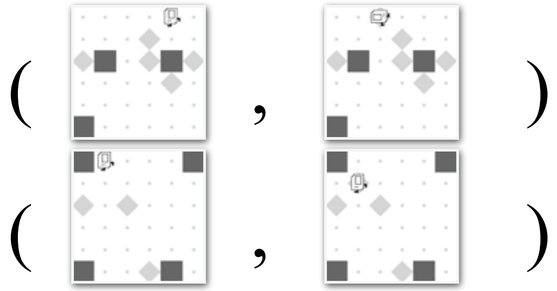
Program Synthesis

```
public class TcpClientSample
{
    public static void Main()
    {
        byte[] data = new byte[1024]; string input, stringData;
        TcpClient server;
        try{
            server = new TcpClient(" . . . . ", port);
        }catch (SocketException){
            Console.WriteLine("Unable to connect to server")
            return;
        }
        NetworkStream ns = server.GetStream();
        int recv = ns.Read(data, 0, data.Length);
        stringData  = Encoding.
            ASCII.GetString(data, 0, recv);
        Console.WriteLine(stringData);
        while(true){
            input = Console.ReadLine();
            if (input == "exit") break;
                newchild.Properties["ou"].Add
                ("Auditing Department");
                newchild.CommitChanges();
                newchild.Close();
```

Program Synthesis

Task specification

Input/output pairs



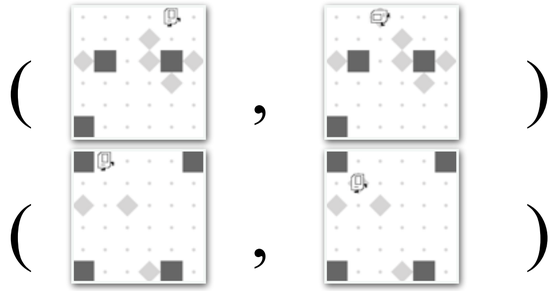
Devlin et al. “*Robustfill: Neural program learning under noisy i/o*.” ICML 2017
Balog, et al. "Deepcoder: Learning to write programs." ICLR 2017
Rudy R et al. “*Leveraging grammar and reinforcement learning for neural program synthesis*.” ICLR 2018

Program Synthesis

Task specification

Program

Input/output pairs



```
def run()
   if frontIsClear():
      move()
   else:
      turnLeft()
   move()
   turnLeft
   repeat(2):
      turnRight()
      putMarker()
```

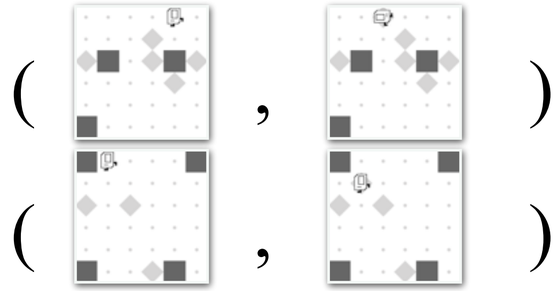

Devlin et al. “*Robustfill: Neural program learning under noisy i/o.*” ICML 2017
Balog, et al. "Deepcoder: Learning to write programs." ICLR 2017
Rudy R et al. “*Leveraging grammar and reinforcement learning for neural program synthesis.*” ICLR 2018

Program Synthesis

Task specification

Program

Input/output pairs



```
def run()
   if frontIsClear():
      move()
   else:
      turnLeft()
   move()
   turnLeft
   repeat(2):
      turnRight()
      putMarker()
```

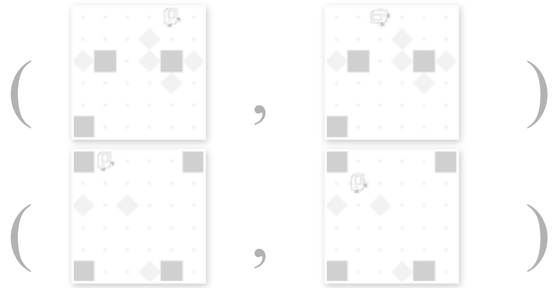

Devlin et al. “*Robustfill: Neural program learning under noisy i/o.*” ICML 2017
Balog, et al. "Deepcoder: Learning to write programs." ICLR 2017
Rudy R et al. “*Leveraging grammar and reinforcement learning for neural program synthesis.*” ICLR 2018

Program Synthesis

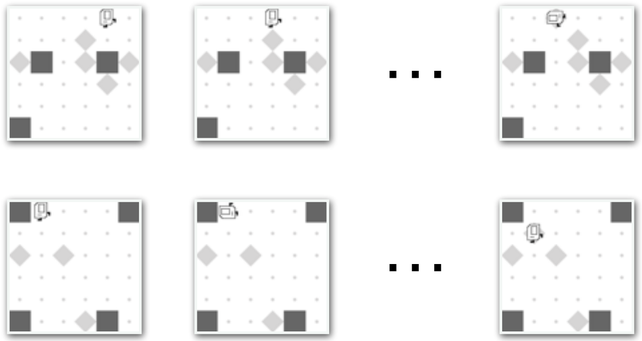
Task specification

Program

Input/output pairs



Demonstration Sequences

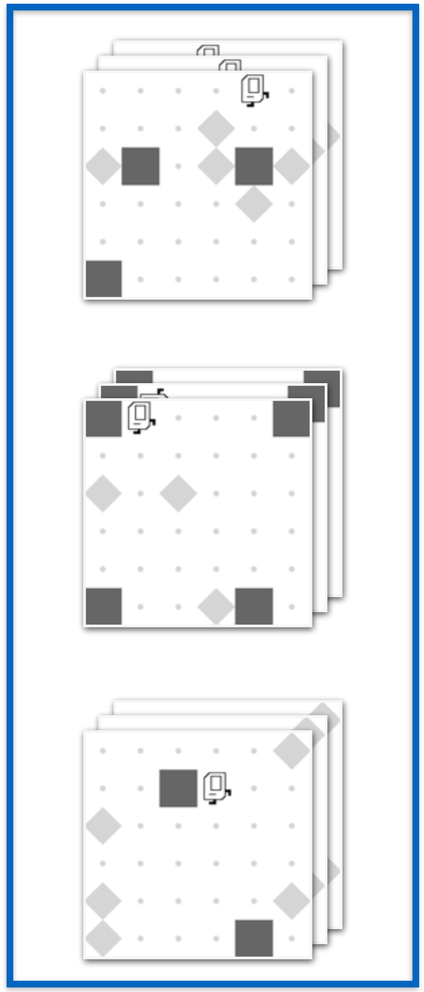


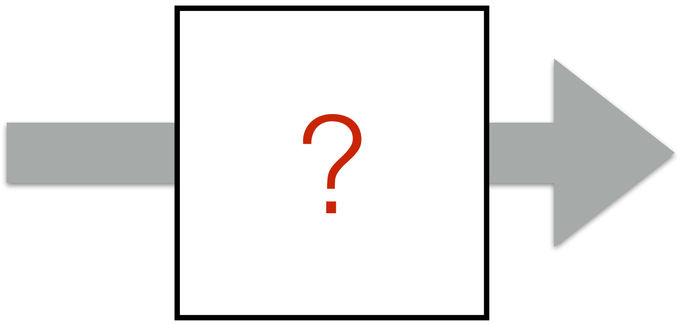
```
def run()
    if frontIsClear():
        move()
    else:
        turnLeft()
    move()
    turnLeft
    repeat(2):
        turnRight()
        putMarker()
```

Problem Formulation

Input

a set of demo videos

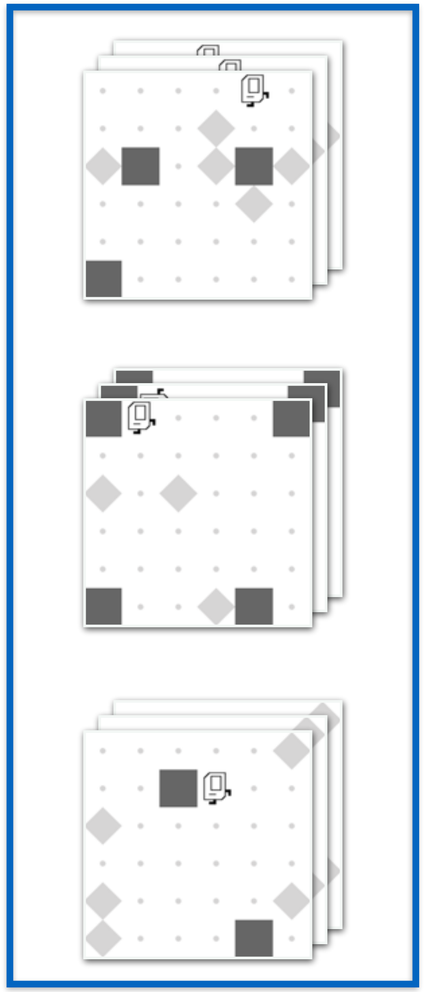


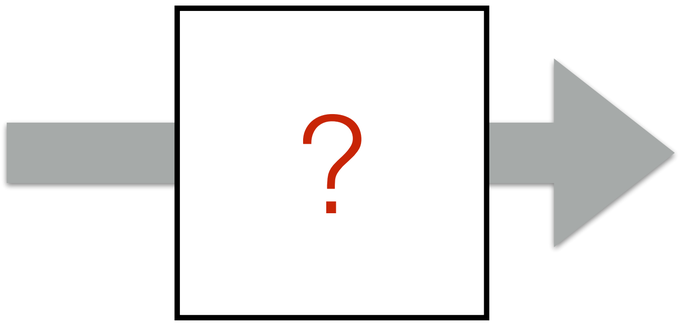


Problem Formulation

Input

a set of demo videos



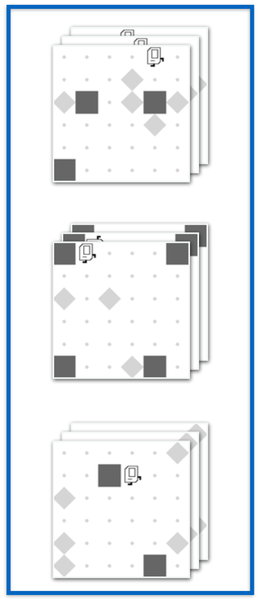


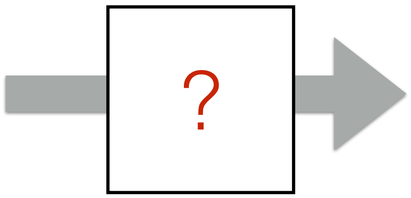
Output

a program describing the demonstrated behavior

Problem Formulation

Input
a set of demo videos





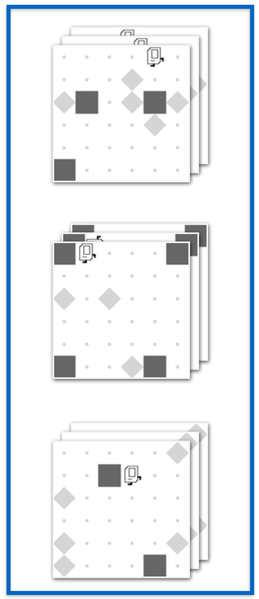
Output
a program describing the demonstrated behavior

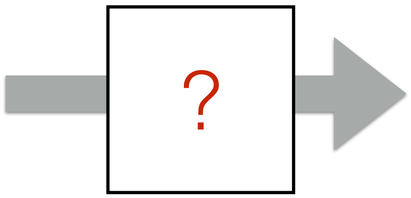
Challenges

- **Extracting unique behaviors** in each demo
- **Summarizing** diverse behaviors as a program

Problem Formulation

Input
a set of demo videos





Output
a program describing the demonstrated behavior

Challenges

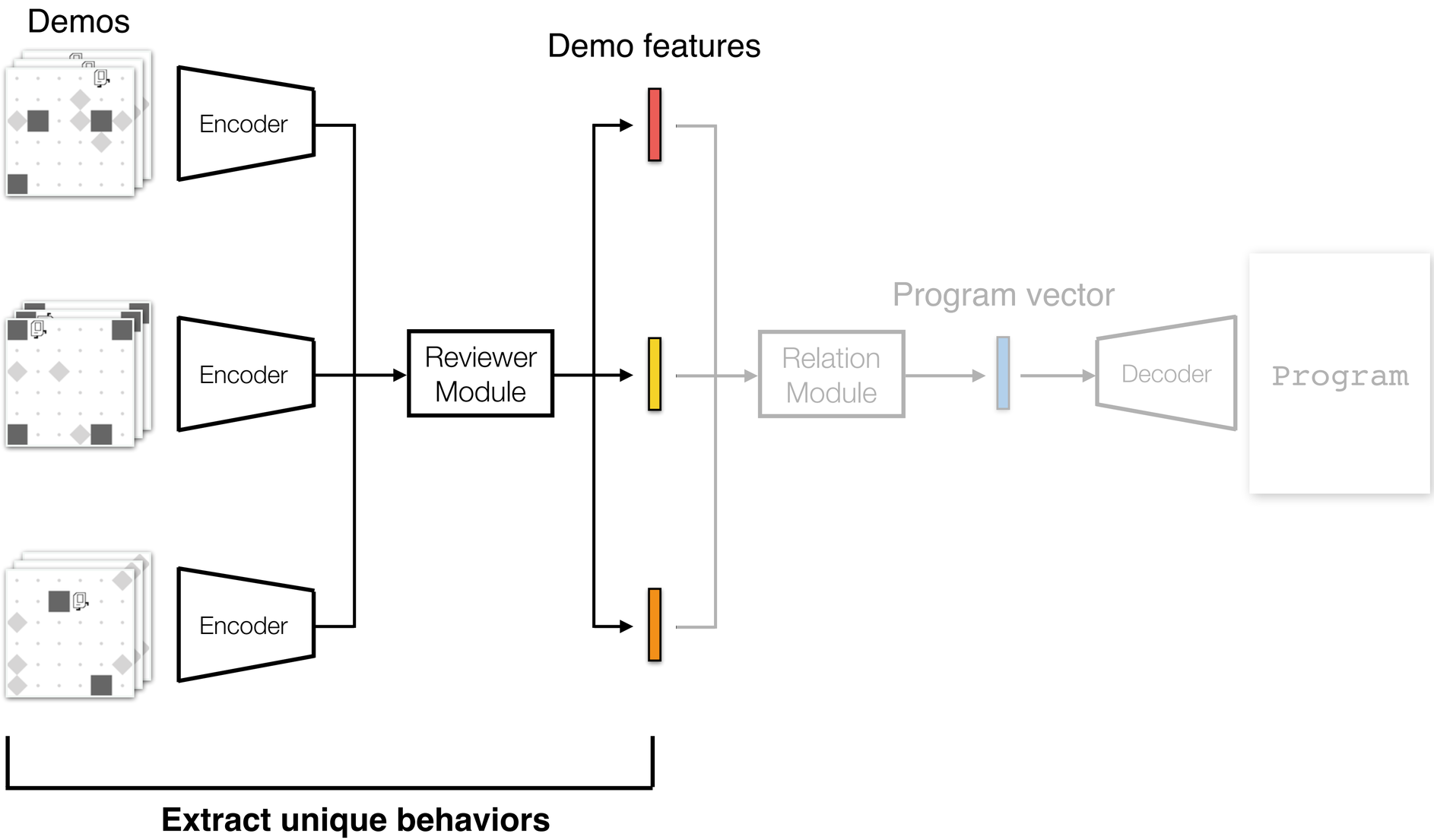
- **Extracting unique behaviors** in each demo

  Reviewer module

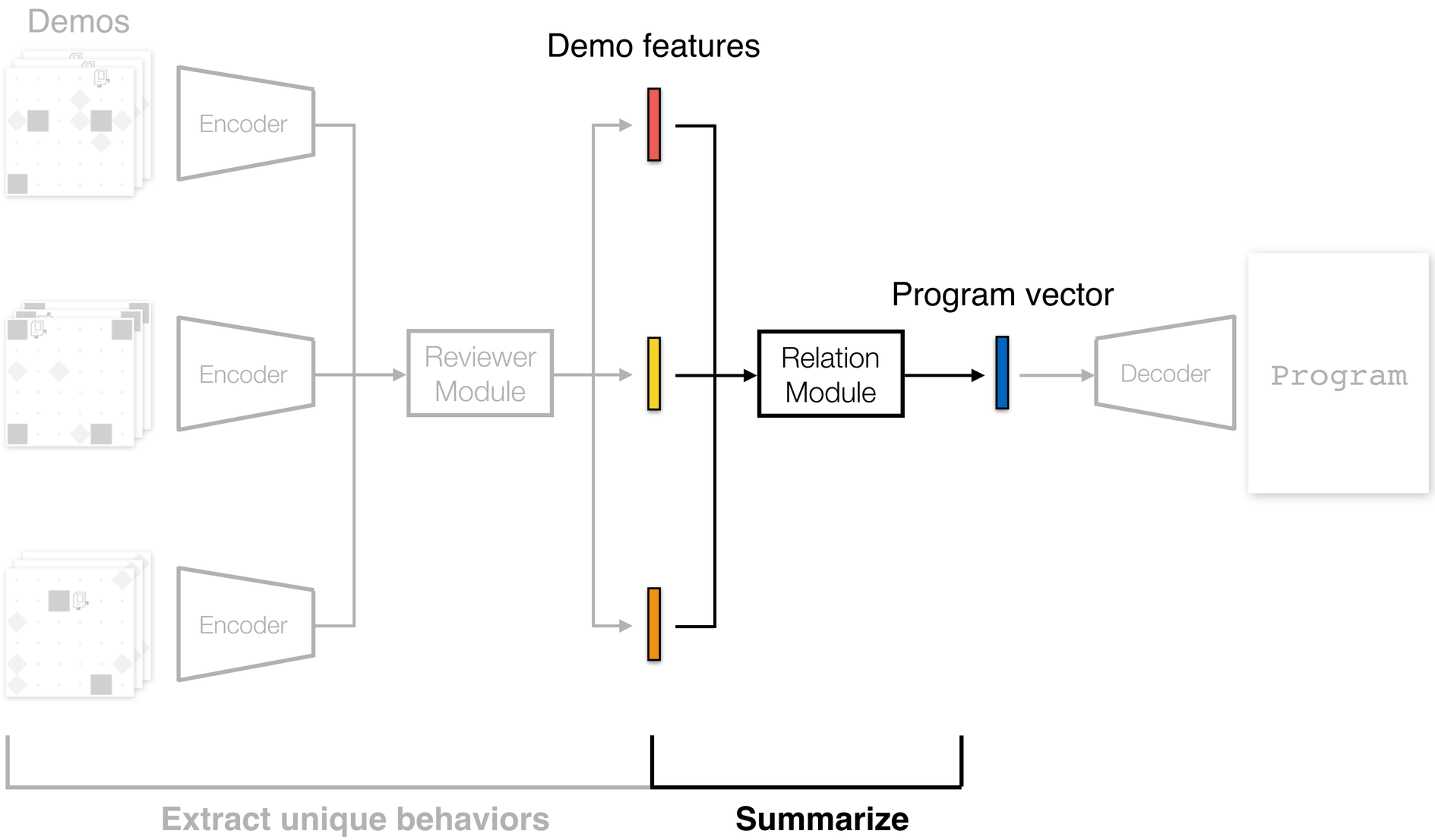
- **Summarizing** diverse behaviors as a program

