Neural Program Synthesis from Diverse Demonstration Videos

University of Southern California [1] POSTECH [2] Pohang University of Science and Technology

Shao-Hua Sun*[1], Hyeonwoo Noh*[2], Sriram Somasundaram[1], Joseph Lim[1]

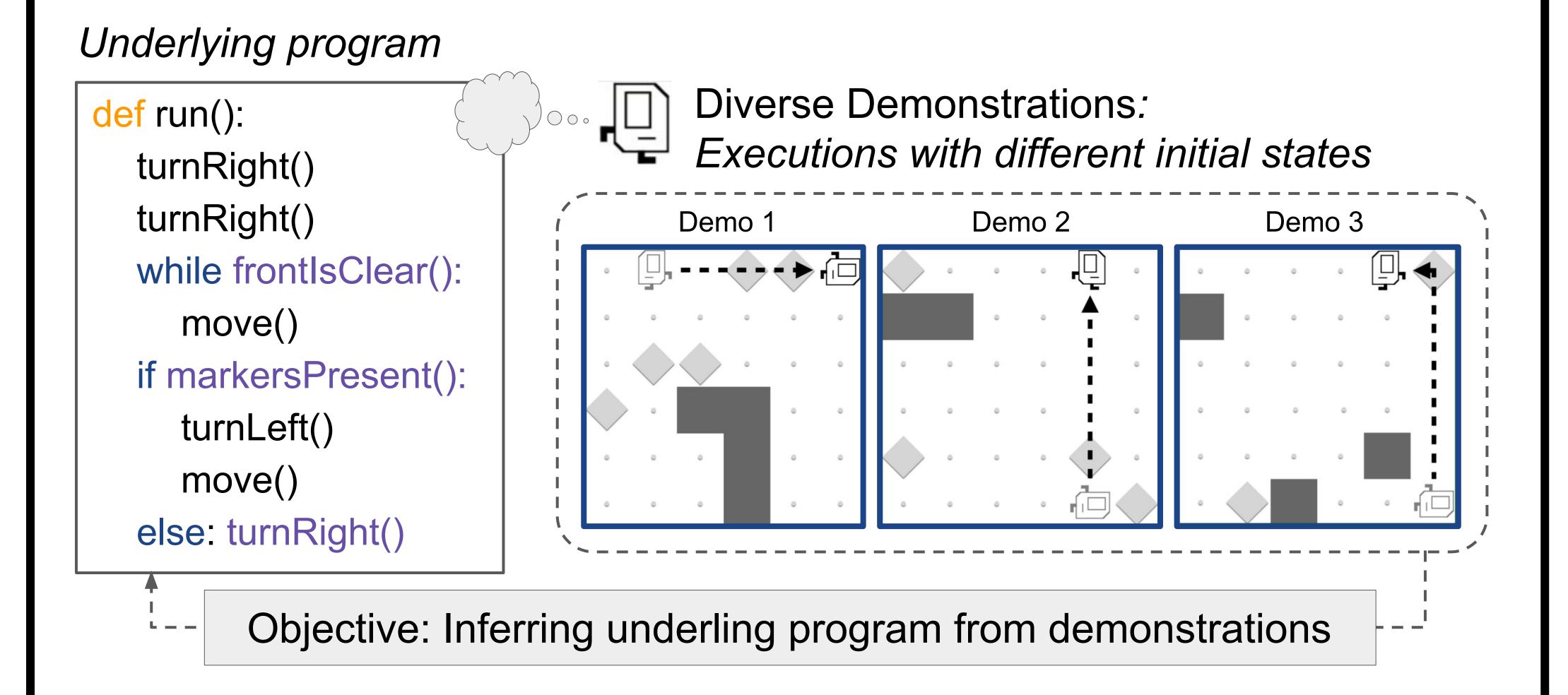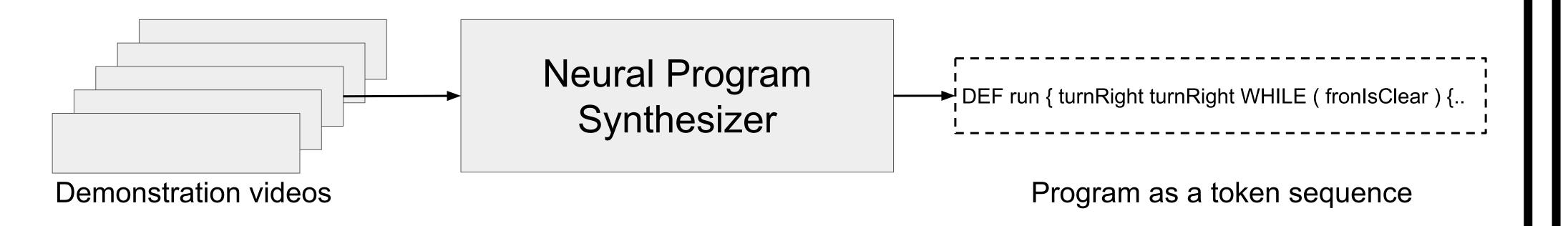shaohuas@usc.edu shgusdngogo@postech.ac.kr sriramso@usc.edu limjj@usc.edu

https://shaohua0116.github.io/demo2program/

Problem Statement

Q: Could we infer underlying program of a behavior from diverse demonstration videos?
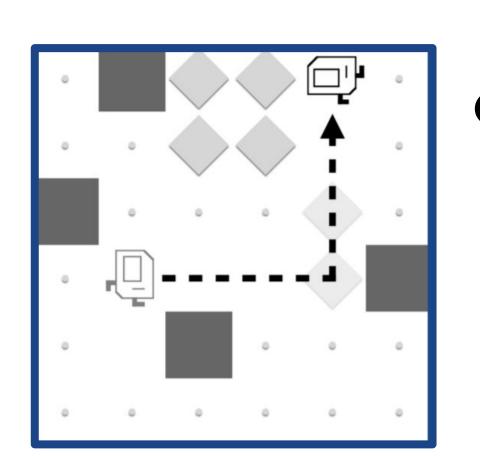
Underlying program

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

Diverse Demonstrations:
Executions with different initial states

Demo 1   Demo 2   Demo 3

Objective: Inferring underling program from demonstrations

❏ Our approach: Neural Program Synthesis

Demonstration videos → Neural Program Synthesizer → DEF run { turnRight turnRight WHILE (frontIsClear) {..
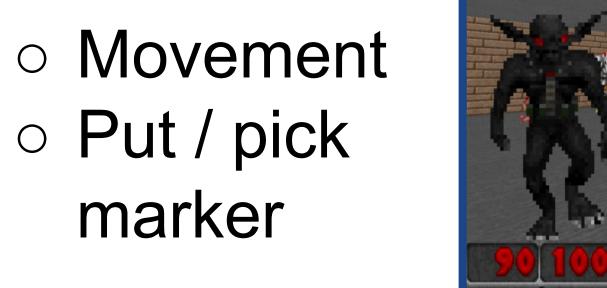
Program as a token sequence

❏ Contribution

- Program synthesis for explicit modeling of underlying programs
- Architecture for summarizing diverse demonstrations
- Auxiliary objectives enhancing program synthesis
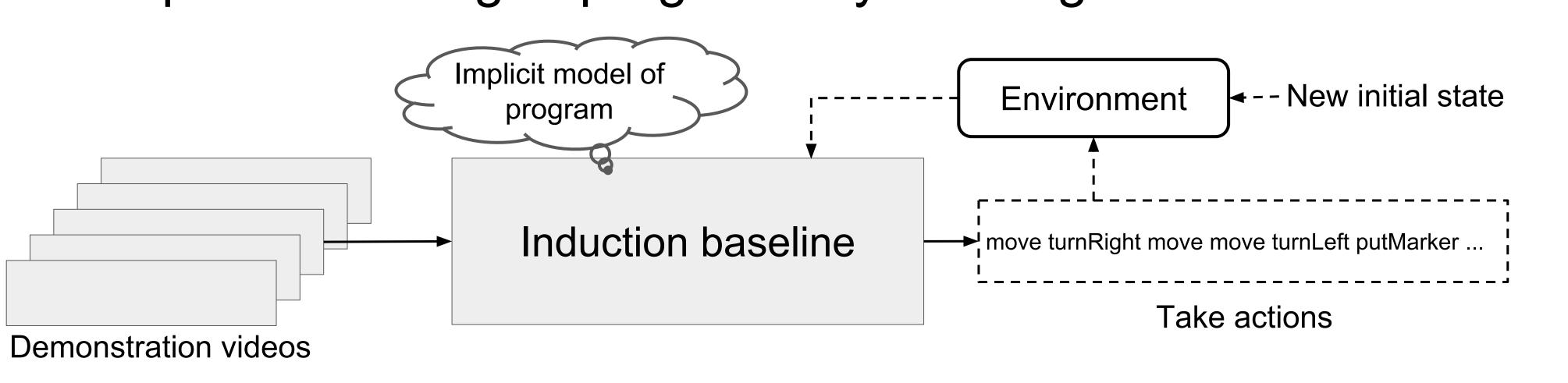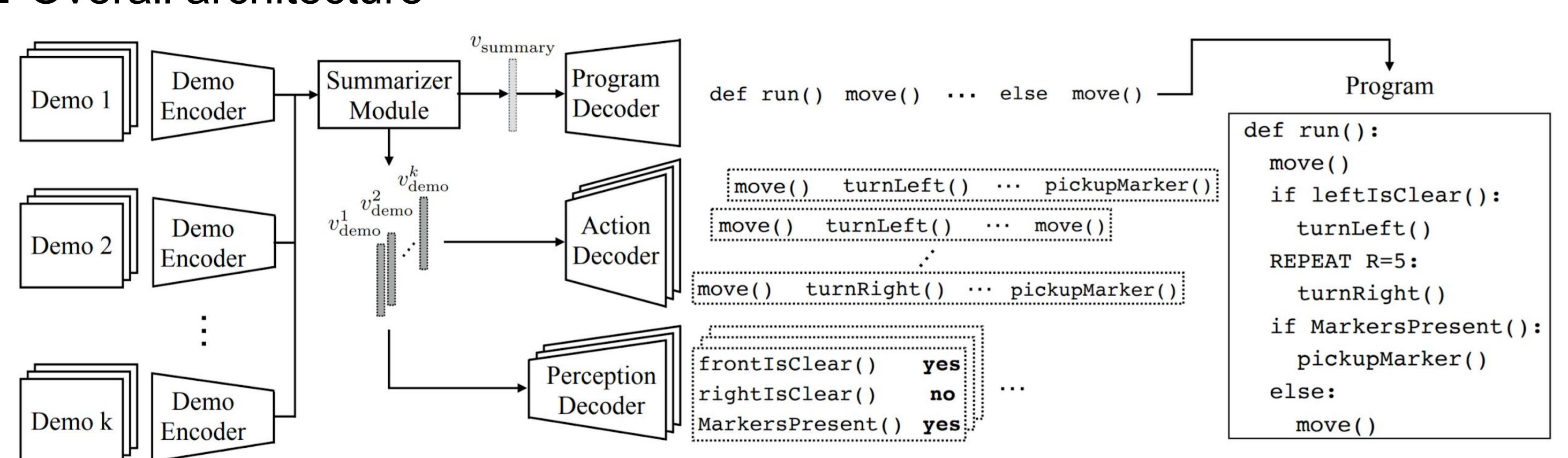
Experimental Setting