  Relation module

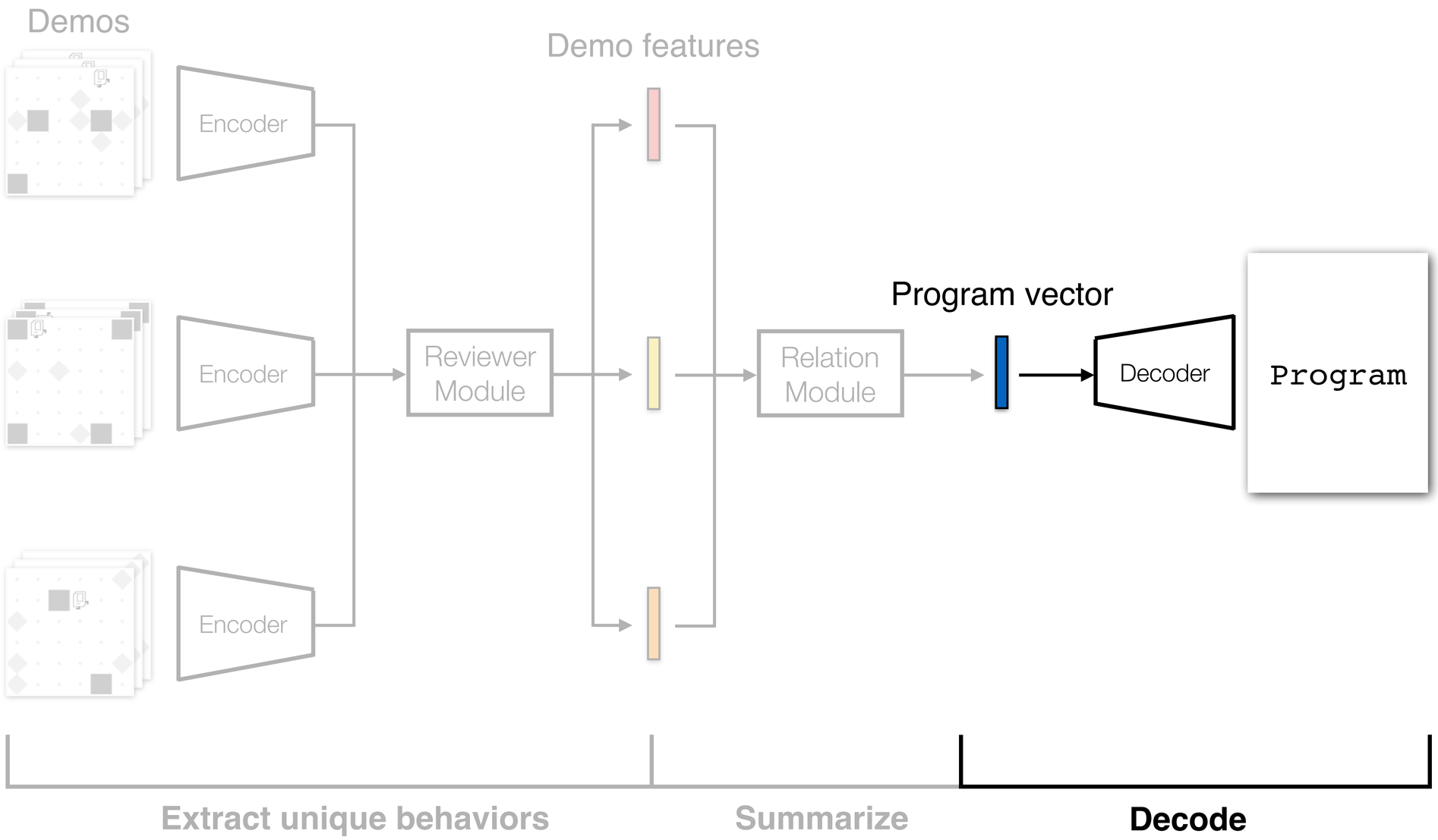
Model Overview



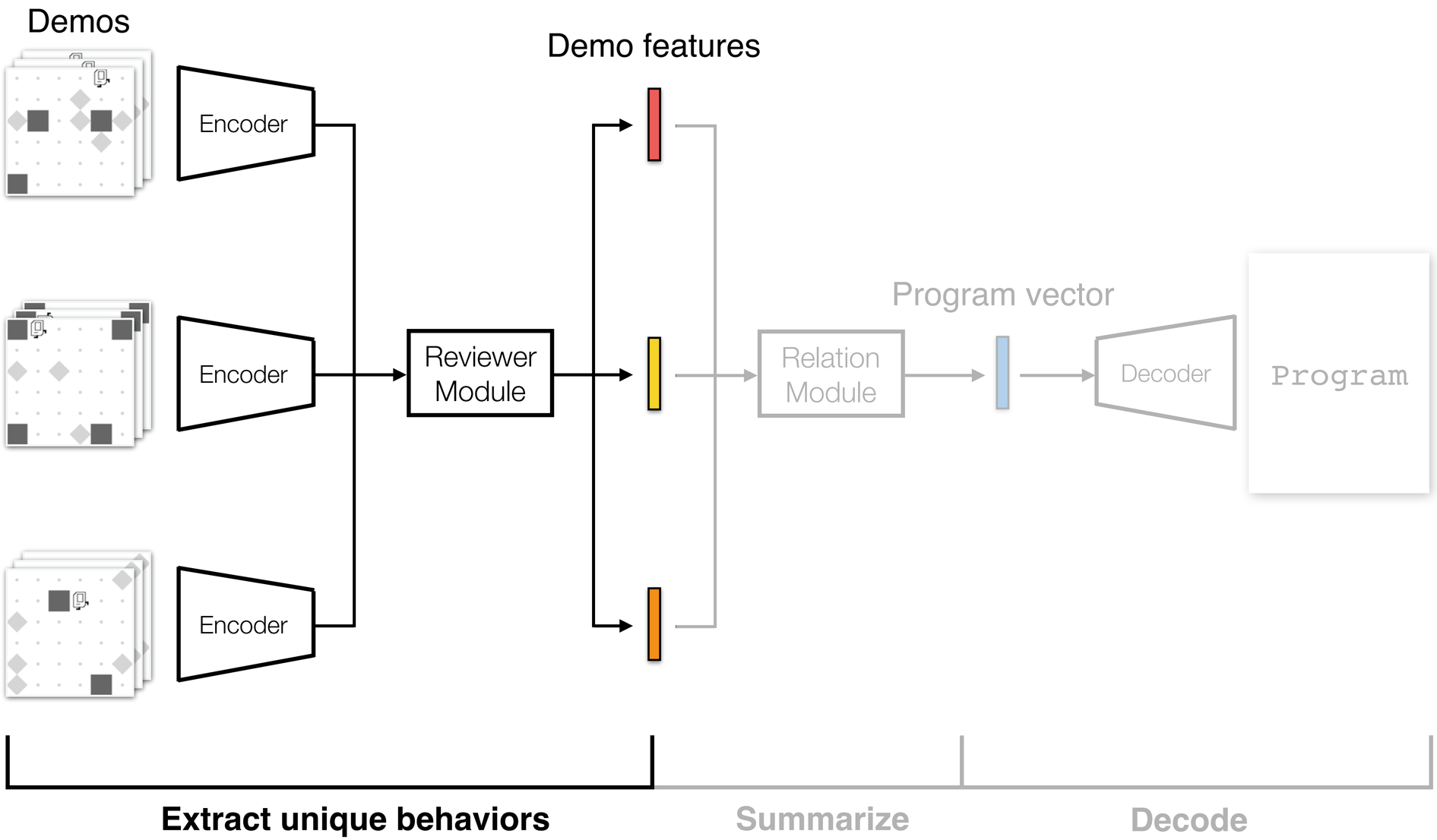
Model Overview



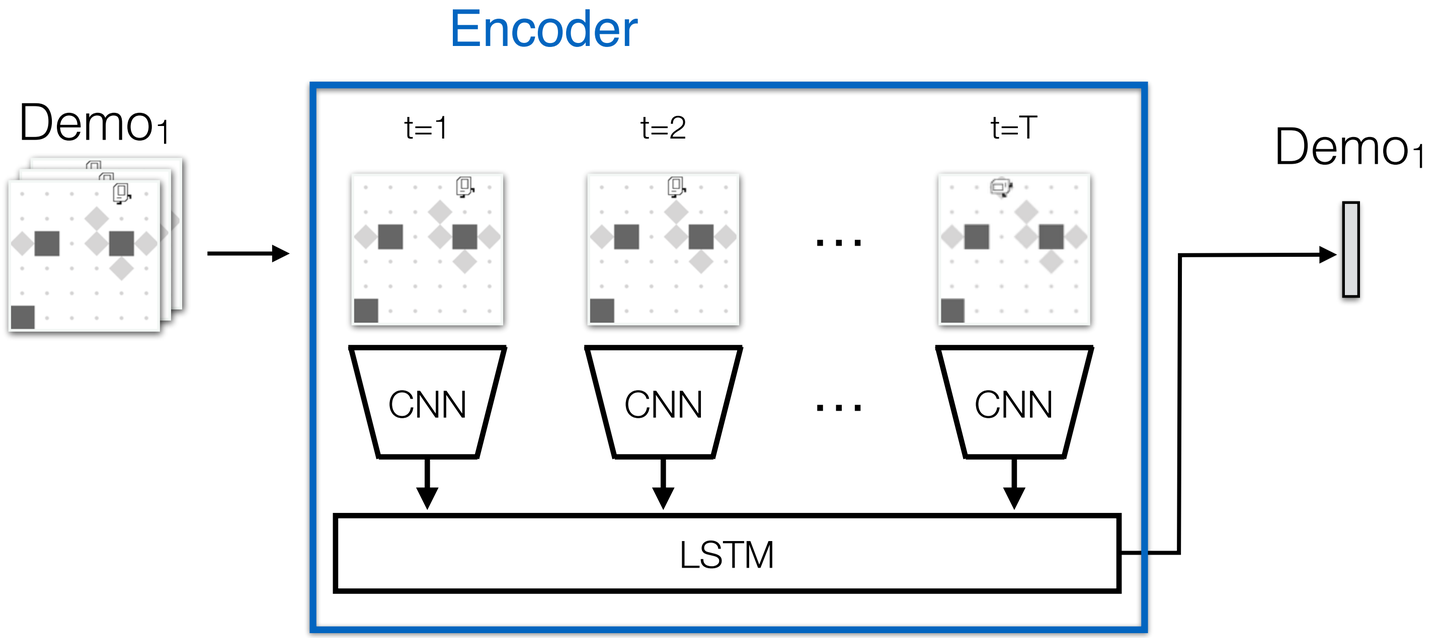
Model Overview



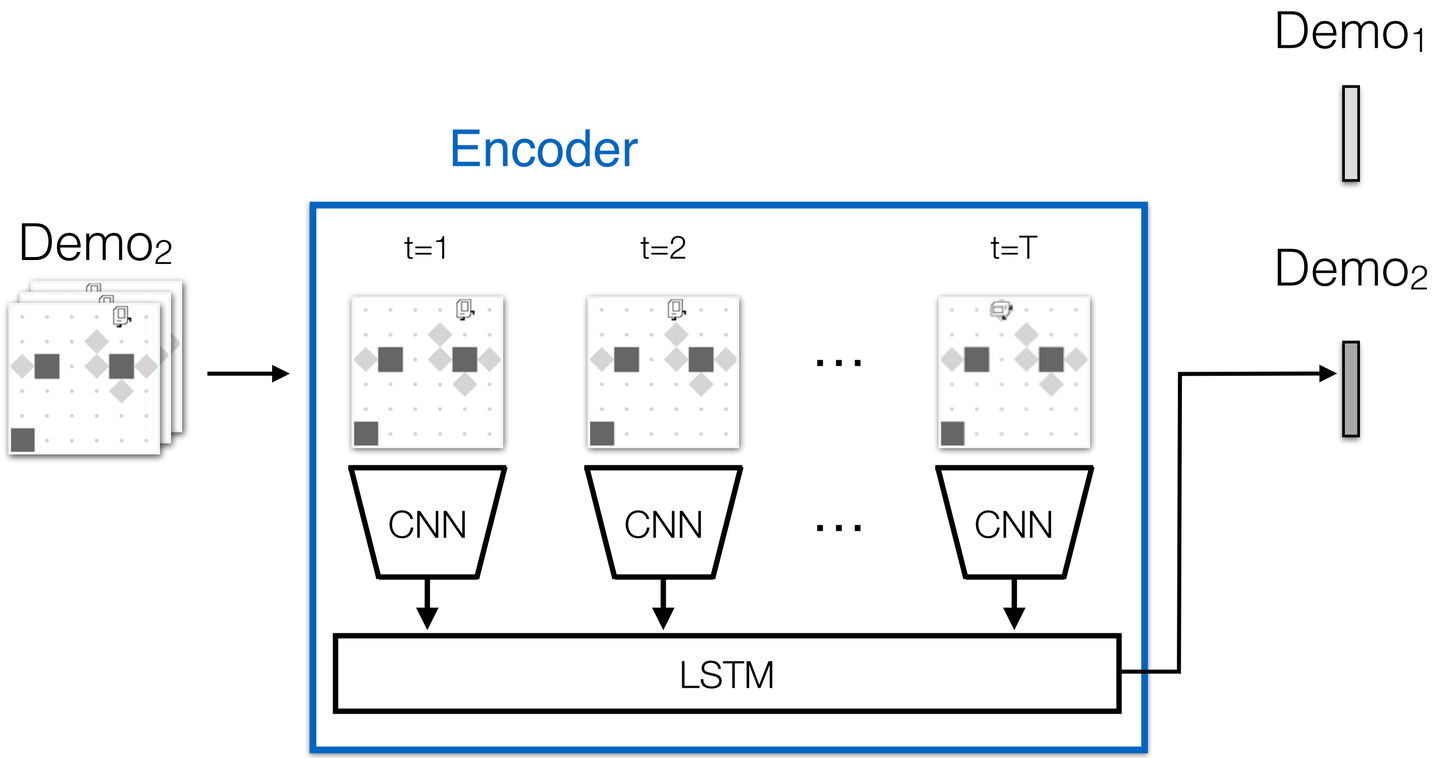
Model Overview



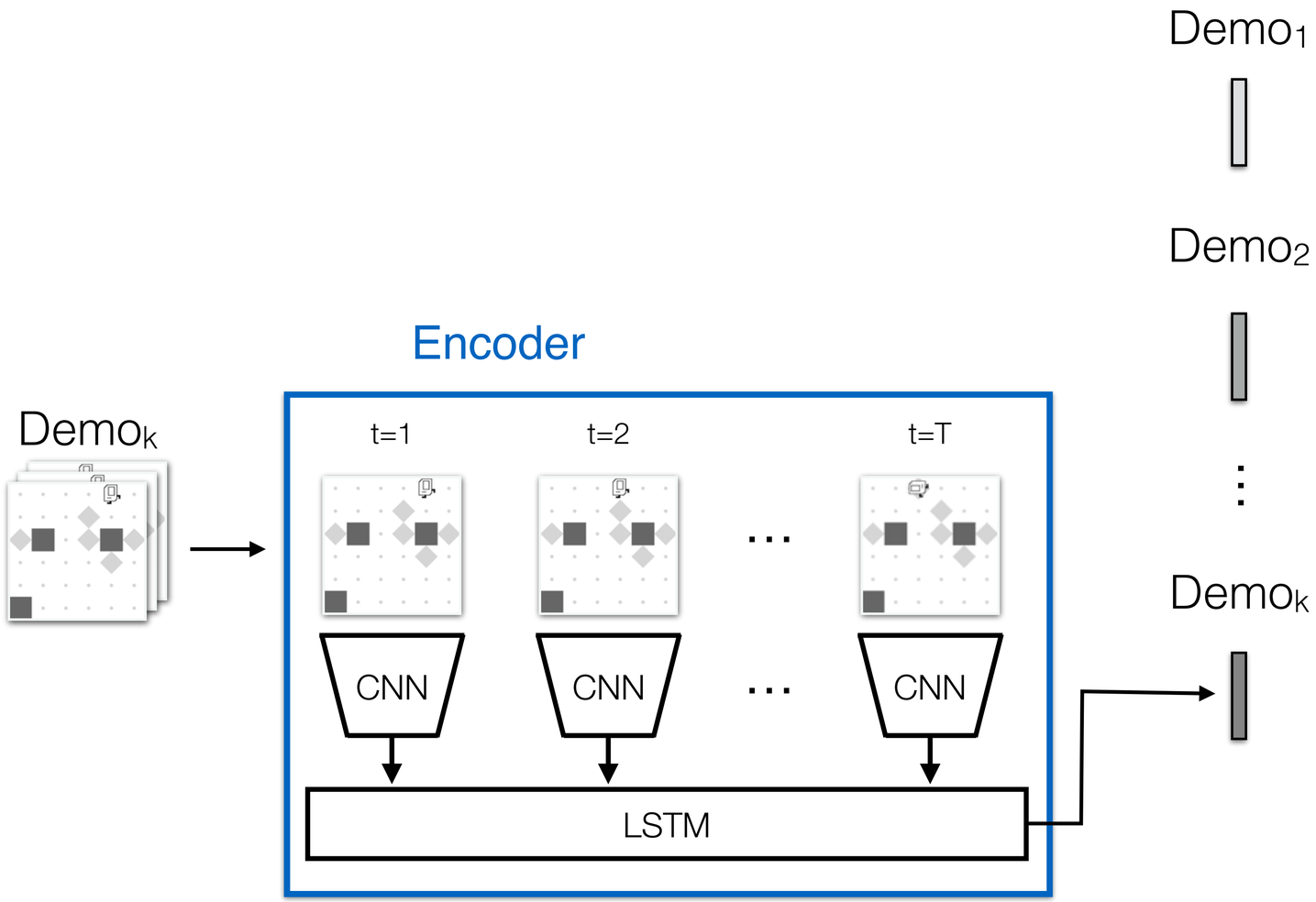
Reviewer Module



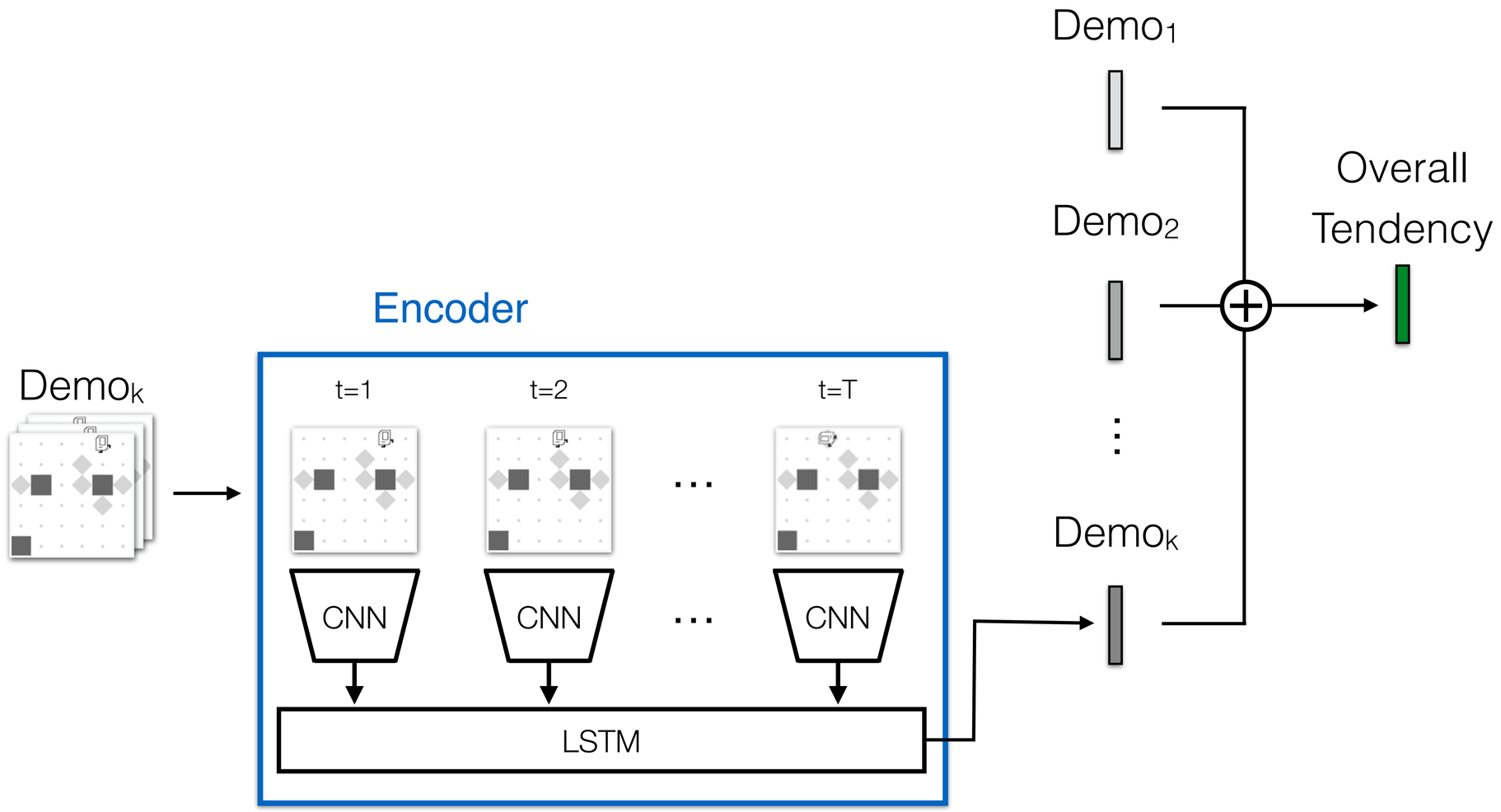
Reviewer Module



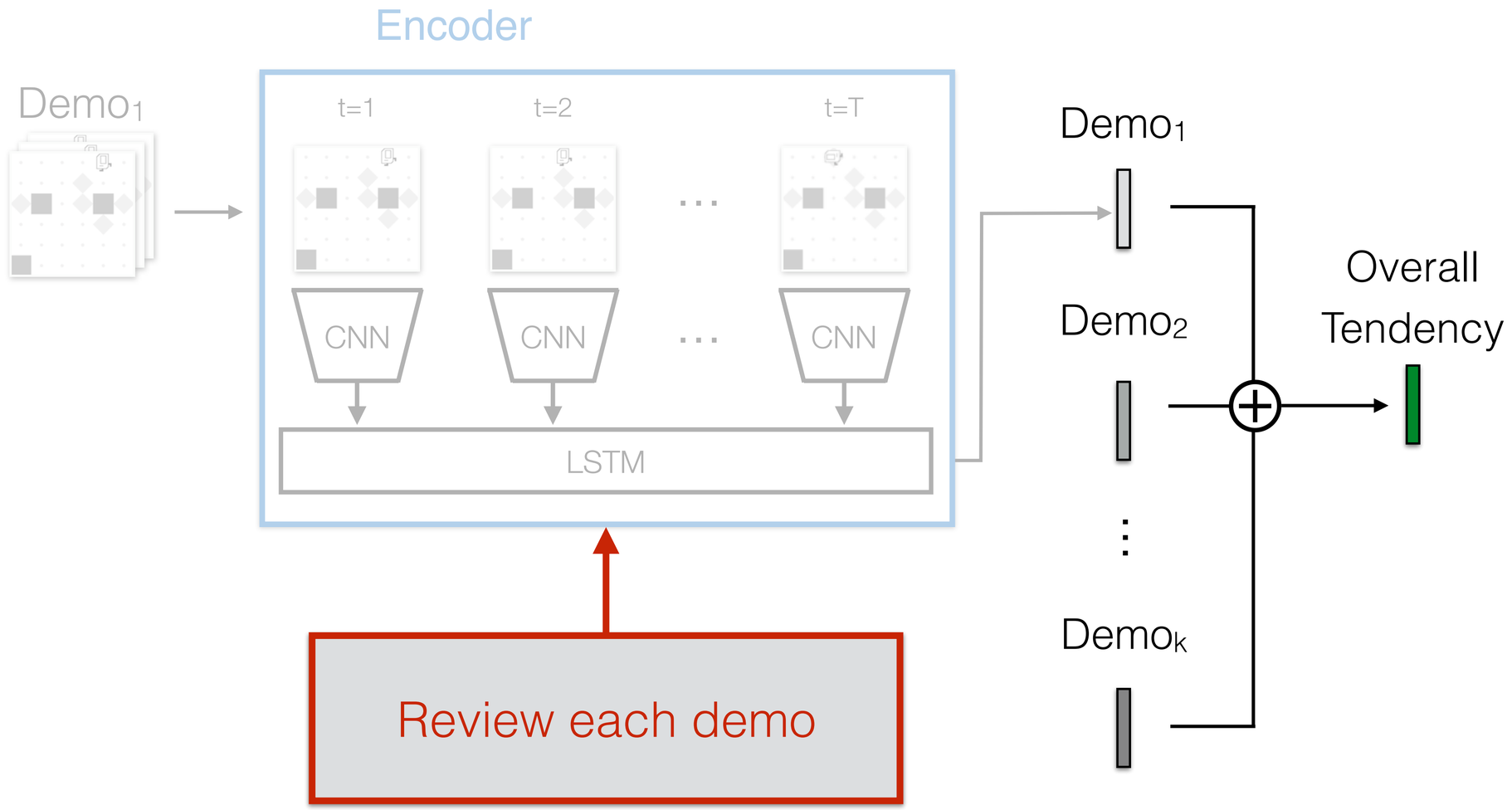
Reviewer Module