- *Karel*:
  - Movement
  - Put / pick marker
- *ViZDoom*:
  - Movement
  - Kill monster
  - Select weapon

❏ Induction baseline (Few-shot imitation)

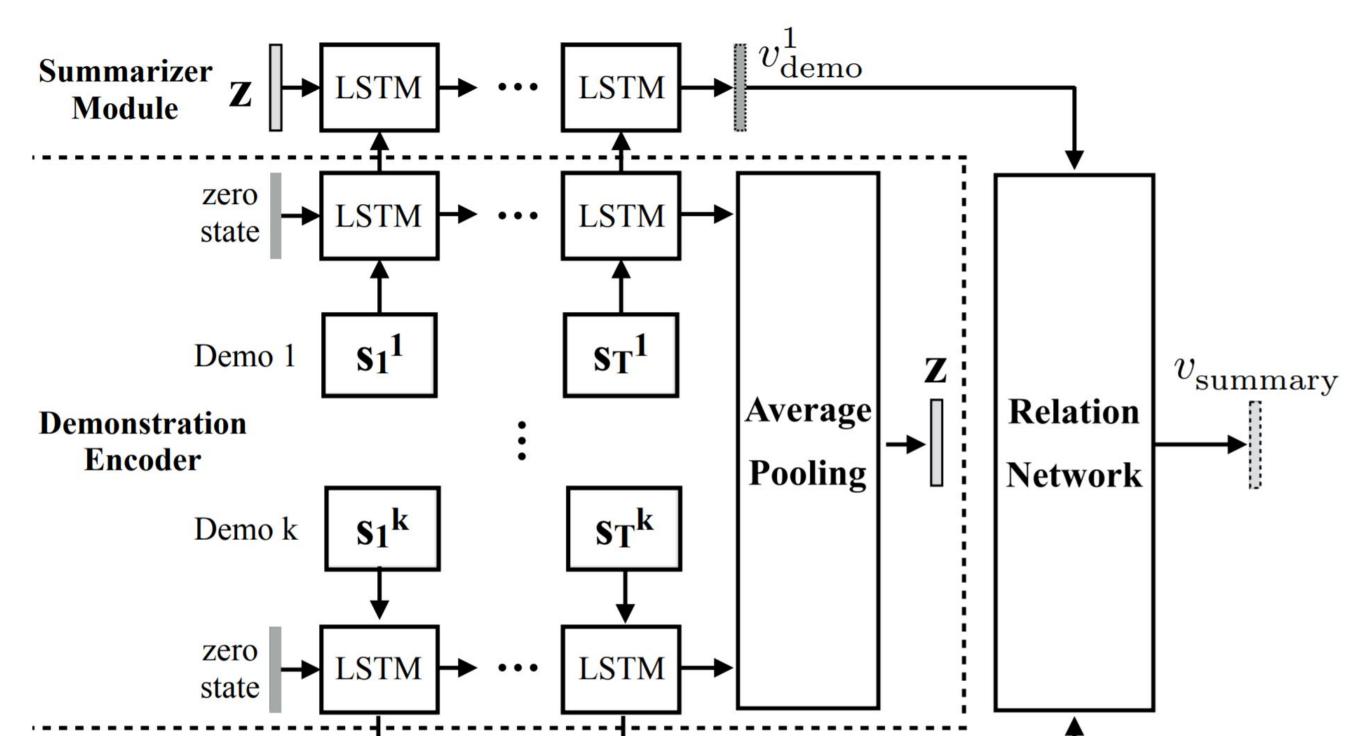- Implicit modeling of programs by learning to imitate

Implicit model of program → Environment ← New initial state

Demonstration videos → Induction baseline → move turnRight move move turnLeft putMarker ...

Take actions

❏ Synthesis baseline

- Program synthesis without summarizer / multi-task loss

Proposed Model

❏ Overall architecture

Demo 1 → Demo Encoder
Demo 2 → Demo Encoder
Demo k → Demo Encoder
→ Summarizer Module → v_{summary} → Program Decoder → Program

v_{demo}^1 v_{demo}^2 v_{demo}^k

Action Decoder:
move() turnLeft() ··· pickupMarker()
move() turnLeft() ··· move()
move() turnRight() ··· pickupMarker()

Perception Decoder:
frontIsClear() **yes**
rightIsClear() **no**
MarkersPresent() **yes**

```
def run():
    move()
    if leftIsClear():
        turnLeft()
    REPEAT R=5:
        turnRight()
    if MarkersPresent():
        pickupMarker()
    else:
        move()
```

❏ Summarizer module

- *Summarizer module*:
  - Inferring actions and underlying conditions from demonstrations
- *Multi-task loss*:
  - Learning representation for inferring action / perception from demonstrations
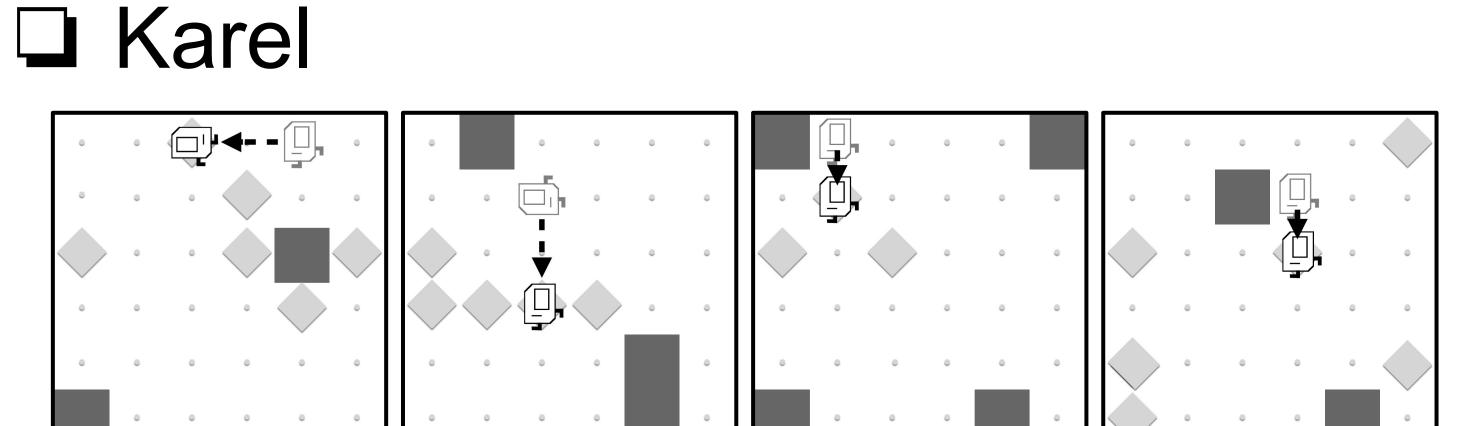
Qualitative Results

❏ Karel

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

- Ours-full
```
def run():
    if frontIsClear():
        move()
    else:
        turnLeft()
    move()
    turnLeft()
    turnRight()
    putMarker()
    turnRight()
    putMarker()
```
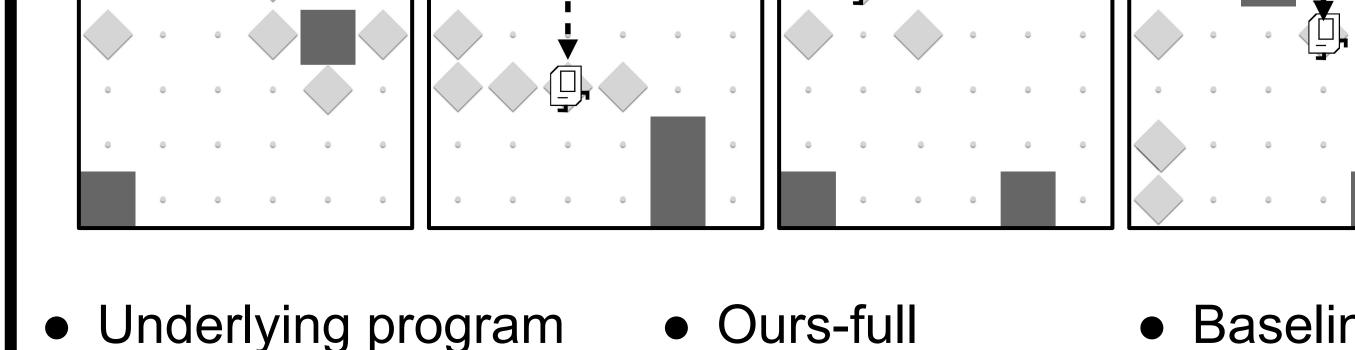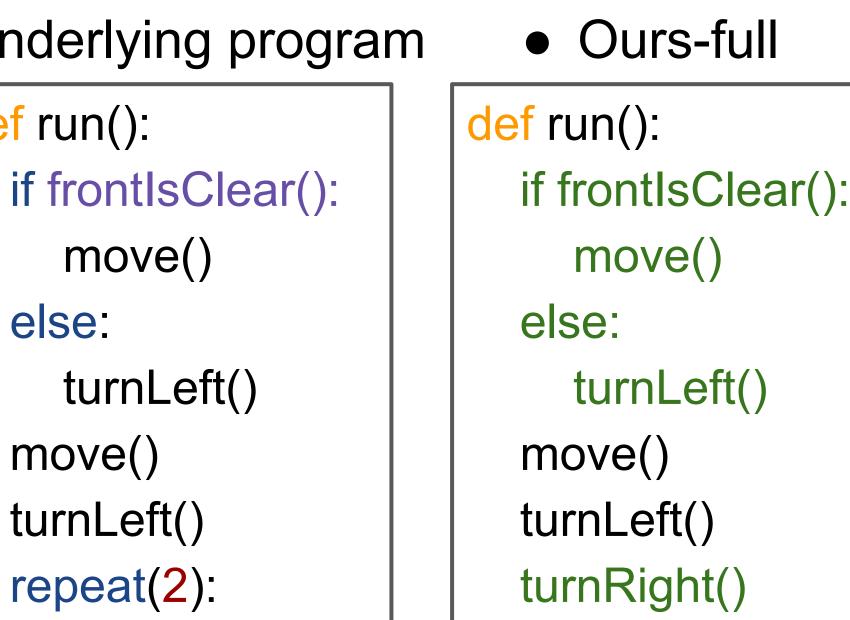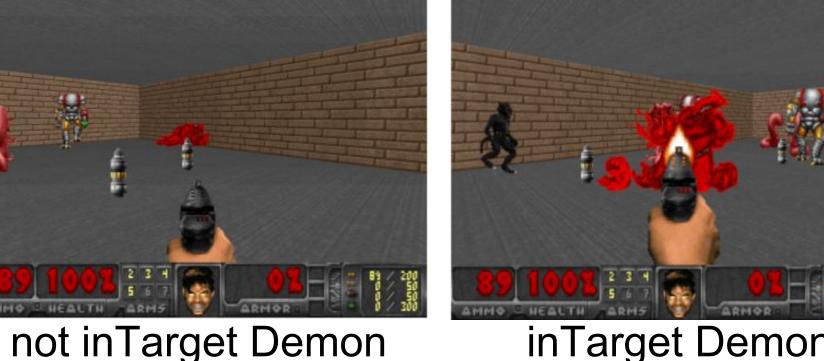
- Baseline
```
def run():
    move()
    move()
    turnLeft()
    turnRight()
    putMarker()
    turnRight()
    putMarker()
```

❏ ViZDoom

- Underlying program
```
def run():
    if inTarget HellKnight:
        attack()
    if not inTarget Demon:
        moveRight()
    else:
        attack()
```

- Synthesized program
```
def run():
    if inTarget HellKnight:
        attack()
    if inTarget Demon:
        attack()
    else:
        moveRight()
```

Demo 1   Demo 2

inTarget HellKnight   inTarget HellKnight and inTarget Demon

inTarget HellKnight → attack()   inTarget HellKnight → attack()

not inTarget Demon → moveRight()   inTarget Demon → attack()

Experimental Results

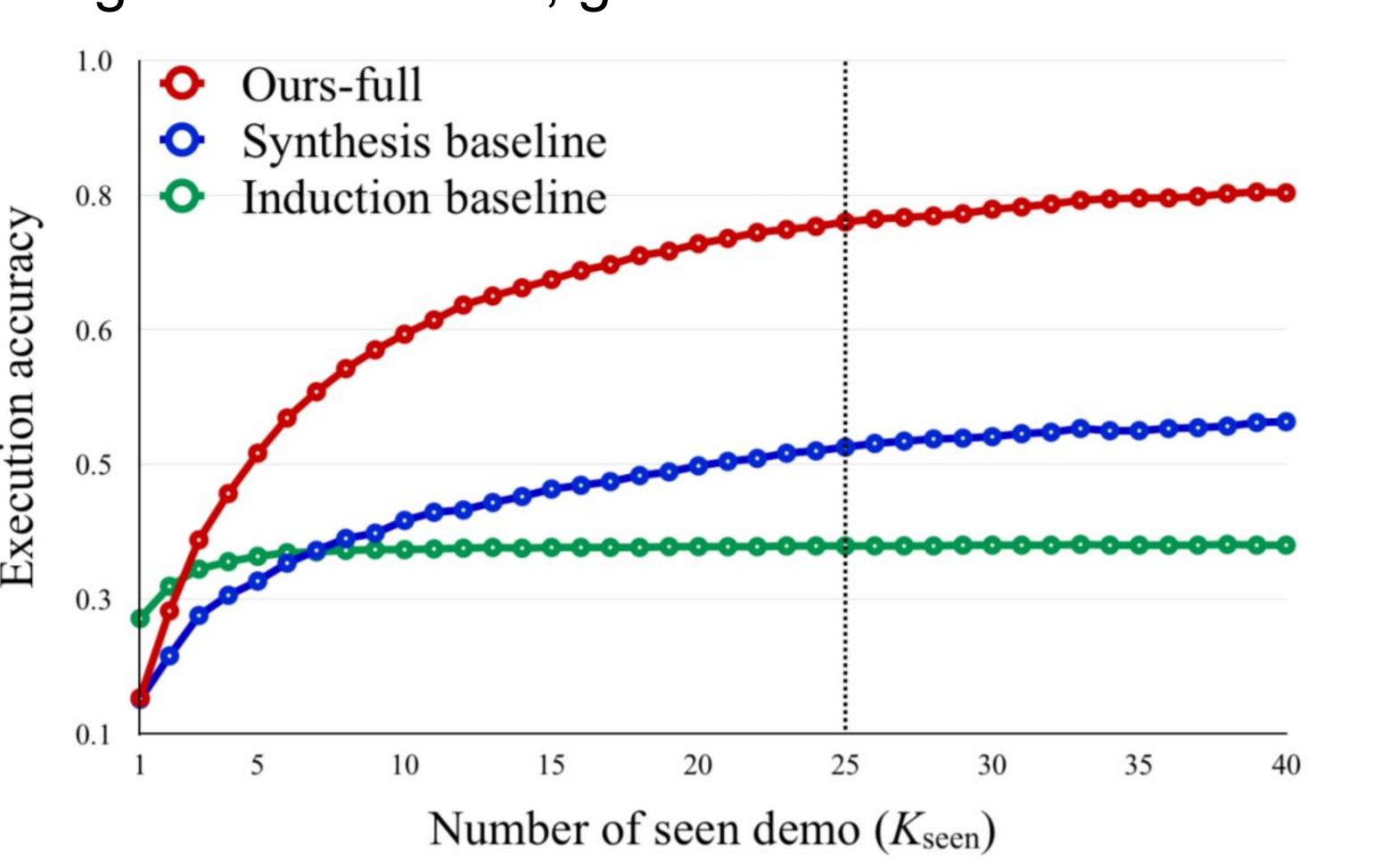### ❏ Advantage of explicit modeling of program
❏ Effect of summarizer / multi-task loss

- Karel

Methods	Execution	Program	Sequence
Induction baseline	62.8% (69.1%)	-	-
Synthesis baseline	64.1%	42.4%	35.7%
+ summarizer (ours)	68.6%	45.3%	38.3%
+ multi-task loss (ours-full)	**72.1%**	**48.9%**	**41.0%**

- ViZDoom

Methods	Execution	Program	Sequence
Induction baseline	35.1% (60.6%)	-	-
Synthesis baseline	48.2%	39.9%	33.1%
Ours-full	**78.4%**	**62.5%**	**53.2%**

❏ Generalization to more / less demonstrations

- Training with 25 demos, generalization to more / less demos

❏ Effect of summarizer with more demonstrations

- Advantage of summarizer is bigger with more demonstrations

Methods	k=3	k=5	k=10
Synthesis baseline	58.5%	60.1%	64.1%
+ summarizer (ours)	**60.6%**	**63.1%**	**68.6%**
Improvement	2.1%	3.0%	4.5%

❏ If-else experiment - inferring condition

- Single IFELSE statement with 1 condition, 2 different actions
- Inferring correct condition is essential

Methods	Execution	Program	Sequence
Induction baseline	26.5% (83.1%)	-	-
Synthesis baseline	59.9%	44.4%	36.1%
Ours-full	**89.4%**	**69.1%**	**58.8%**

Neural Program Synthesis from Diverse Demonstration Videos

Shao-Hua Sun[*1], Hyeonwoo Noh[*2], Sriram Somasundaram[1], Joseph Lim[1]