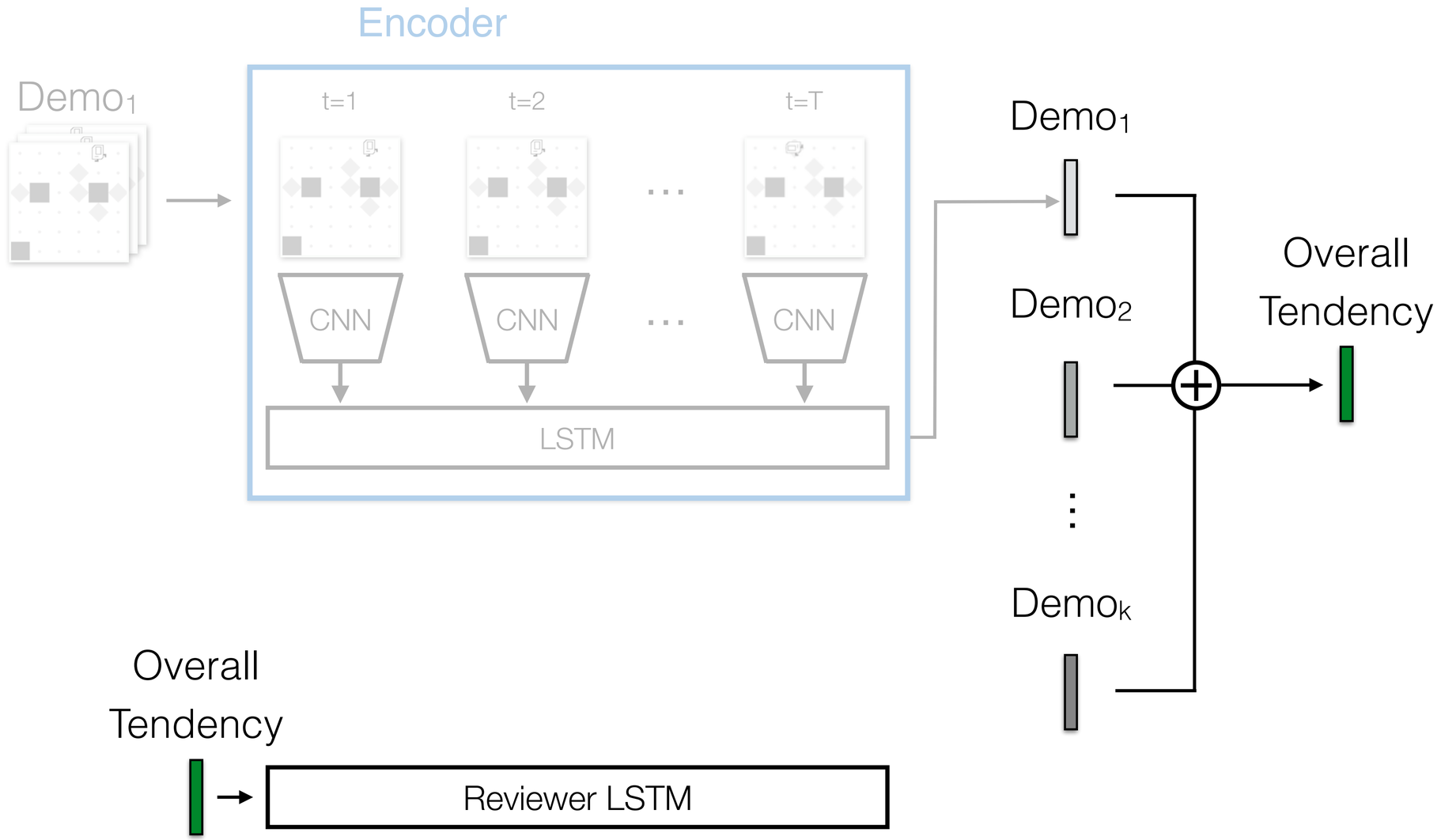
Reviewer Module



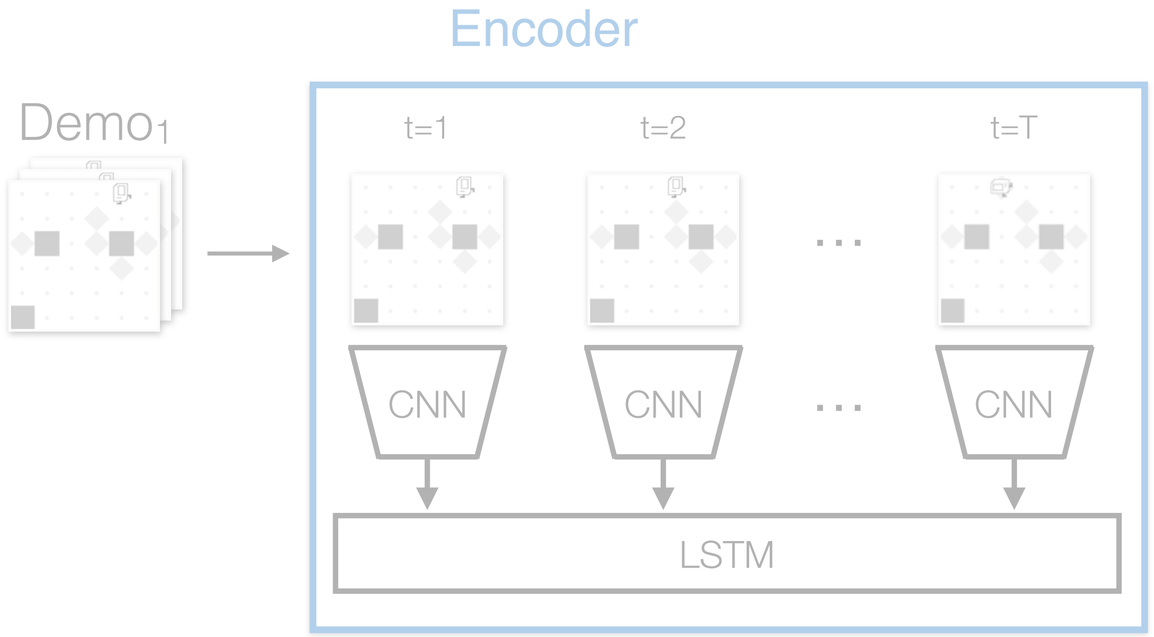
Reviewer Module



Reviewer Module

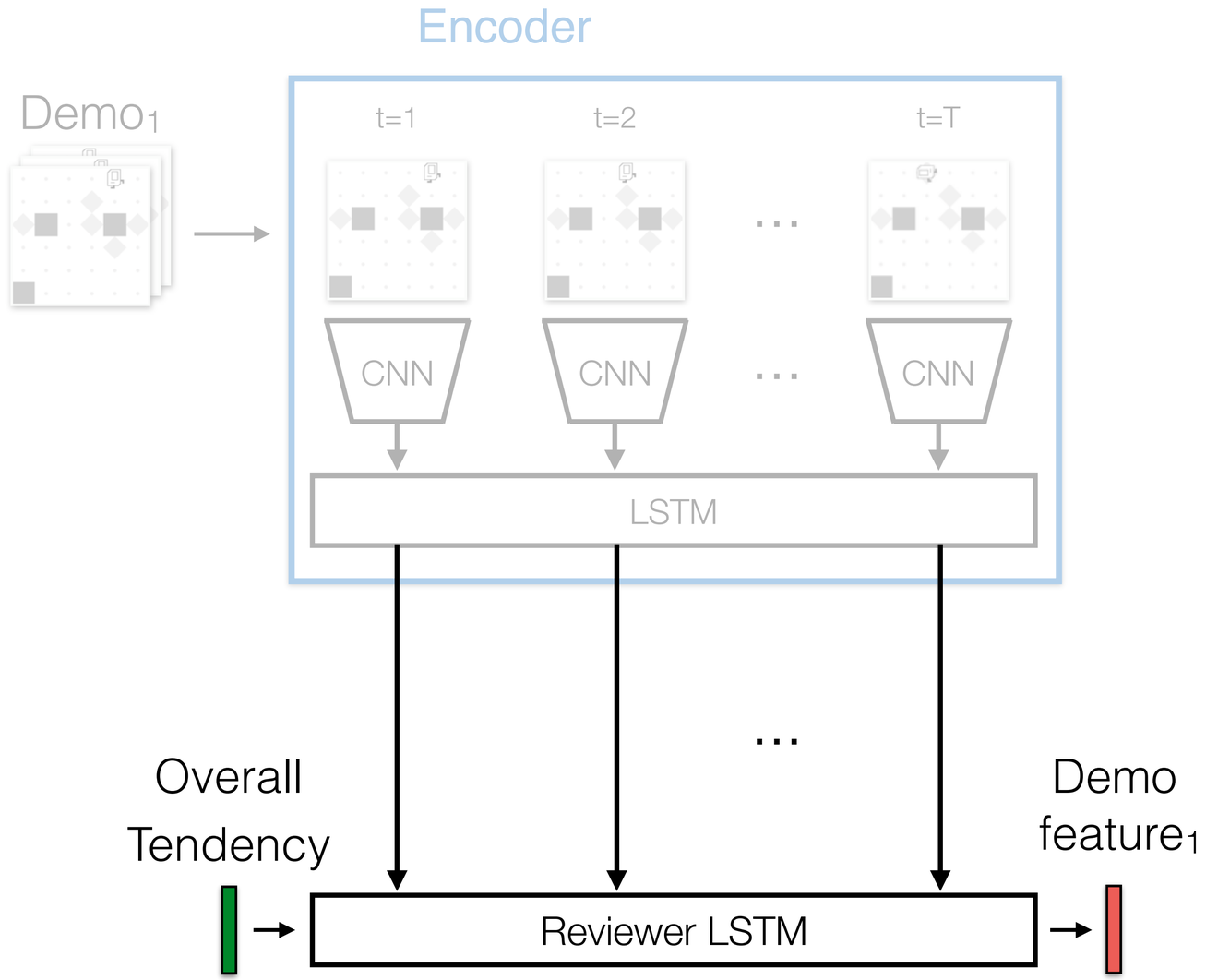


Reviewer Module

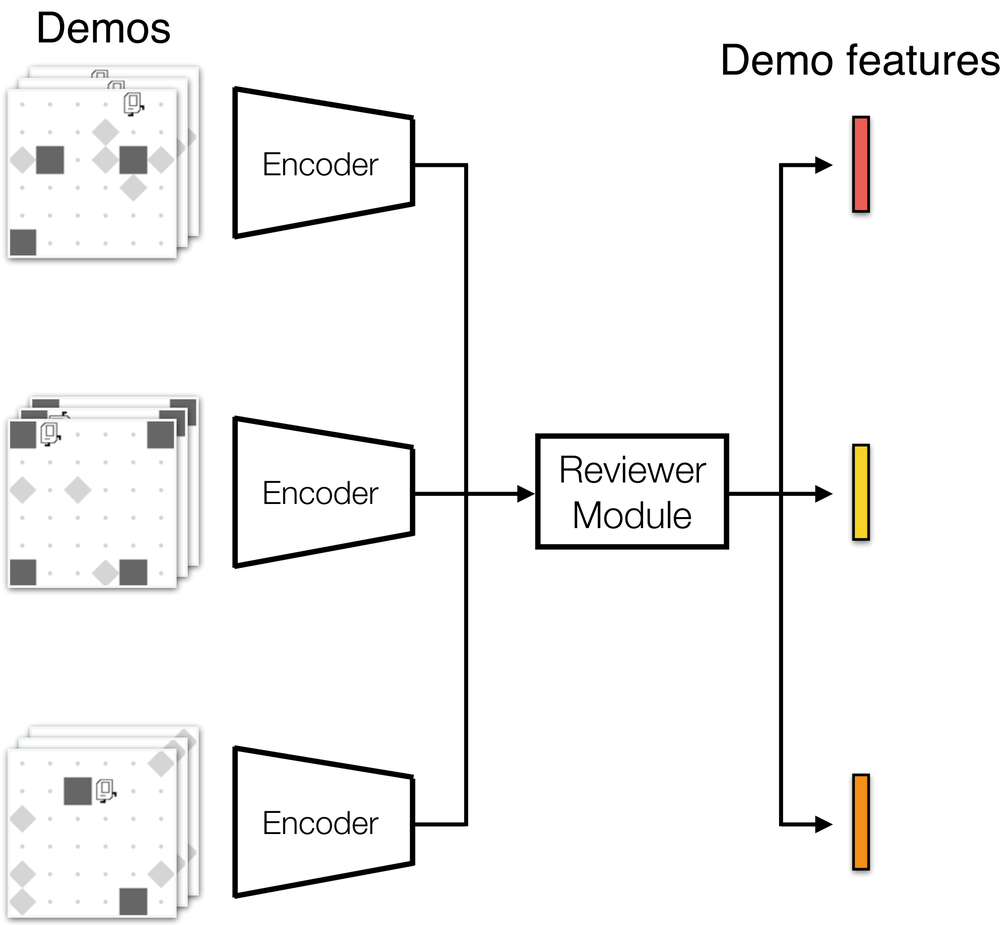


Overall Tendency

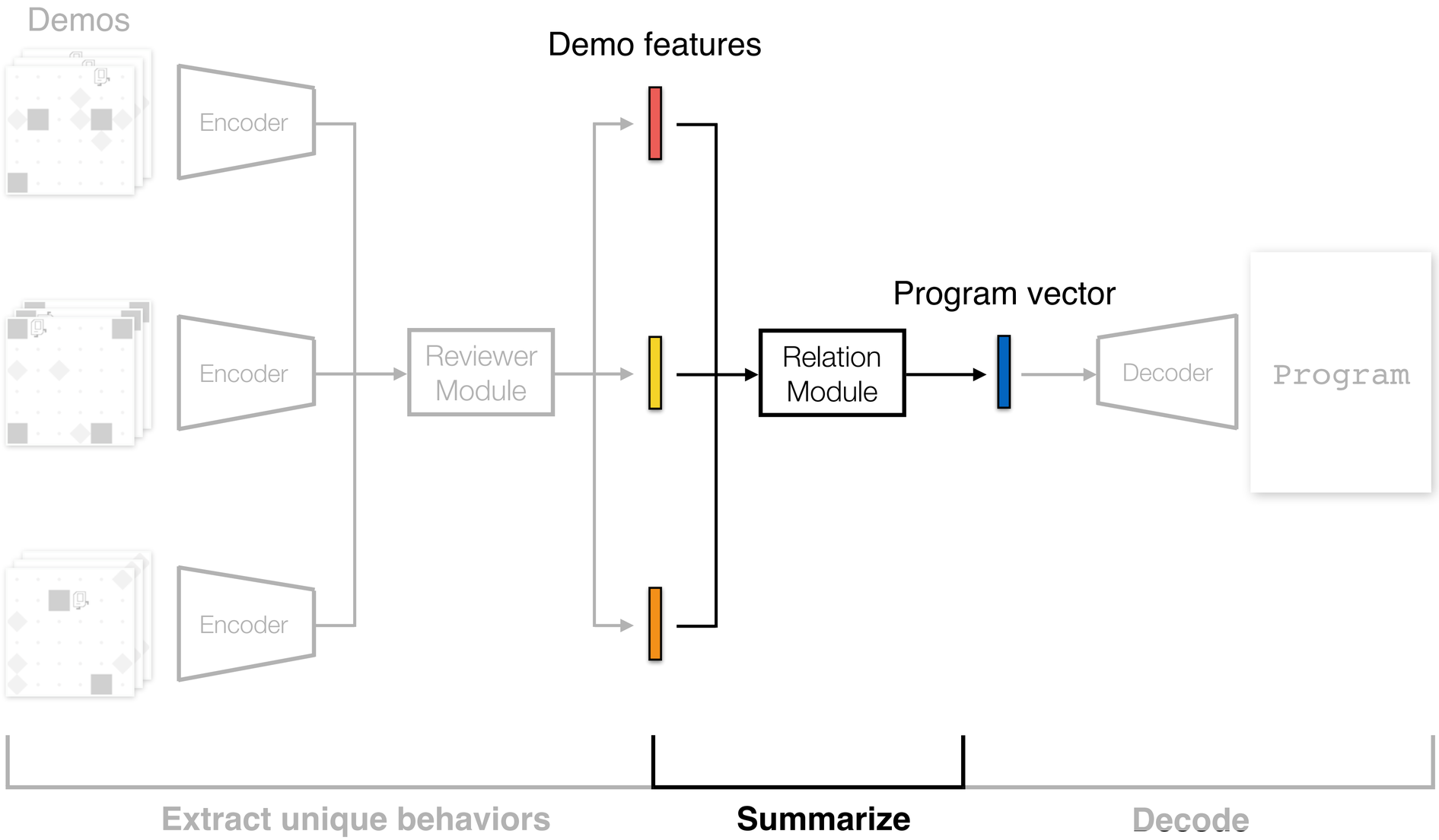
Reviewer Module



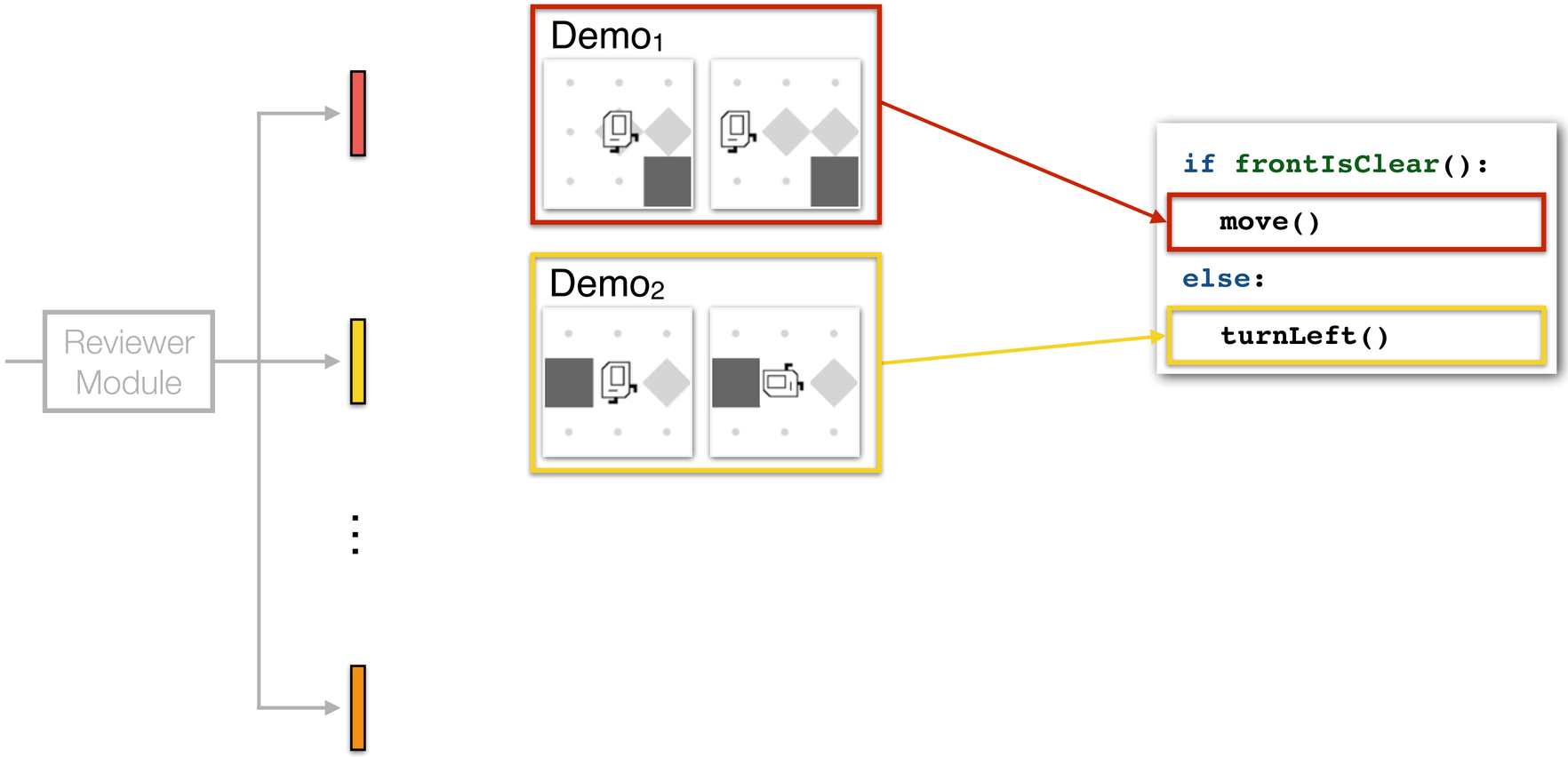
Model Overview



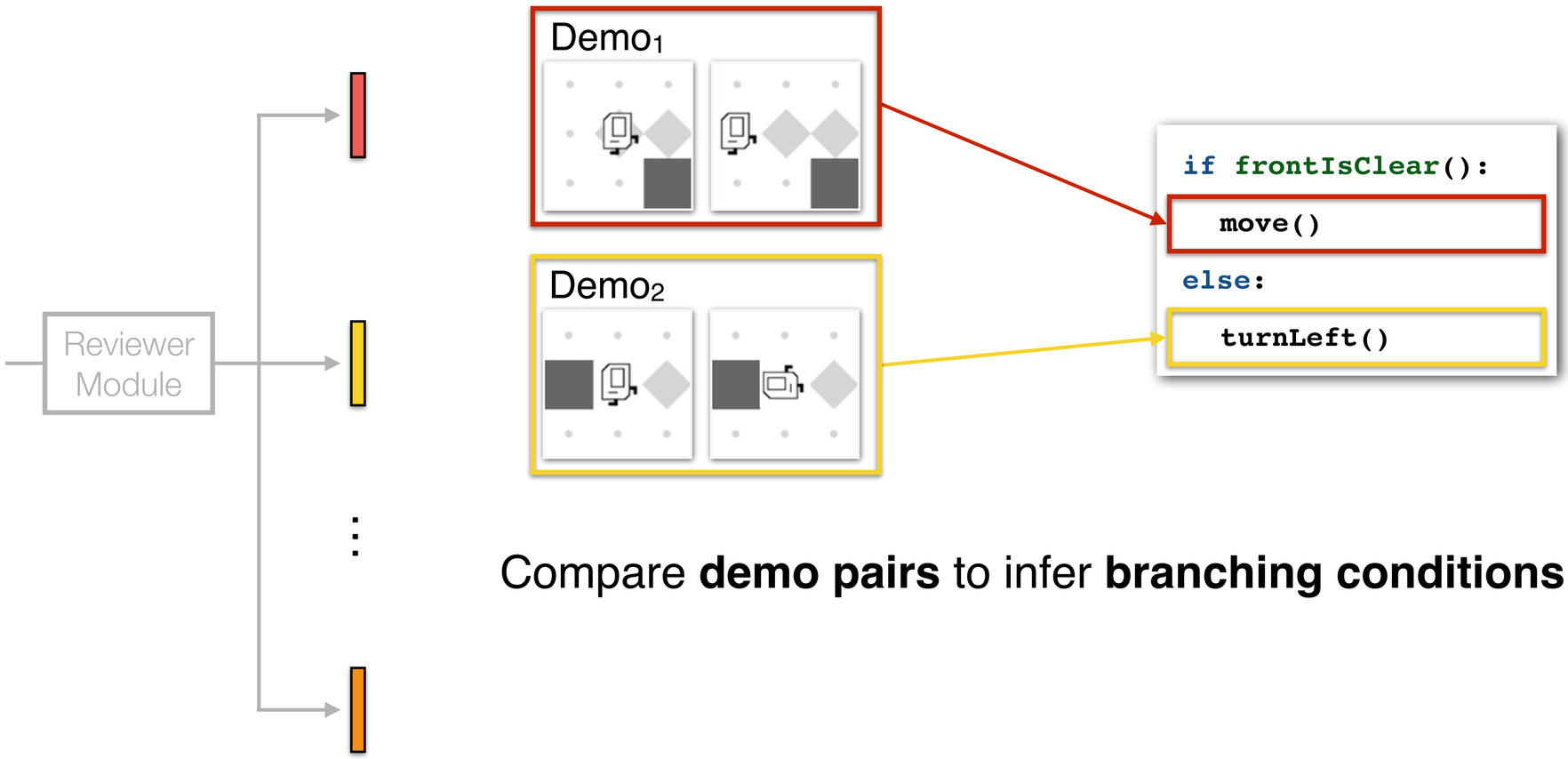