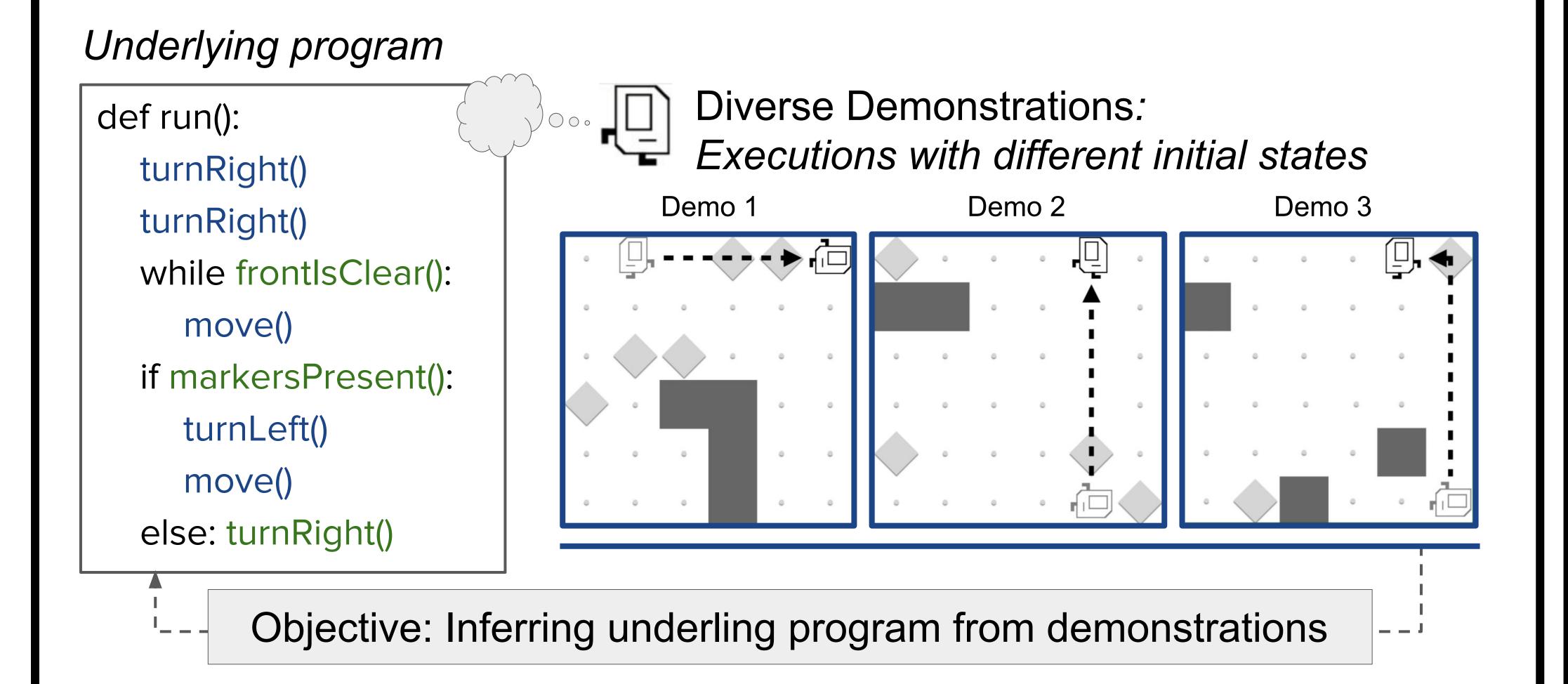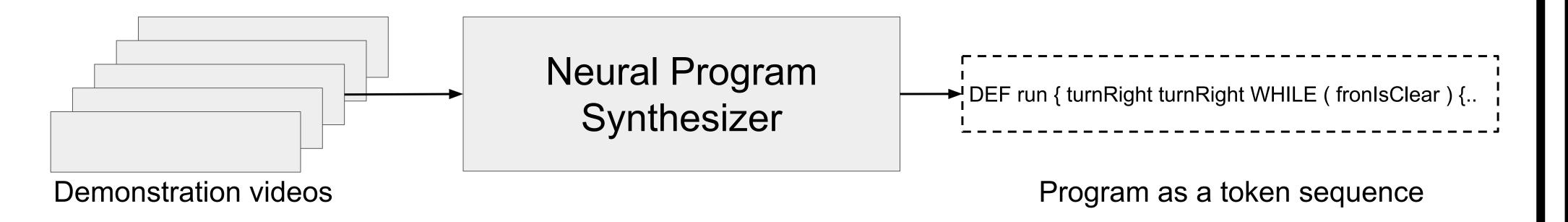shaohuas@usc.edu shgusdngogo@postech.ac.kr   sriramso@usc.edu   limjj@usc.edu

https://shaohua0116.github.io/demo2program/

Problem Statement

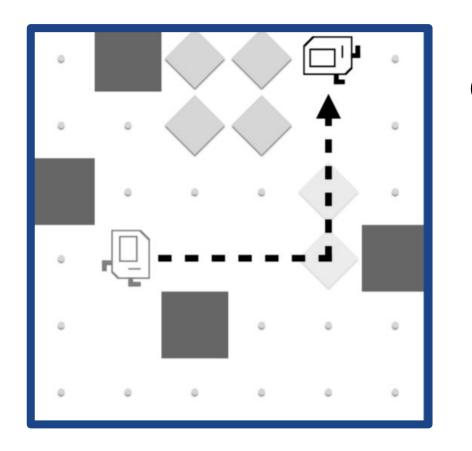Q: Could we infer underlying program of a behavior from diverse demonstration videos?

Underlying program

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

Diverse Demonstrations:
Executions with different initial states

Demo 1   Demo 2   Demo 3

Objective: Inferring underling program from demonstrations

❑ Our approach: Neural Program Synthesis

Demonstration videos → Neural Program Synthesizer → DEF run { turnRight turnRight WHILE (frontIsClear) {..

Program as a token sequence

❑ Contribution
- Program synthesis for explicit modeling of underlying programs
- Architecture for summarizing diverse demonstrations
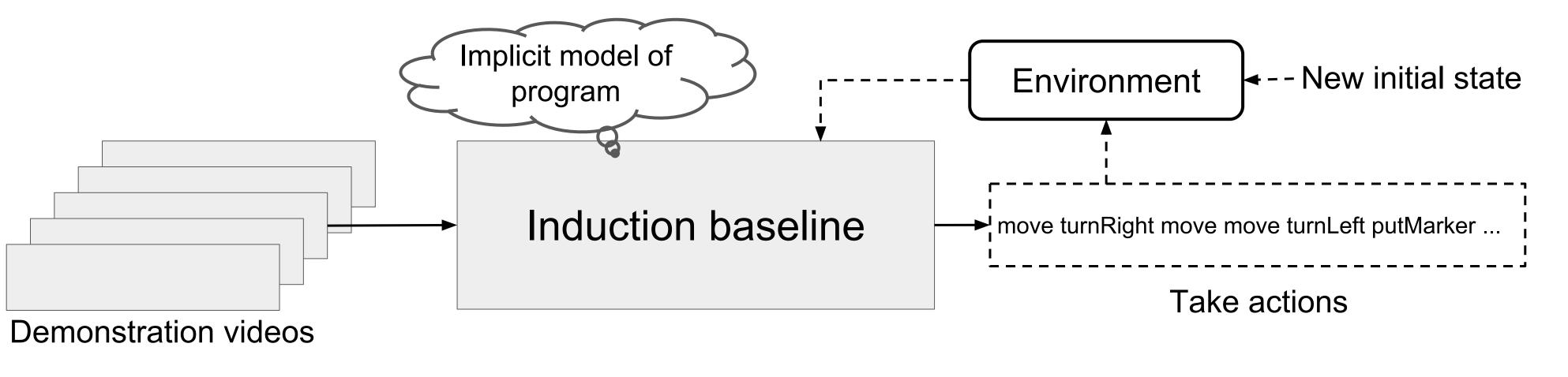- Auxiliary objectives enhancing program synthesis

Experimental Setting

- *Karel*:
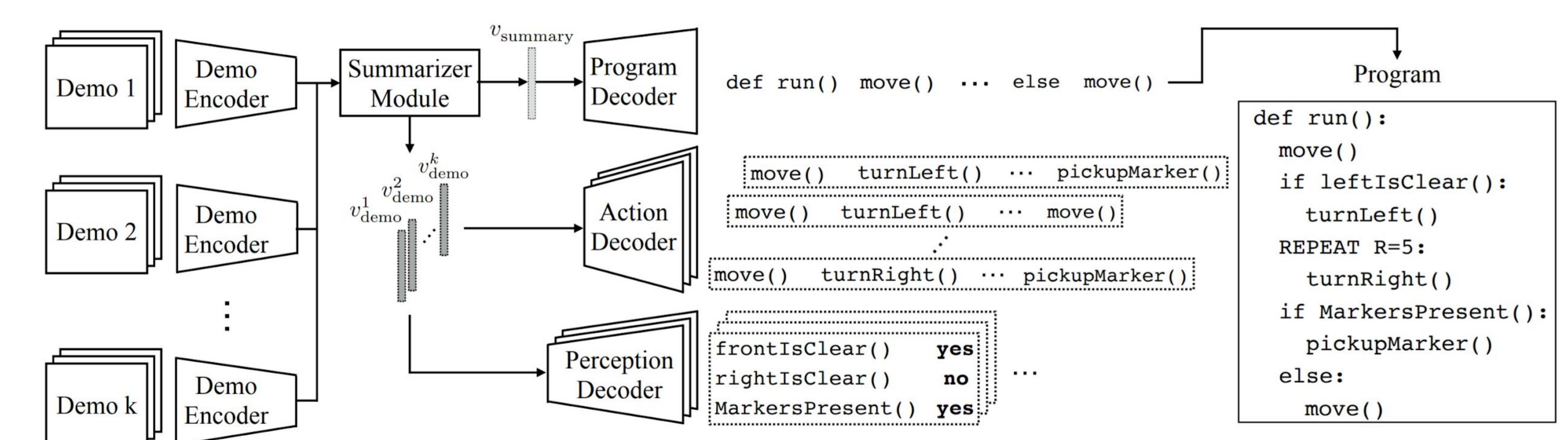 - Movement
  - Put / pick marker
- *Vizdoom*:
  - Movement
  - Kill monster
  - Select weapon

❑ Induction baseline (Few-shot imitation)
- Implicit modeling of programs by learning to imitate

Implicit model of program → Environment ←-- New initial state

Demonstration videos → Induction baseline → move turnRight move move turnLeft putMarker ...