Model Overview



Relation Module

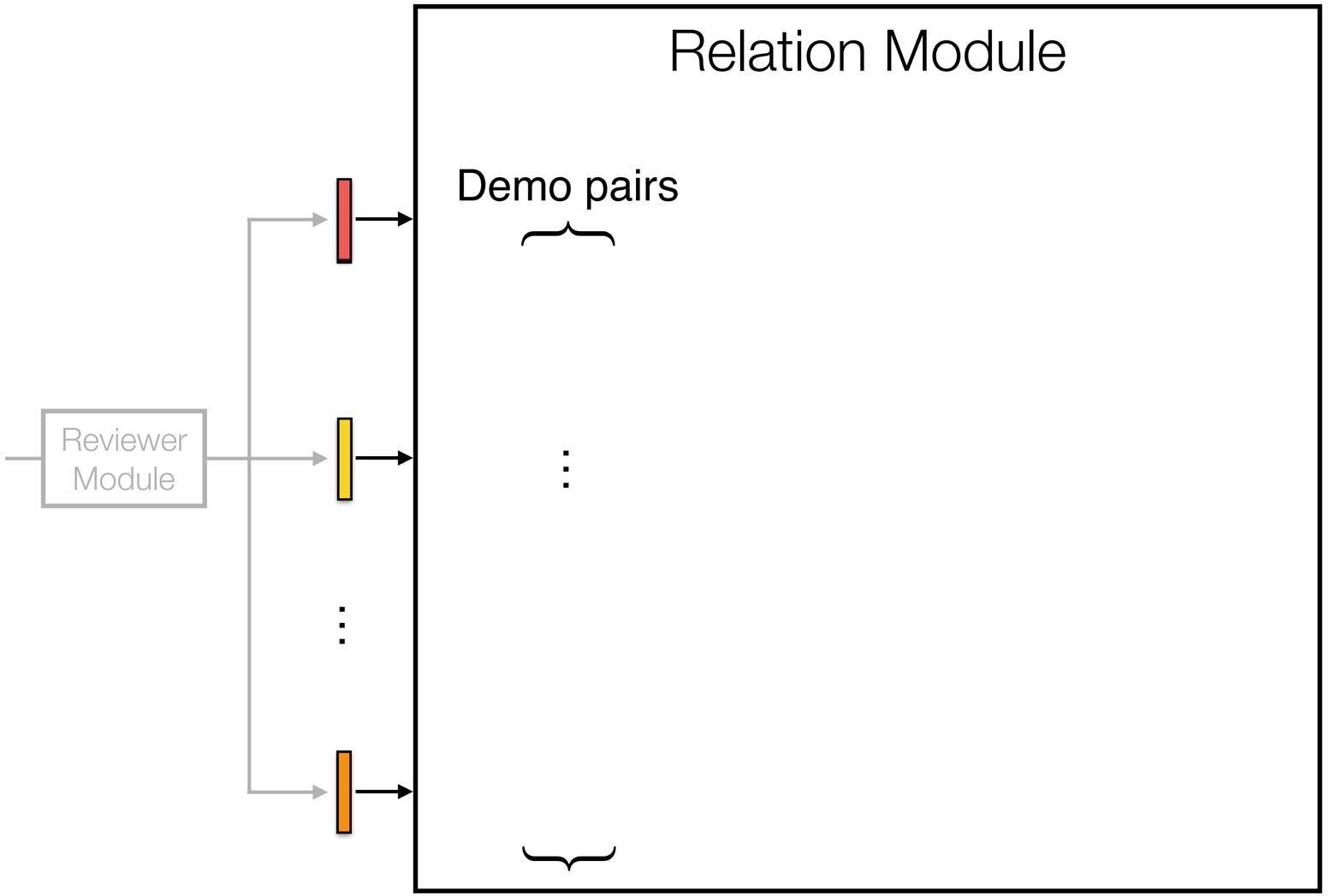


Relation Module



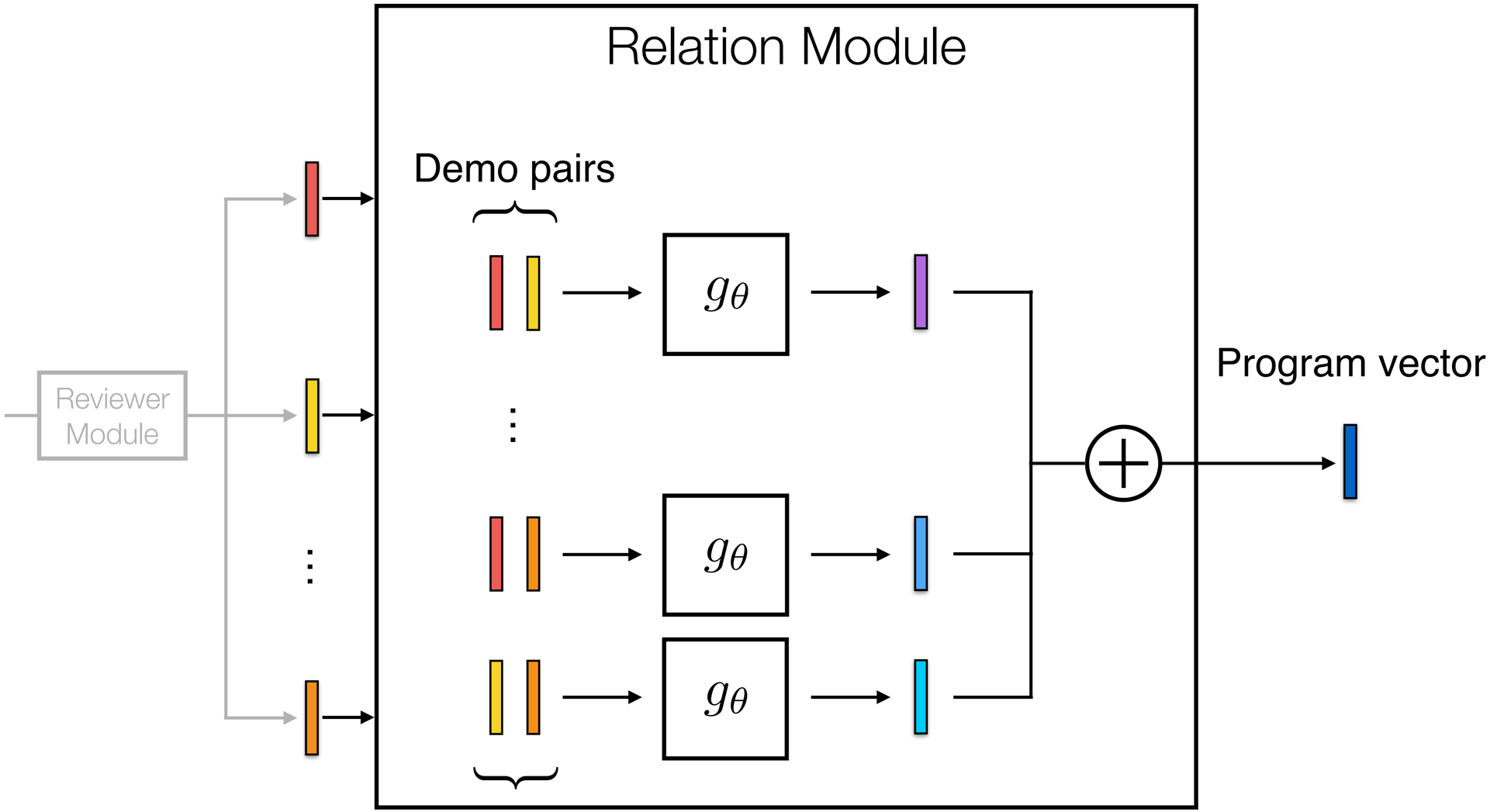
Compare **demo pairs** to infer **branching conditions**

Relation Module



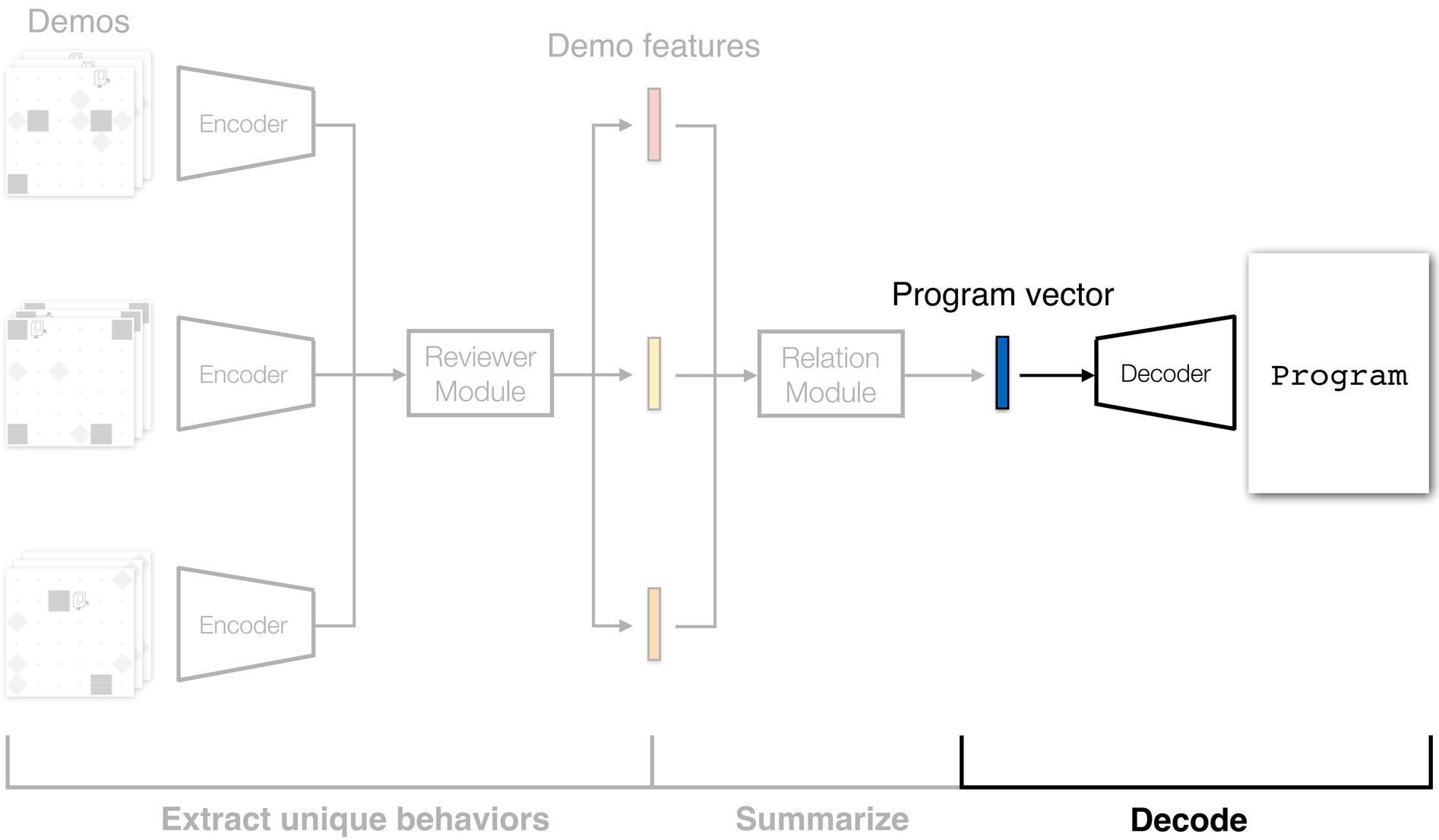
Santoro et al. "A simple neural network module for relational reasoning." NIPS 2017

Relation Module

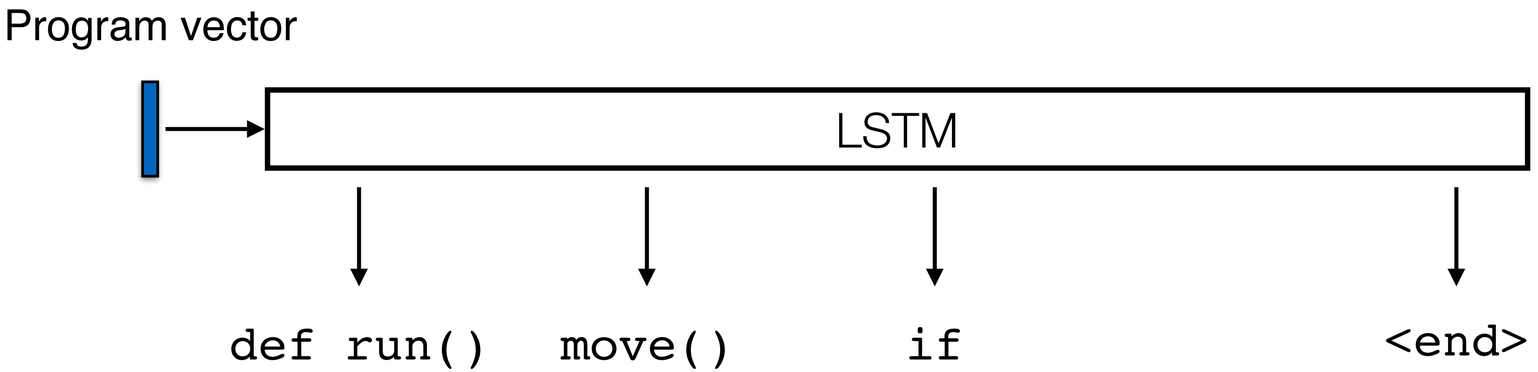


Santoro et al. "A simple neural network module for relational reasoning." NIPS 2017

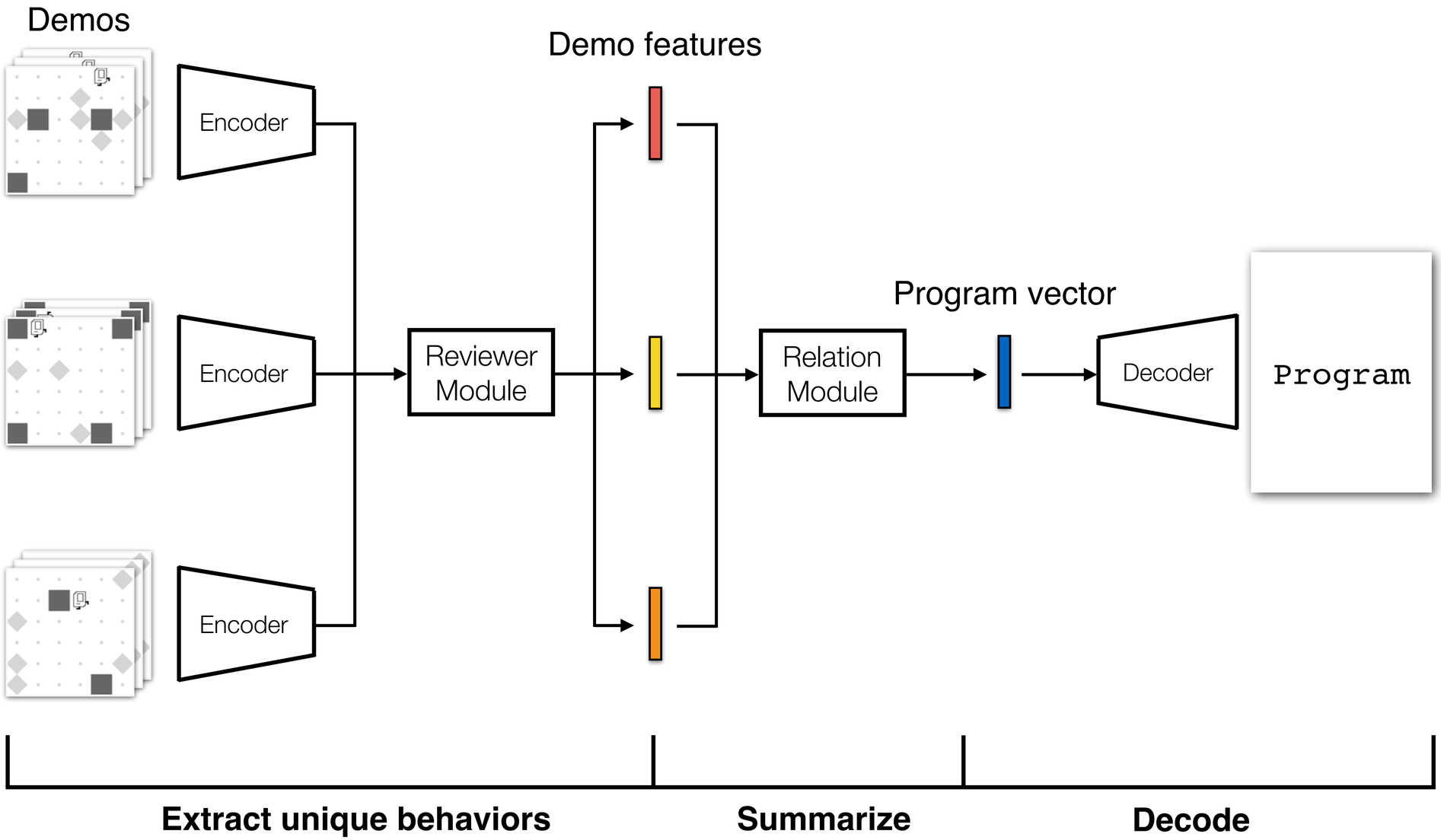
Model Overview



Decode a Program



Model Overview

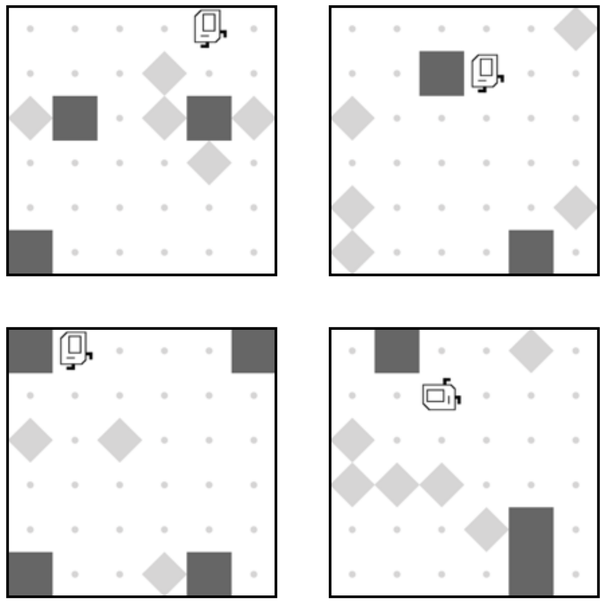


Experiments

Environments

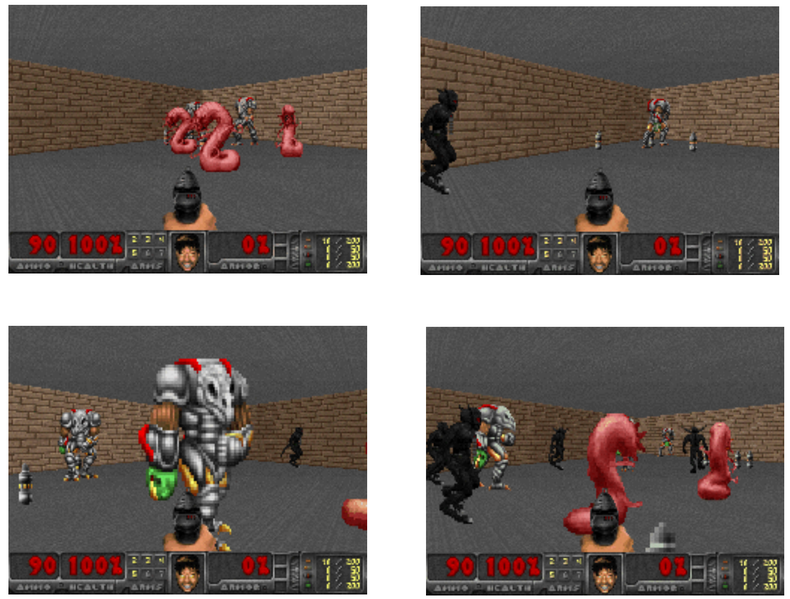
Karel

```
def run()
    if frontIsClear():
        move()
    else:
        turnLeft()
    move()
    turnLeft
    repeat(2):
        turnRight()
        putMarker()
```