Take actions

❑ Synthesis baseline
- Program synthesis without summarizer / multi-task loss

Proposed Model

❑ Overall architecture

Demo 1 → Demo Encoder
Demo 2 → Demo Encoder
Demo k → Demo Encoder

v_{demo}^1, v_{demo}^2, v_{demo}^k

Summarizer Module → $v_{summary}$ → Program Decoder

Action Decoder

Perception Decoder

```
def run() move() ··· else move()
```
Program

```
move()  turnLeft() ··· pickupMarker()
move()  turnLeft() ··· move()
move()  turnRight() ··· pickupMarker()
```

```
frontIsClear()      yes
rightIsClear()      no      ···
MarkersPresent()    yes
```

```
def run():
    move()
    if leftIsClear():
        turnLeft()
    REPEAT R=5:
        turnRight()
    if MarkersPresent():
        pickupMarker()
    else:
        move()
```

❑ Summarizer module

Summarizer Module: Z → LSTM ··· LSTM → v_{demo}^1

Demonstration Encoder

Demo 1: s_1^1 ··· s_T^1
Demo k: s_1^k ··· s_T^k

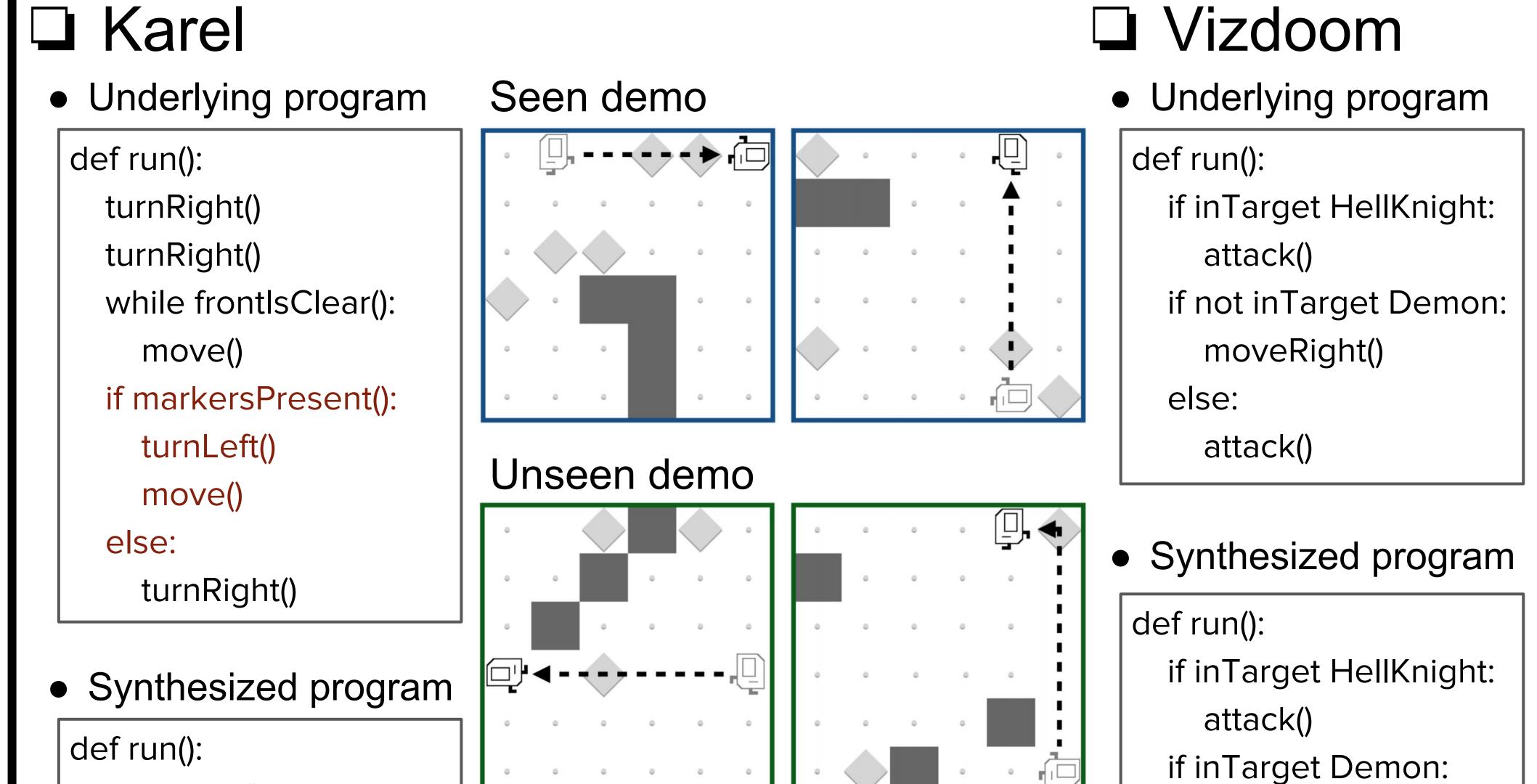Average Pooling → Z → Relation Network → $v_{summary}$

- *Summarizer module*:
  - Inferring actions and underlying conditions from demonstrations
- *Multi-task loss*:
  - Learning representation for inferring action / perception from demonstrations
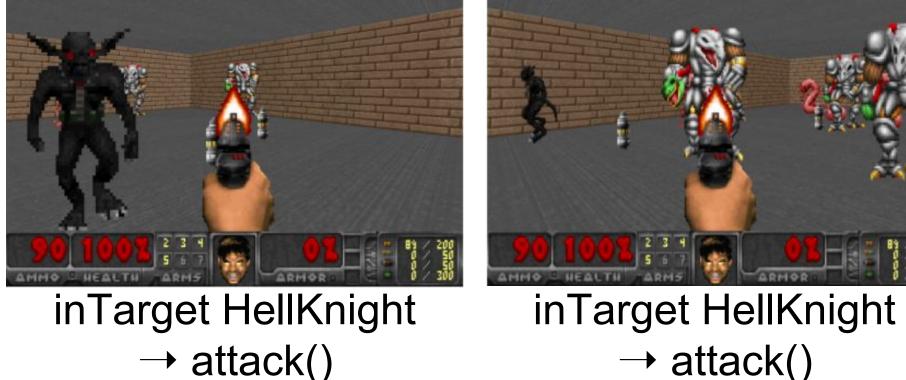
Qualitative Results

❑ Karel
- Underlying program

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

- Synthesized program

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

Seen demo

Unseen demo

❑ Vizdoom
- Underlying program

```
def run():
    if inTarget HellKnight:
        attack()
    if not inTarget Demon:
        moveRight()
    else:
        attack()
```

- Synthesized program

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

Demo 1   Demo 2

inTarget HellKnight | inTarget HellKnight and inTarget Demon

inTarget HellKnight → attack() | inTarget HellKnight → attack()

not inTarget Demon → moveRight() | inTarget Demon → attack()

Experimental Results

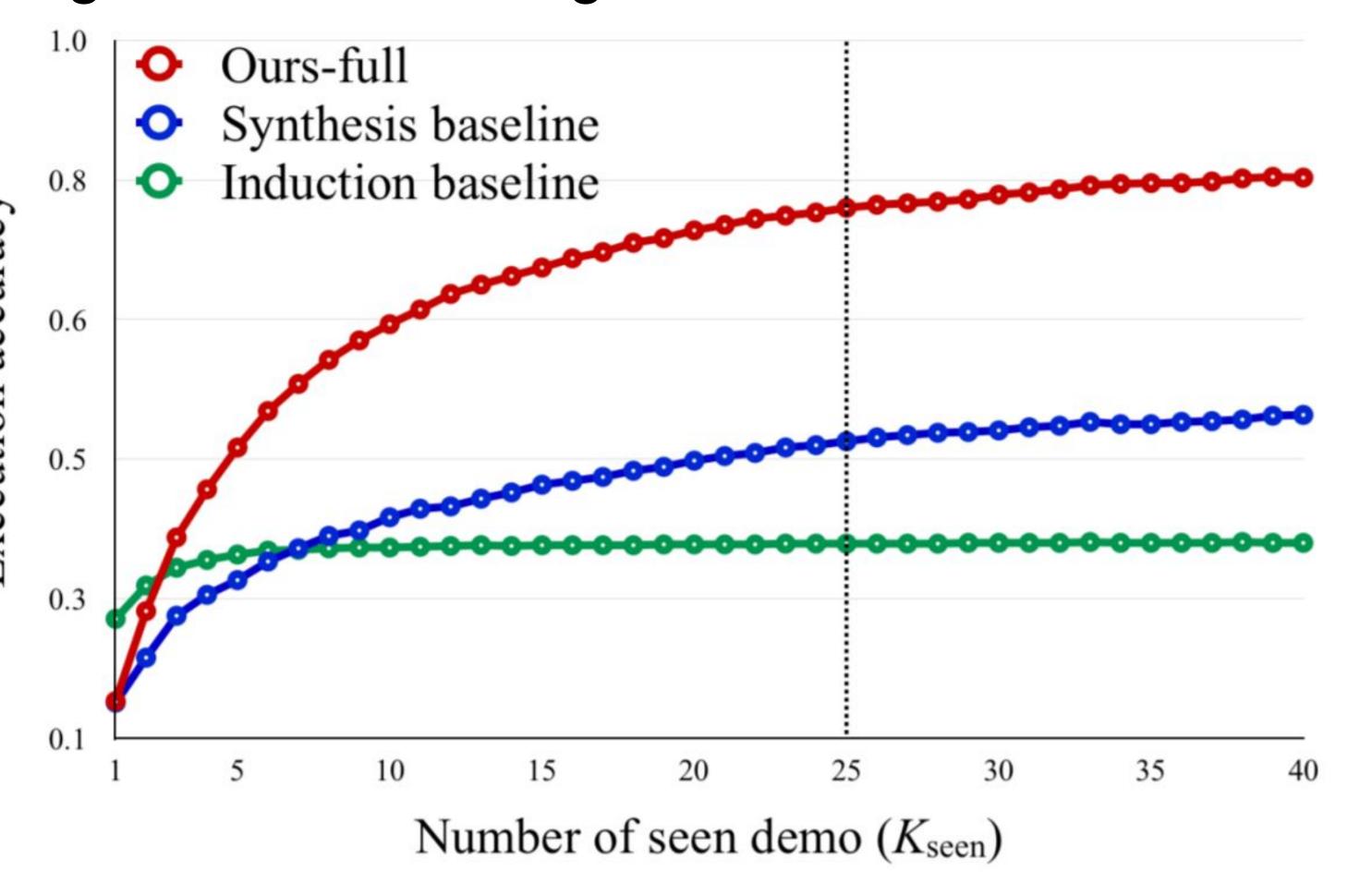### ❑ Advantage of explicit modeling of program
❑ Effect of summarizer / multi-task loss