ViZDoom

```
def run()
    while(inTarget(
            HellKnight)):
        attack()
        moveForward()
    if isThere(Demon):
        moveRight()
    else:
        moveLeft()
        moveBackward()
```

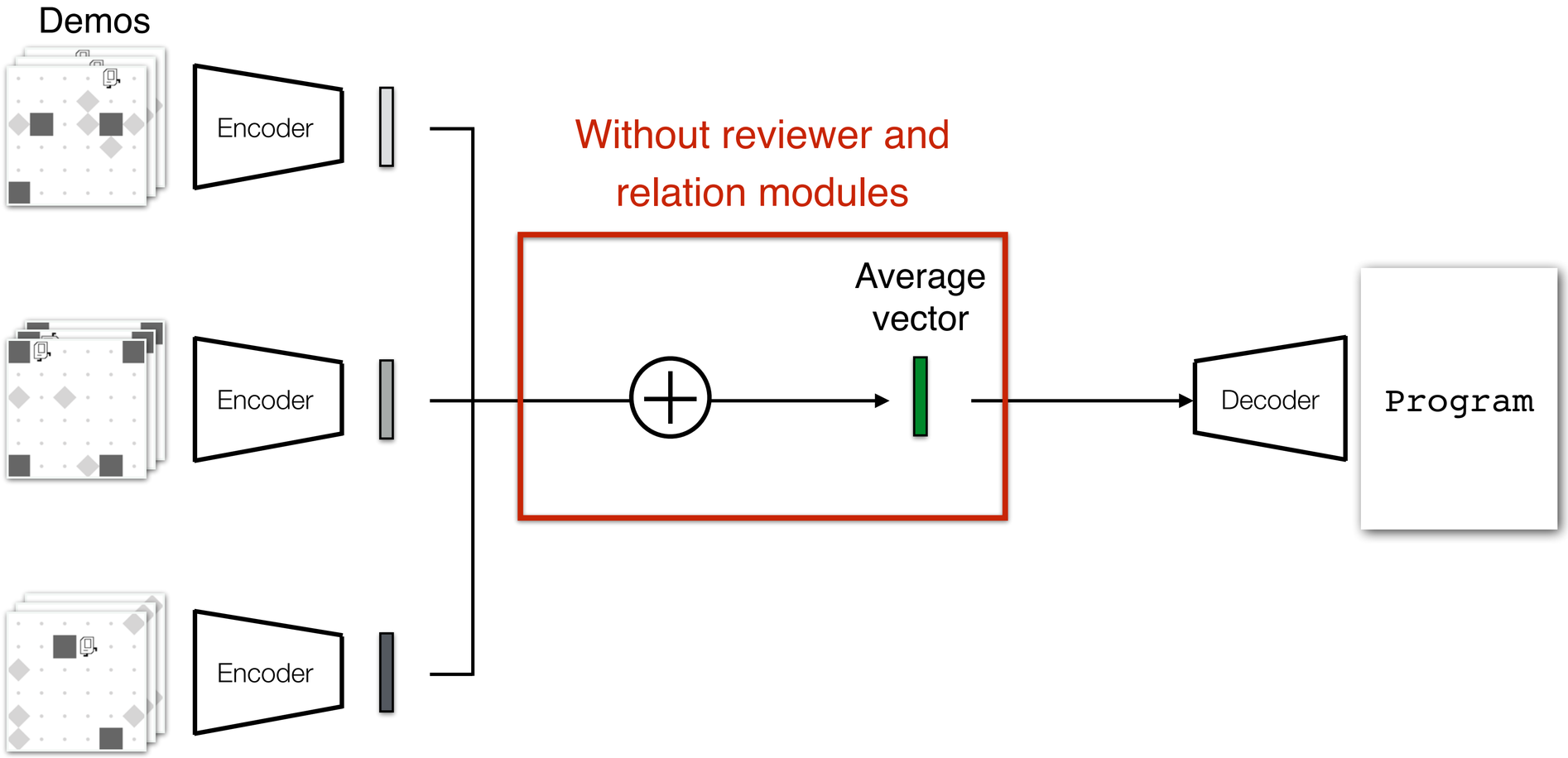


Richard E Pattis. “Karel the robot: a gentle introduction to the art of programming.” John Wiley & Sons, Inc., 1981

Kempka et al., “Vizdoom: A doom-based ai research platform for visual reinforcement learning.” In CIG, 2016

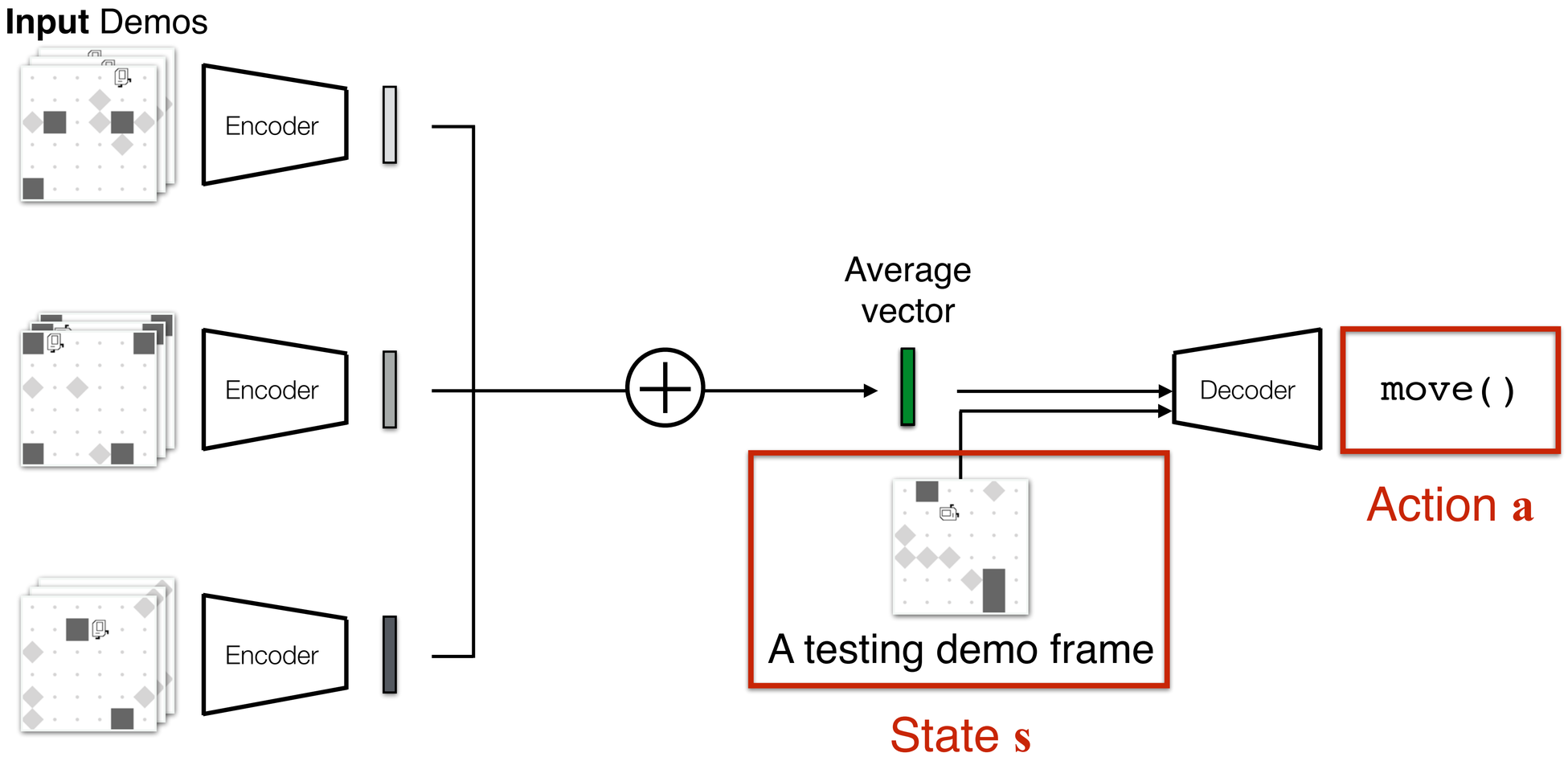
Baselines

- Program **synthesis** baseline
- Program **induction** baseline



Baselines

- Program **synthesis** baseline
- Program **induction** baseline



Example Result: Karel

Ground truth

```
def run():
    if frontIsClear():
        move()
    else:
        turnLeft()
    move()
    repeat(2):
        turnRight()
        putMarker()
```

Synthesis baseline

```
def run():
    move()
    move()
    turnRight()
    putMarker()
    turnRight()
    putMarker()
```

Miss the if-else statement

Example Result: Karel

Ground truth

```
def run():
    if frontIsClear():
        move()
    else:
        turnLeft()
    move()
    repeat(2):
        turnRight()
        putMarker()
```

Synthesis baseline

```
def run():
    move()
    move()
    turnRight()
    putMarker()
    turnRight()
    putMarker()
```

Ours

```
def run():
    if frontIsClear():
        move()
    else:
        turnLeft()
    move()
    turnRight()
    putMarker()
    turnRight()
    putMarker()
```

Example Result: ViZDoom

Ground truth

```
def run():
  if inTarget(Demon):
    attack()
    moveLeft()
  else:
    moveRight()
  if isThere(Demon):
    attack()
  moveLeft()
```

Synthesis baseline

```
def run():
  while(inTarget(
      HellKnight)):
    attack()
  if isThere(Demon):
    moveRight()
    attack()
  else:
    moveLeft()
```

Example Result: ViZDoom

Ground truth

```
def run():
  if inTarget(Demon):
    attack()
    moveLeft()
  else:
    moveRight()
  if isThere(Demon):
    attack()
  moveLeft()
```

Synthesis baseline

```
def run():
  while(inTarget(
      HellKnight)):
    attack()
  if isThere(Demon):
    moveRight()
    attack()
  else:
    moveLeft()
```

Ours

```
def run():
  if inTarget(Demon):
    attack()
    moveLeft()
  else:
    moveRight()
  if isThere(Demon):
    attack()
  moveLeft()
```

Quantitative Result: Infer Programs

Sequence Accuracy

- Measure the accuracy based on code sequences

Synthesized program

```
def run():
  if A():
    x()
  else:
    while(B()):
      y()
    z()
```

Ground truth program

```
def run():
  if A():
    x()
  else:
    repeat(5):
      y()
    z()
```

Synthesized program

```
def run():
  if A():
    x()
  else:
    y()
```

Ground truth program

```
def run():
  if not A():
    y()
  else:
    x()
```

Synthesized program

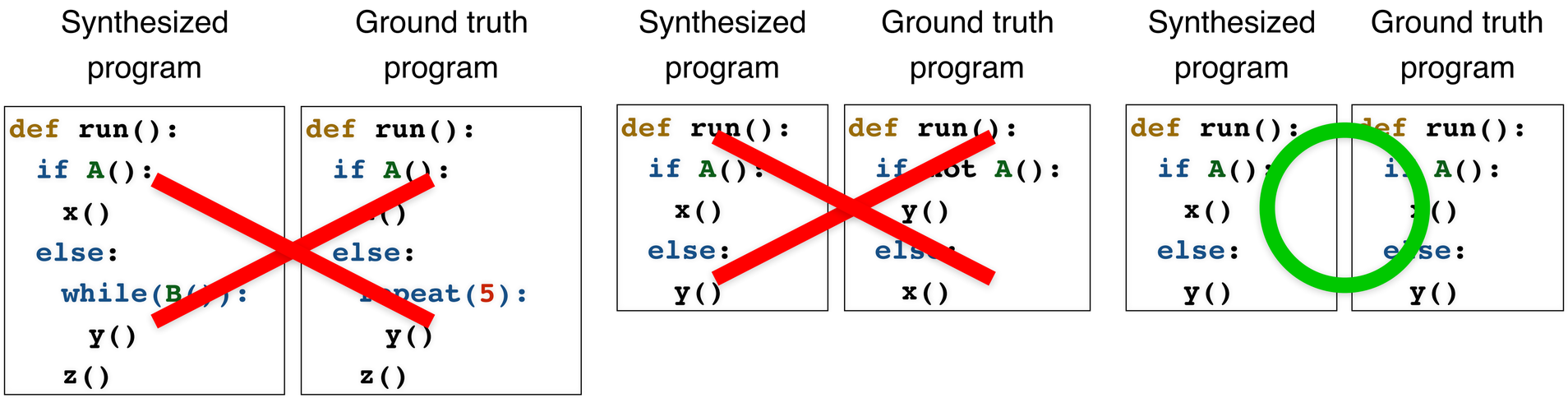
```
def run():
  if A():
    x()
  else:
    y()
```

Ground truth program

```
def run():
  if A():
    x()
  else:
    y()
```

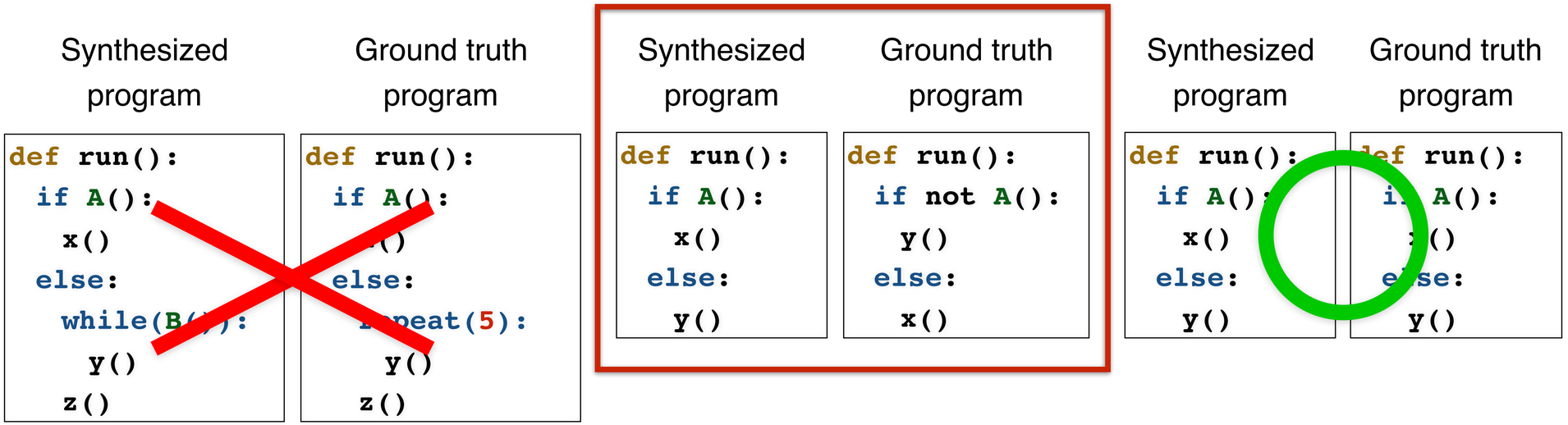
Sequence Accuracy

- Measure the accuracy based on code sequences



Sequence Accuracy

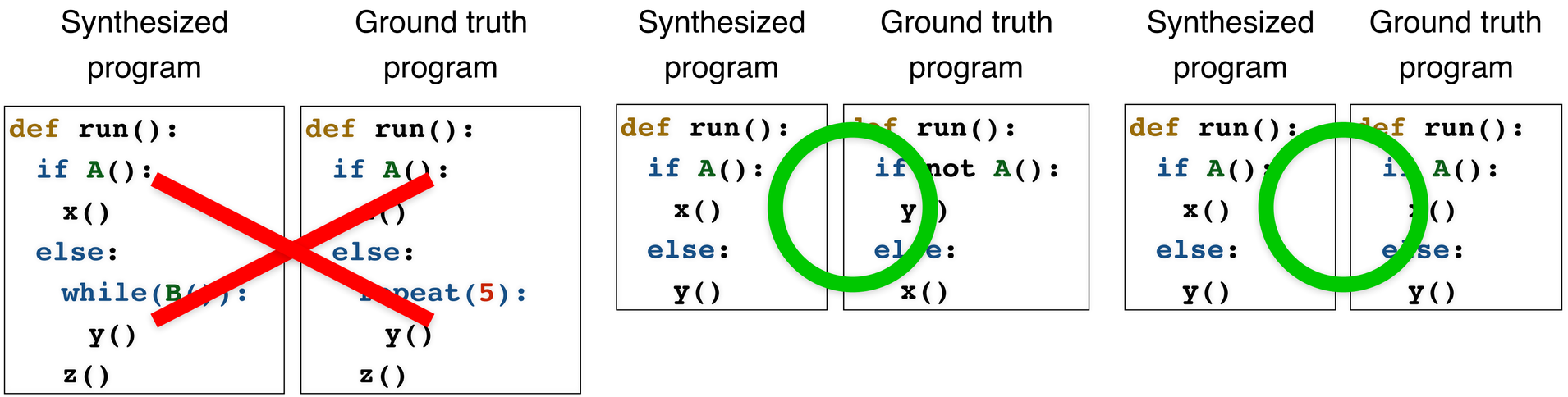
- Measure the accuracy based on code sequences