- Karel

Methods	Execution	Program	Sequence
Induction baseline	62.8% (69.1%)	-	-
Synthesis baseline	64.1%	42.4%	35.7%
+ summarizer (ours)	68.6%	45.3%	38.3%
+ multi-task loss (ours-full)	**72.1%**	**48.9%**	**41.0%**

- Vizdoom

Methods	Execution	Program	Sequence
Induction baseline	35.1% (60.6%)	-	-
Synthesis baseline	48.2%	39.9%	33.1%
Ours-full	**78.4%**	**62.5%**	**53.2%**

❑ Generalization to more / less demonstrations
- Training with 25 demos, generalization to more / less demos

Ours-full | Synthesis baseline | Induction baseline

❑ Effect of summarizer with more demonstrations
- Advantage of summarizer is bigger with more demonstrations

Methods	k=3	k=5	k=10
Synthesis baseline	58.5%	60.1%	64.1%
+ summarizer (ours)	**60.6%**	**63.1%**	**68.6%**
Improvement	2.1%	3.0%	4.5%

❑ If-else experiment - inferring condition
- Single IFELSE statement with 1 condition, 2 different actions
- Inferring correct condition is essential

Methods	Execution	Program	Sequence
Induction baseline	26.5% (83.1%)	-	-
Synthesis baseline	59.9%	44.4%	36.1%
Ours-full	**89.4%**	**69.1%**	**58.8%**

Neural Program Synthesis from Diverse Demonstration Videos

University of Southern California[1]   POSTECH[2]
POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Shao-Hua Sun[*1], Hyeonwoo Noh[*2], Sriram Somasundaram[1], Joseph Lim[1]
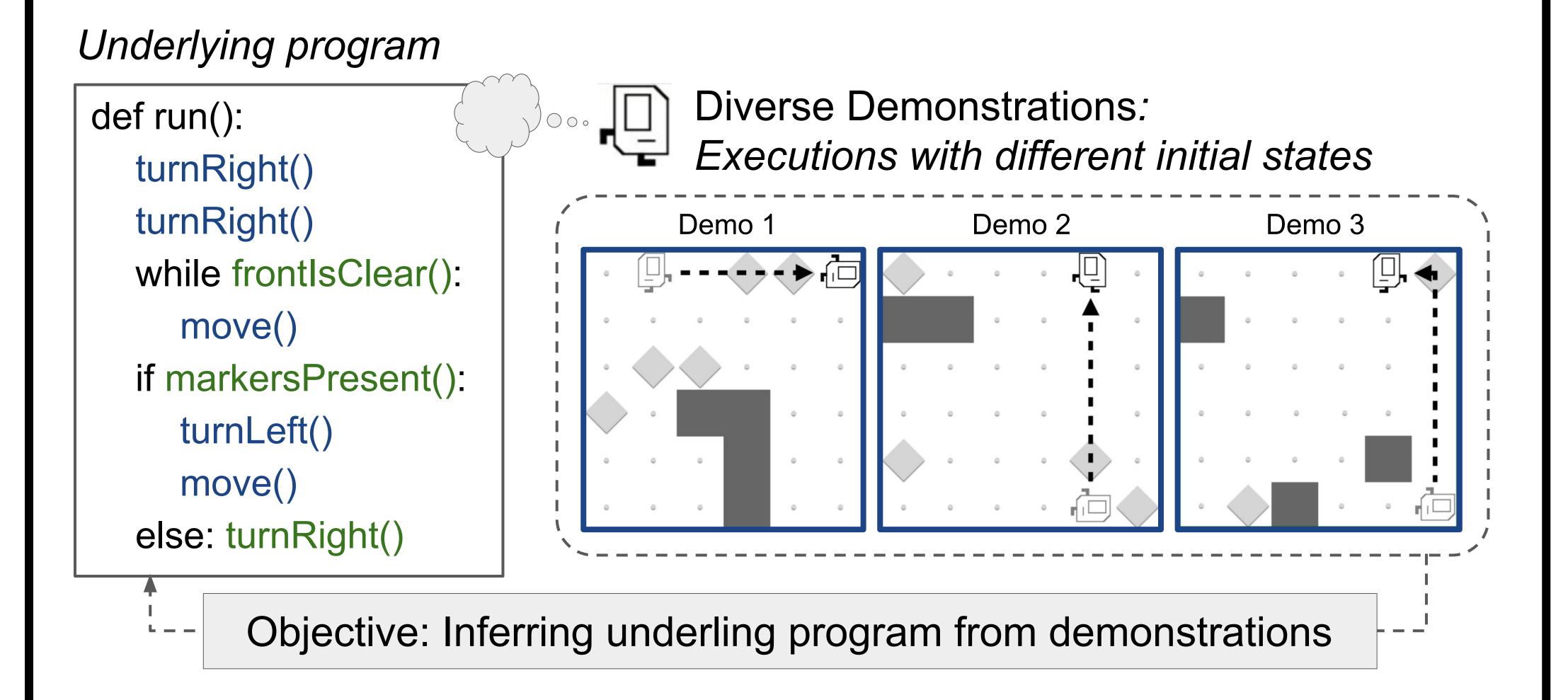
shaohuas@usc.edu   shgusdngogo@postech.ac.kr   sriramso@usc.edu   limjj@usc.edu

https://shaohua0116.github.io/demo2program/

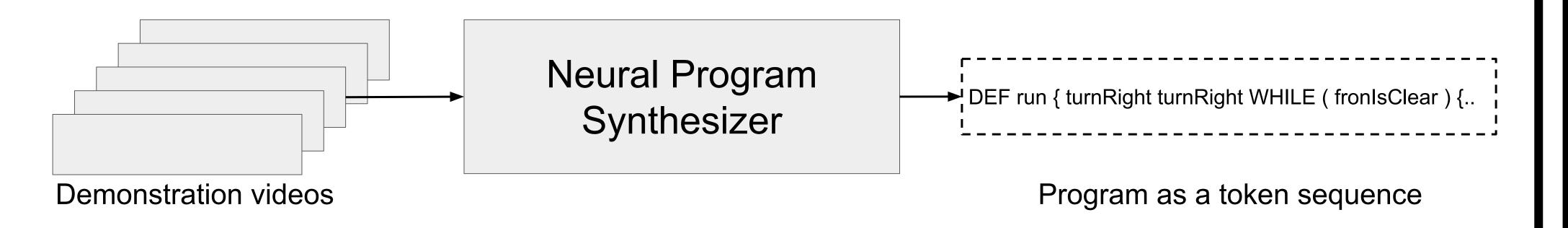## Problem Statement

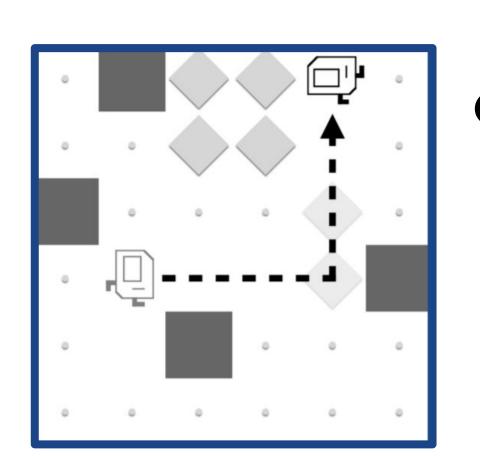Q: Could we infer underlying program of a behavior from diverse demonstration videos?

Underlying program

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

Diverse Demonstrations:
Executions with different initial states

Demo 1   Demo 2   Demo 3

Objective: Inferring underling program from demonstrations

❑ Our approach: Neural Program Synthesis

Demonstration videos → Neural Program Synthesizer → DEF run (turnRight turnRight WHILE (frontIsClear) {..
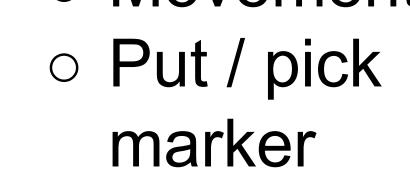
Program as a token sequence

❑ Contribution

● Program synthesis for explicit modeling of underlying programs
● Architecture for summarizing diverse demonstrations
● Auxiliary objectives enhancing program synthesis

Experimental Setting

● *Karel*:
  ○ Movement
  ○ Put / pick marker

● *Vizdoom*:
  ○ Movement
  ○ Kill monster
  ○ Select weapon

❑ Induction baseline (Few-shot imitation)

● Implicit modeling of programs by learning to imitate

Demonstration videos → Induction baseline → move turnRight move move turnLeft putMarker ...