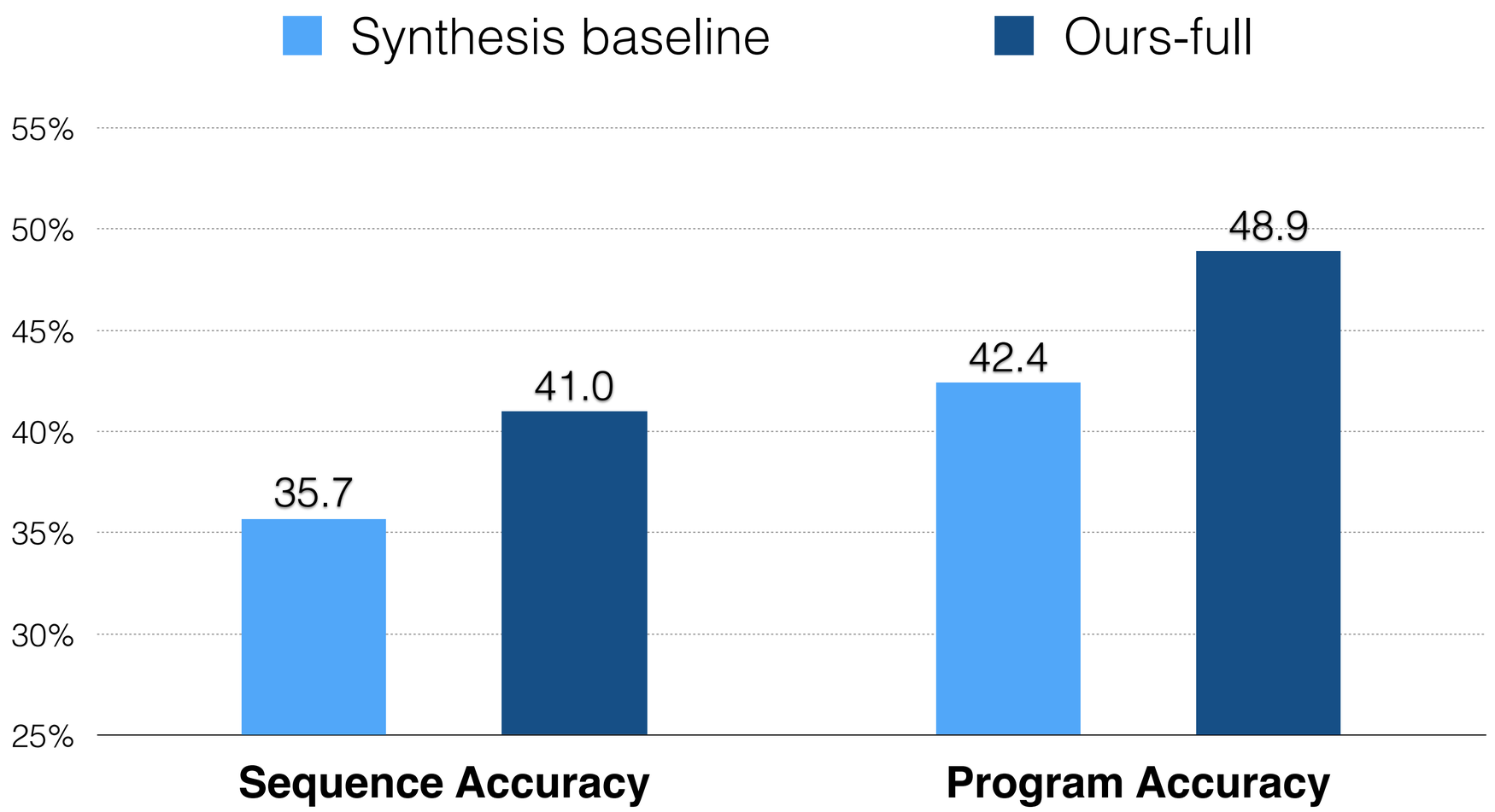
Program aliasing
Different codes with identical program semantics

Program Accuracy

- Compare programs in the program semantic space
  - With some assumptions
    - ex. termination of loops

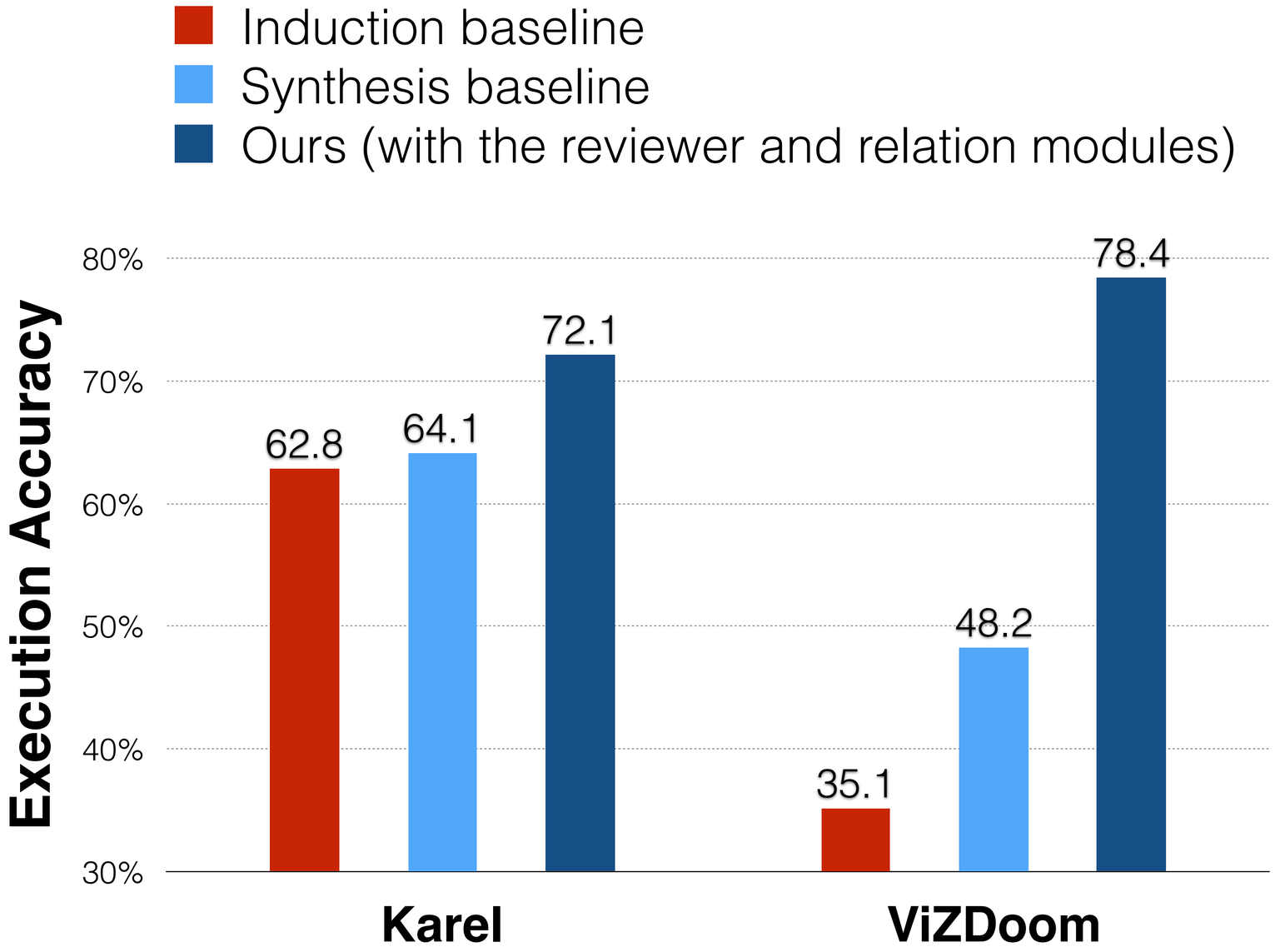


Quantitative Result: Infer Programs

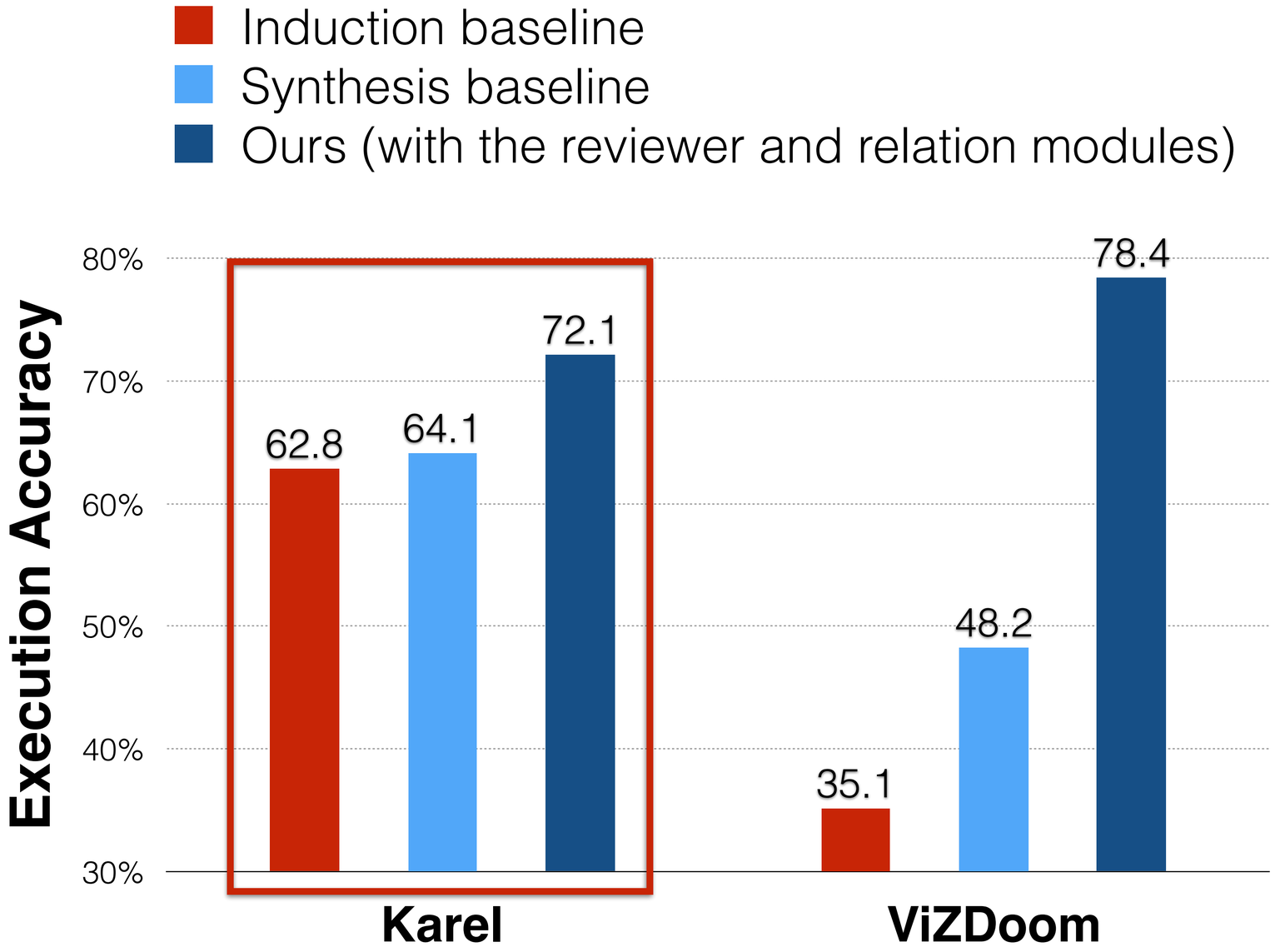


Quantitative Result: Execution

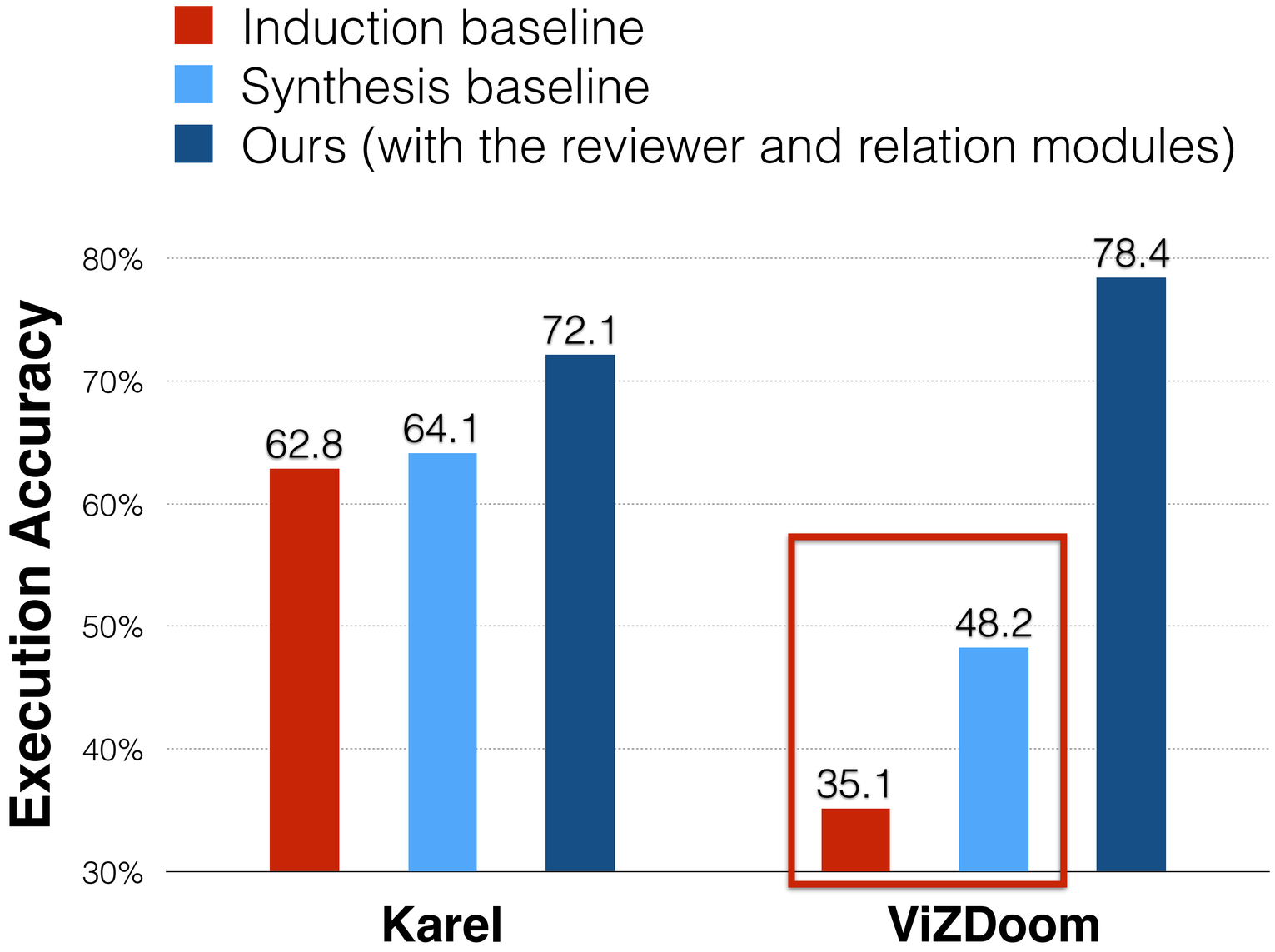
Quantitative Result: Execution



Quantitative Result: Execution

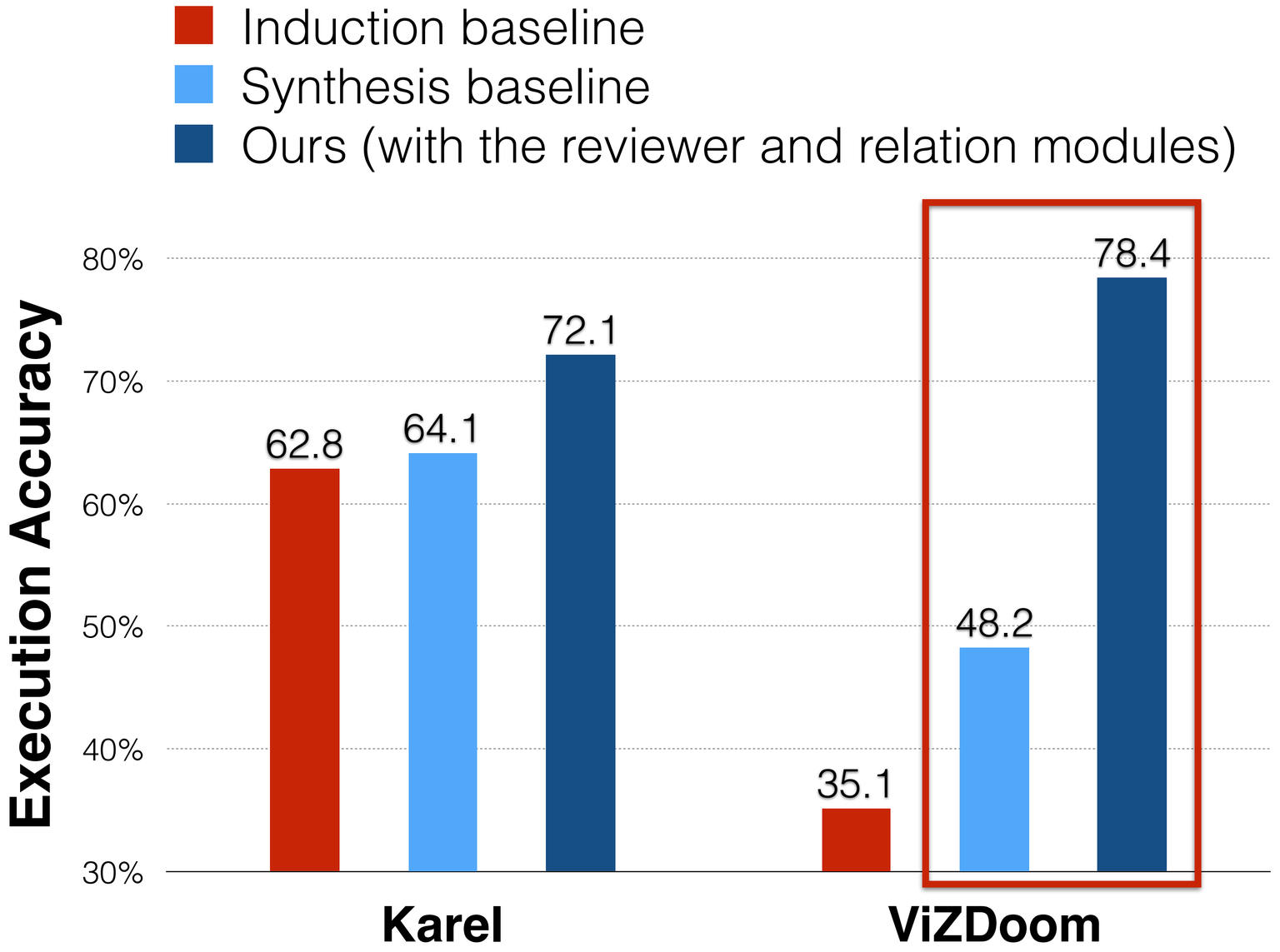


Quantitative Result: Execution



Synthesize programs to describe behaviors

Quantitative Result: Execution



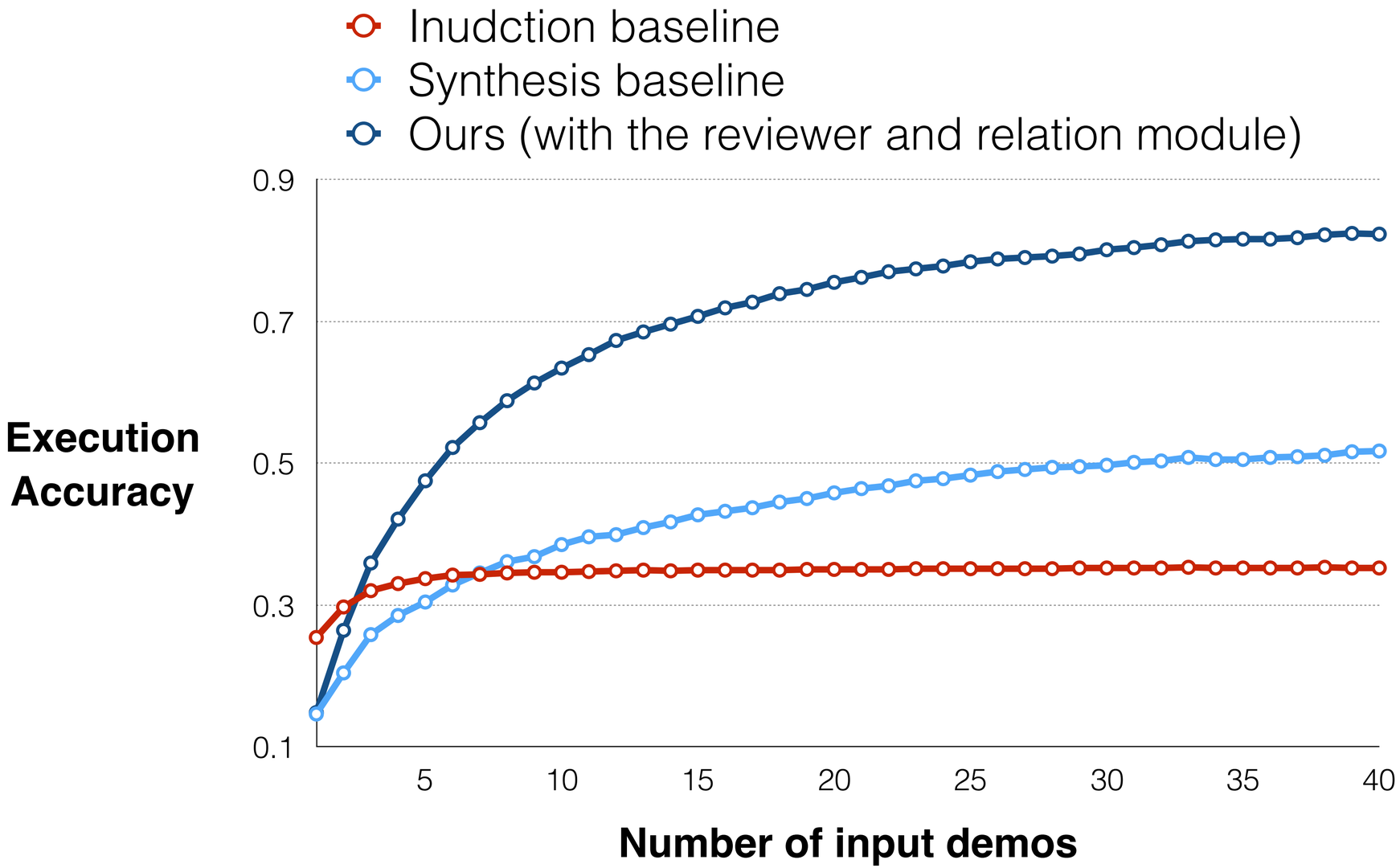
Synthesize programs to describe behaviors

Reviewer module

Relation module

More Results

Effect of Additional Demos

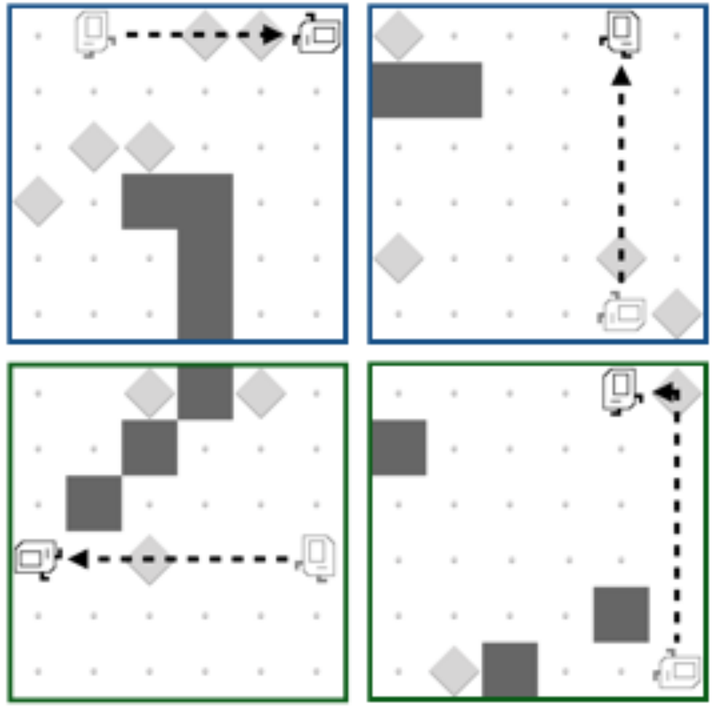


Incomplete Seen Denominations

The seen demos do not completely capture **the underlying behavior**

Underlying Program

```
def run():
  turnRight()
  turnRight()
  while frontIsClear():
    move()
  if markersPresent():
    turnLeft()
    move()
  else:
    turnRight()
```

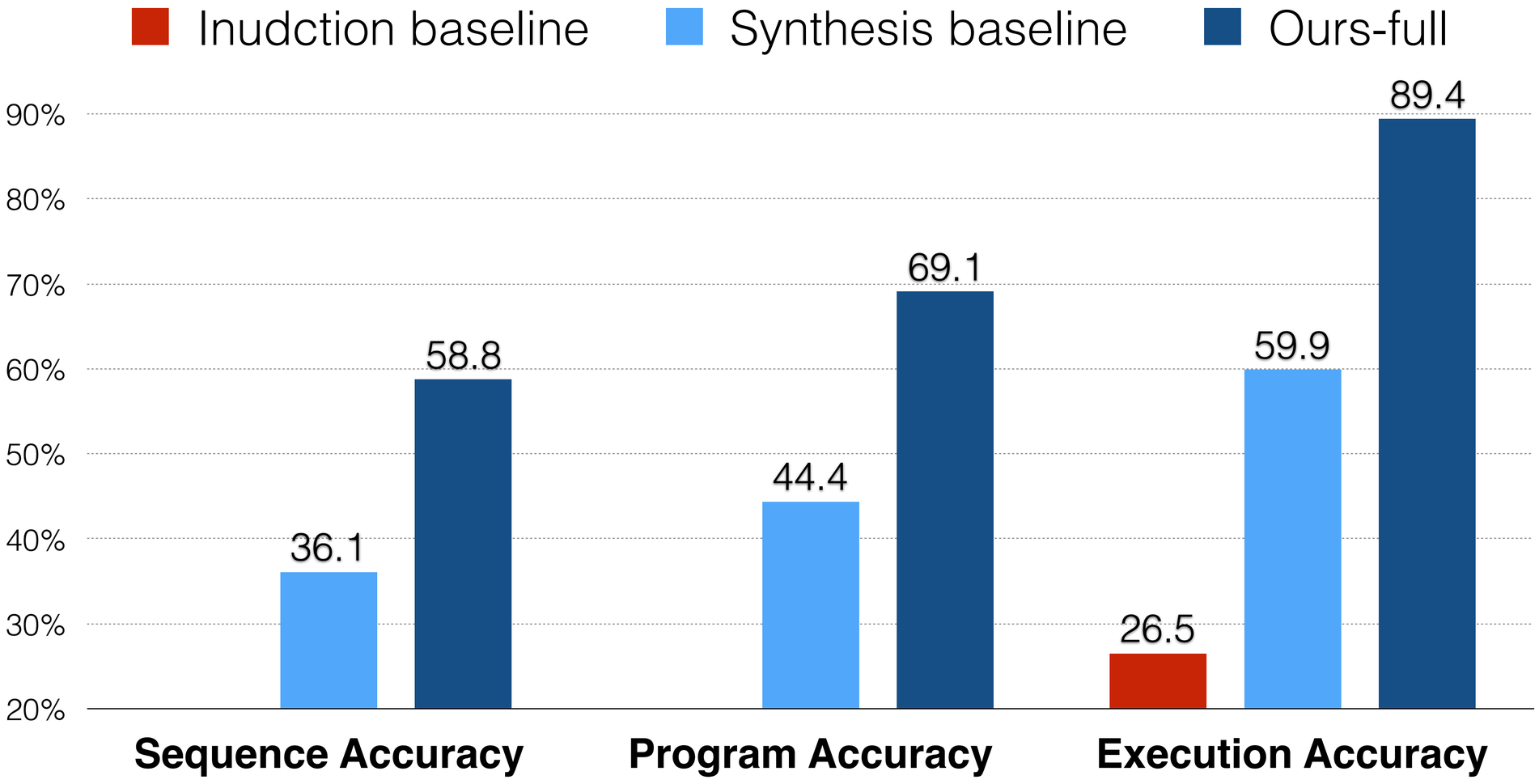


Synthesized Program

```
def run():
  turnRight()
  turnRight()
  while frontIsClear():
    move()
  else:
    turnRight()
```

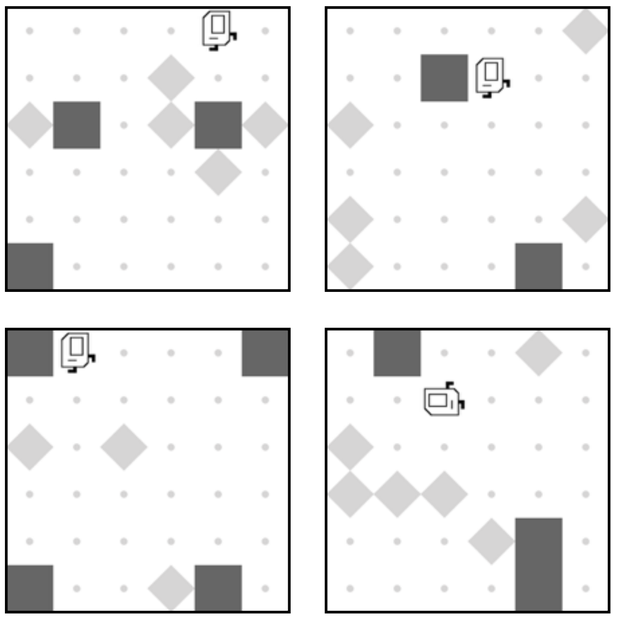
If-else experiment

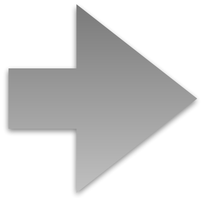
Each program: **single if-else statement** with two branching consequences



Conclusion

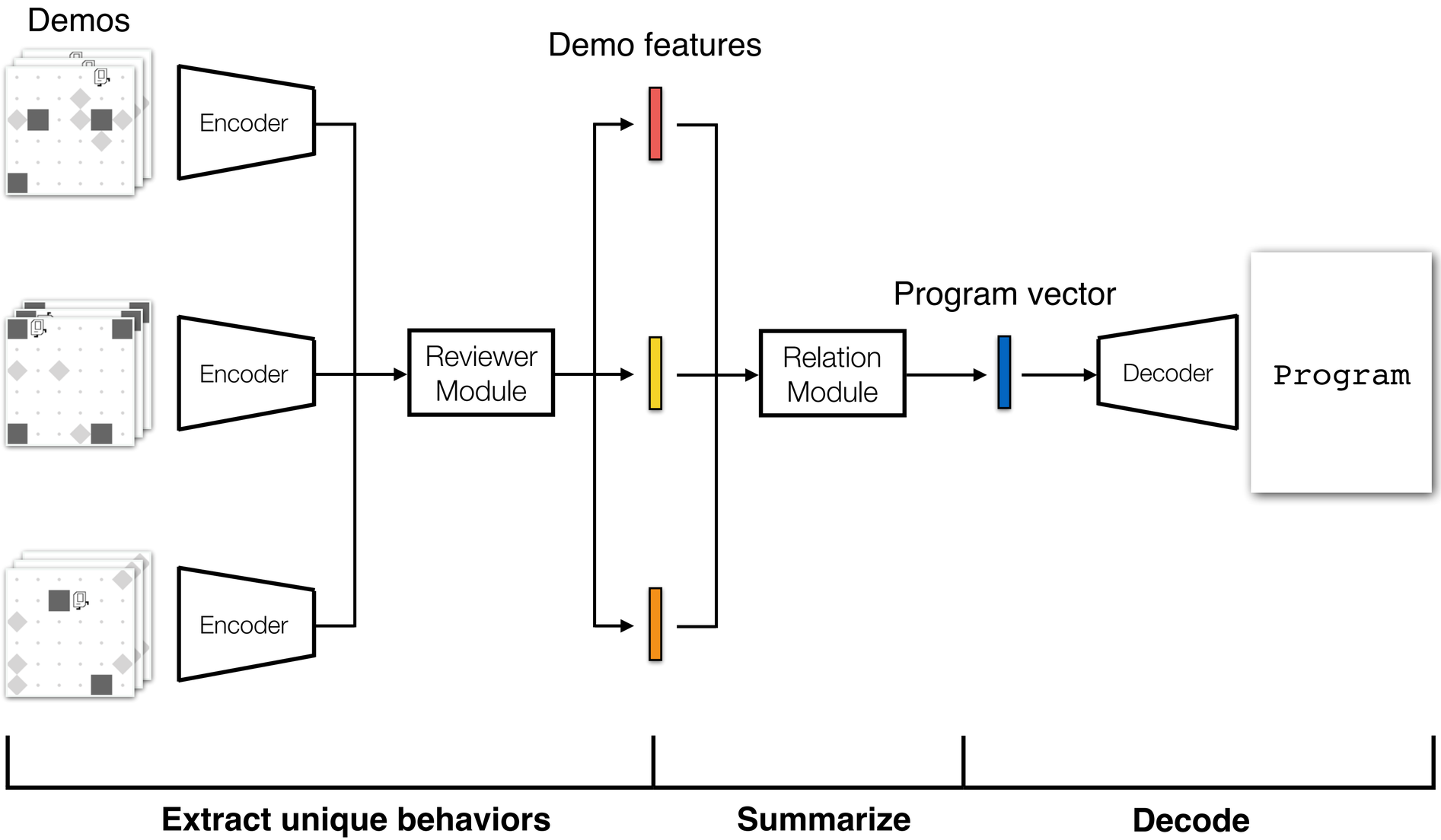
Neural Program Synthesis from Diverse Demonstration Videos



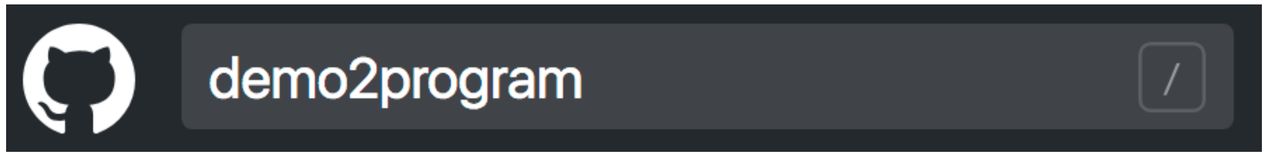


```
def run()
   if frontIsClear():
      move()
   else:
      turnLeft()
   move()
   turnLeft
   repeat(2):
      turnRight()
      putMarker()
```

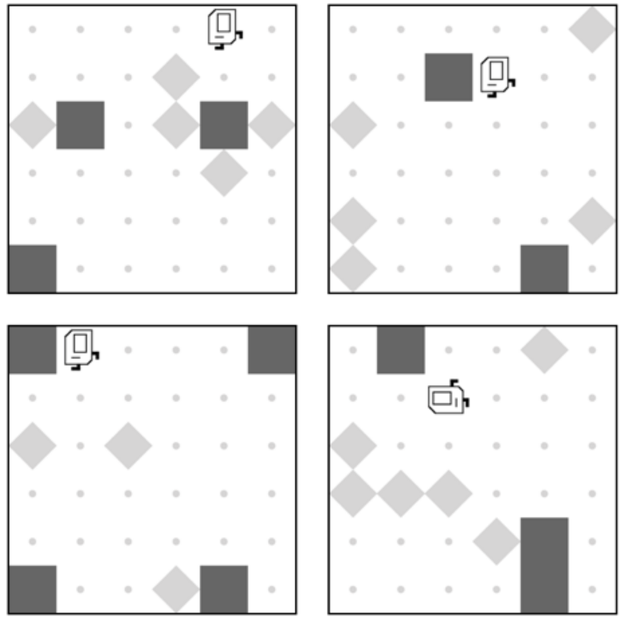
Neural Program Synthesis from Diverse Demonstration Videos



Codes, datasets, and checkpoints are available at



Questions?



```
def run()
  if frontIsClear():
    move()
  else:
    turnLeft()
  move()
  turnLeft()
  repeat(2):
    turnRight()
    putMarker()
```