Implicit model of program — → Environment ← -- New initial state

Take actions

❑ Synthesis baseline

● Program synthesis without summarizer / multi-task loss

Proposed Model

❑ Overall architecture


```
def run():
    move()
    if leftIsClear():
        turnLeft()
    REPEAT R=5:
        turnRight()
    if MarkersPresent():
        pickupMarker()
    else:
        move()
```
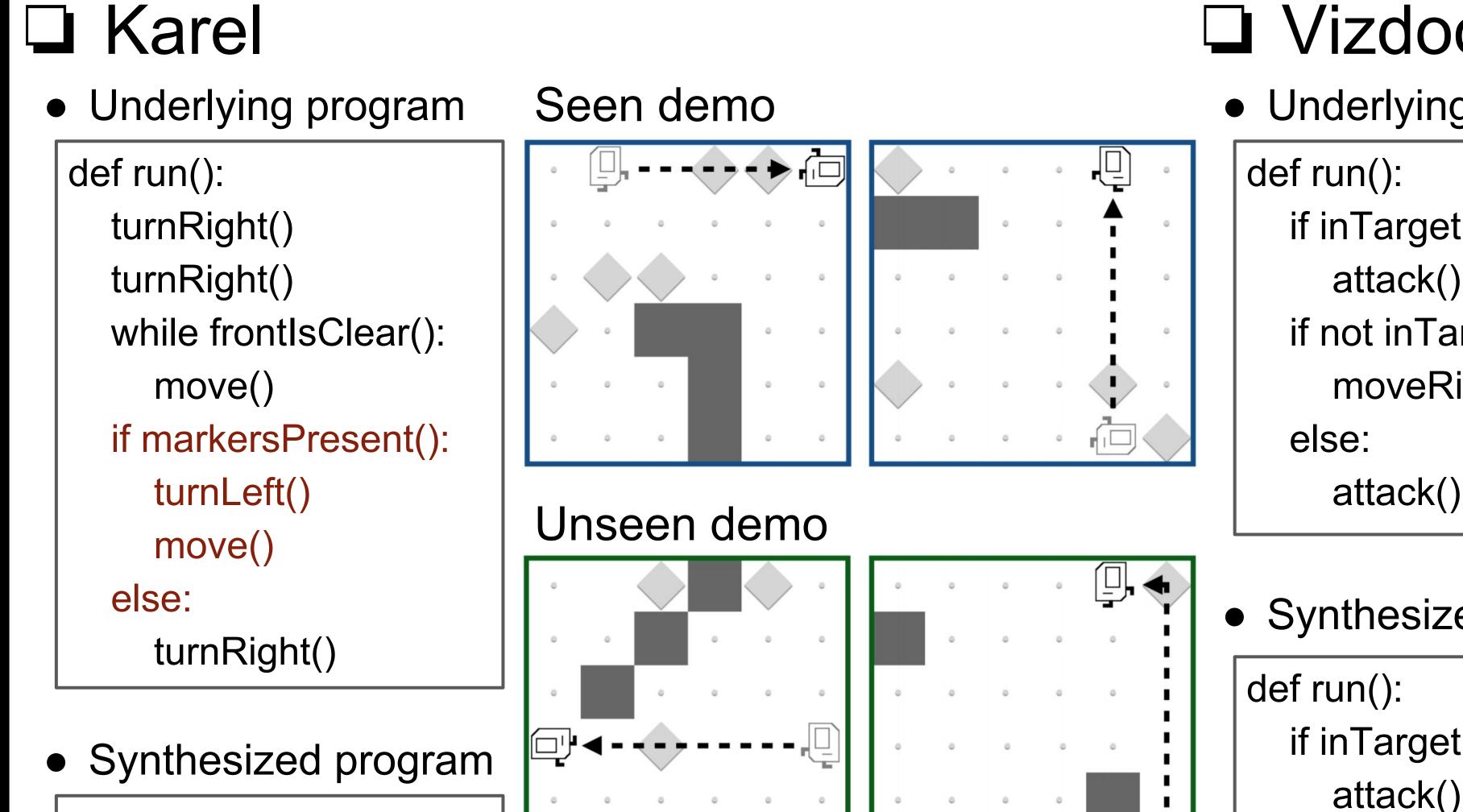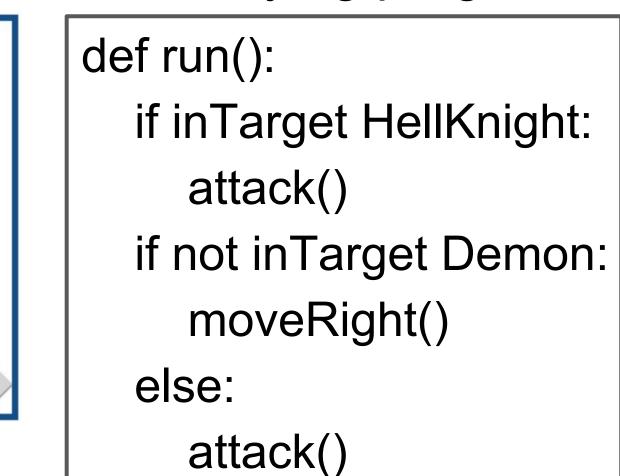
❑ Summarizer module

● *Summarizer module*:
  ○ Inferring actions and underlying conditions from demonstrations

● *Multi-task loss*:
  ○ Learning representation for inferring action / perception from demonstrations

Qualitative Results

❑ Karel

● Underlying program

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

● Synthesized program

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

Seen demo

Unseen demo

❑ Vizdoom

● Underlying program

```
def run():
    if inTarget HellKnight:
        attack()
    if not inTarget Demon:
        moveRight()
    else:
        attack()
```

● Synthesized program

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

Demo 1   Demo 2

inTarget HellKnight   inTarget HellKnight and inTarget Demon

inTarget HellKnight → attack()   inTarget HellKnight → attack()

not inTarget Demon → moveRight()   inTarget Demon → attack()

Experimental Results

❑ Advantage of explicit modeling of program
❑ Effect of summarizer / multi-task loss

● Karel

Methods	Execution	Program	Sequence
Induction baseline	62.8% (69.1%)	-	-
Synthesis baseline	64.1%	42.4%	35.7%
+ summarizer (ours)	68.6%	45.3%	38.3%
+ multi-task loss (ours-full)	**72.1%**	**48.9%**	**41.0%**

● Vizdoom

Methods	Execution	Program	Sequence
Induction baseline	35.1% (60.6%)	-	-
Synthesis baseline	48.2%	39.9%	33.1%
Ours-full	**78.4%**	**62.5%**	**53.2%**

❑ Generalization to more / less demonstrations

● Training with 25 demos, generalization to more / less demos

❑ Effect of summarizer with more demonstrations

● Advantage of summarizer is bigger with more demonstrations

Methods	k=3	k=5	k=10
Synthesis baseline	58.5%	60.1%	64.1%
+ summarizer (ours)	**60.6%**	**63.1%**	**68.6%**
Improvement	2.1%	3.0%	4.5%

❑ If-else experiment - inferring condition

● Single IFELSE statement with 1 condition, 2 different actions
● Inferring correct condition is essential

Methods	Execution	Program	Sequence
Induction baseline	26.5% (83.1%)	-	-
Synthesis baseline	59.9%	44.4%	36.1%
Ours-full	**89.4%**	**69.1%**	**58.8%**

Neural Program Synthesis from Diverse Demonstration Videos

USC[1]

POSTECH[2]
POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Shao-Hua Sun[*1], Hyeonwoo Noh[*2], Sriram Somasundaram[1], Joseph Lim[1]

shaohuas@usc.edu shgusdngogo@postech.ac.kr   sriramso@usc.edu   limjj@usc.edu

ICML | 2018
Thirty-fifth International Conference on Machine Learning
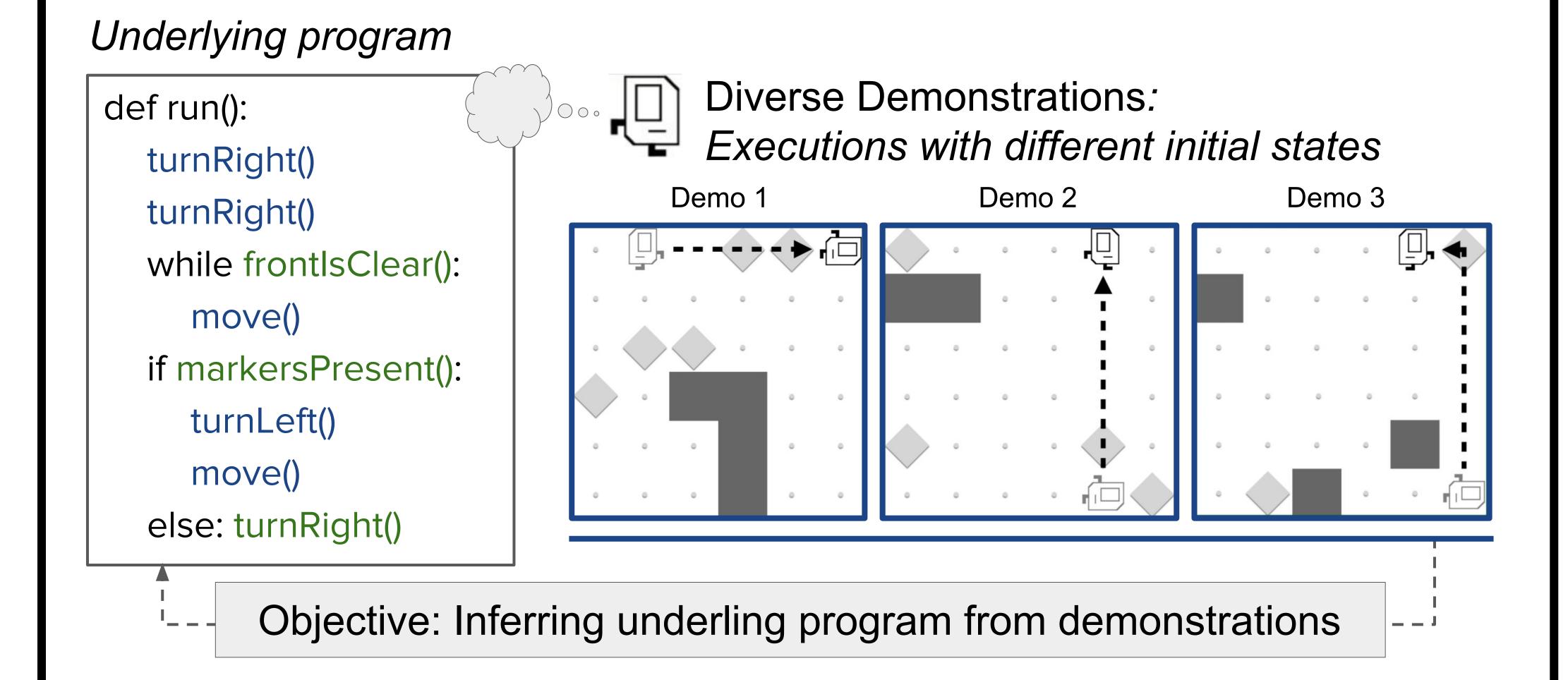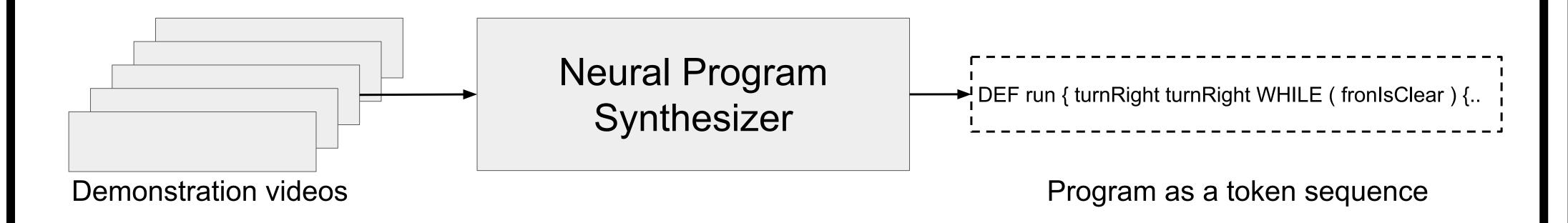
https://shaohua0116.github.io/demo2program/

Problem Statement

Q: Could we infer underlying program of a behavior from diverse demonstration videos?
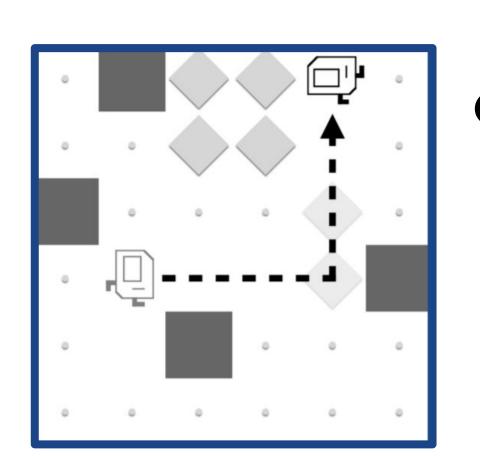
Underlying program

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

Diverse Demonstrations:
Executions with different initial states

Demo 1   Demo 2   Demo 3

Objective: Inferring underling program from demonstrations

❑ Our approach: Neural Program Synthesis

Demonstration videos → Neural Program Synthesizer → DEF run (turnRight turnRight WHILE (frontIsClear) {..
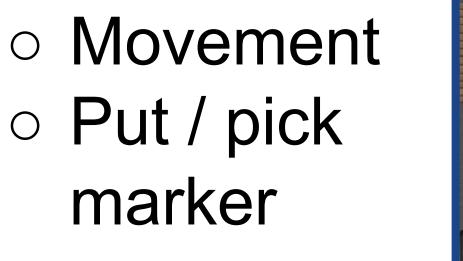
Program as a token sequence

❑ Contribution
- Program synthesis for explicit modeling of underlying programs
- Architecture for summarizing diverse demonstrations
- Auxiliary objectives enhancing program synthesis

Experimental Setting

- *Karel*:
  - Movement
  - Put / pick marker
- *Vizdoom*:
  - Movement
  - Kill monster
  - Select weapon

❑ Induction baseline (Few-shot imitation)
- Implicit modeling of programs by learning to imitate

Demonstration videos → Induction baseline → Environment → New initial state
Implicit model of program
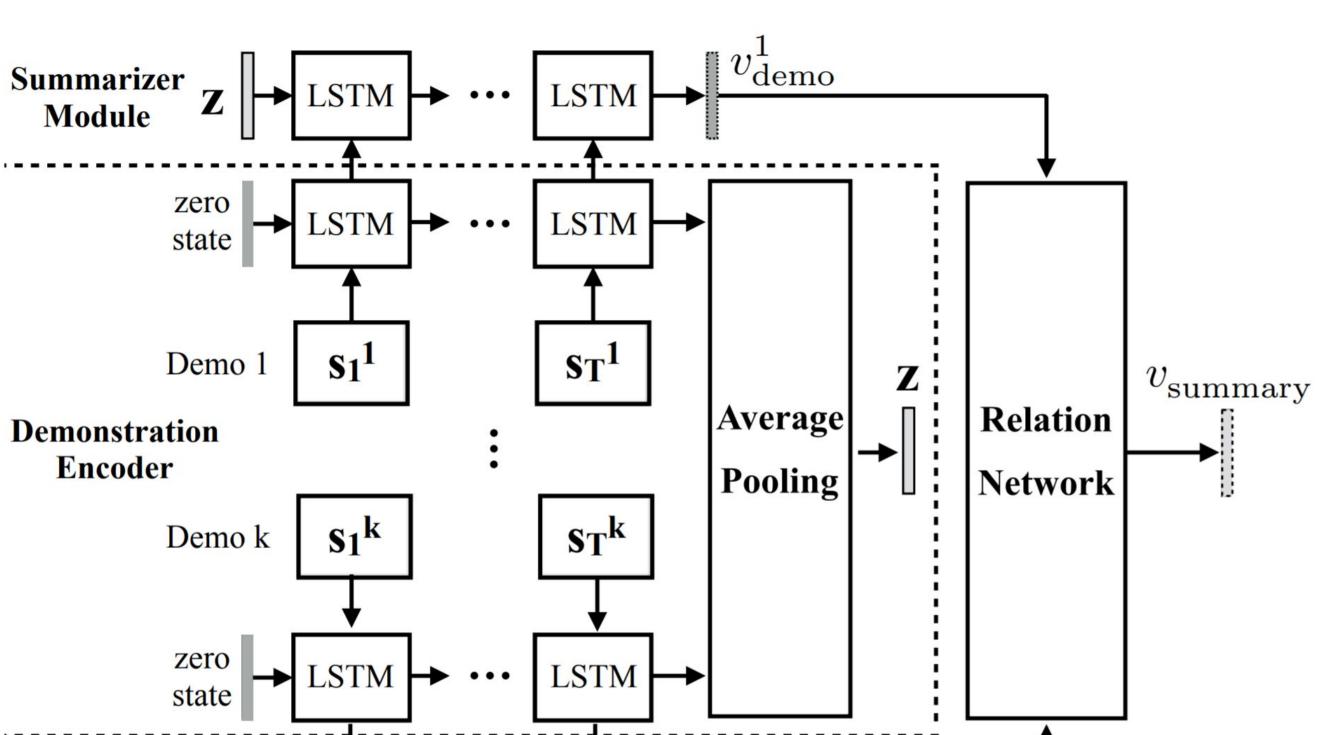move turnRight move move turnLeft putMarker ...
Take actions

❑ Synthesis baseline
- Program synthesis without summarizer / multi-task loss

Proposed Model

❑ Overall architecture

Demo 1 → Demo Encoder → Summarizer Module → $v_{summary}$ → Program Decoder → Program
Demo 2 → Demo Encoder
Demo k → Demo Encoder

$v_{demo}^1, v_{demo}^2, v_{demo}^k$ → Action Decoder

```
def run() move() ··· else move()
```

```
move()  turnLeft()  ···  pickupMarker()
move()  turnLeft()  ···  move()
move()  turnRight()  ···  pickupMarker()
```

Perception Decoder

```
frontIsClear()     yes
rightIsClear()     no      ···
MarkersPresent()   yes
```

```
def run():
    move()
    if leftIsClear():
        turnLeft()
    REPEAT R=5:
        turnRight()
    if MarkersPresent():
        pickupMarker()
    else:
        move()
```

❑ Summarizer module

Summarizer Module: Z → LSTM ··· LSTM → v_{demo}^1
zero state → LSTM → LSTM
Demo 1: s_1^1 → s_T^1
Demonstration Encoder
Demo k: s_1^k → s_T^k
zero state → LSTM → LSTM
Z → LSTM → LSTM → v_{demo}^k

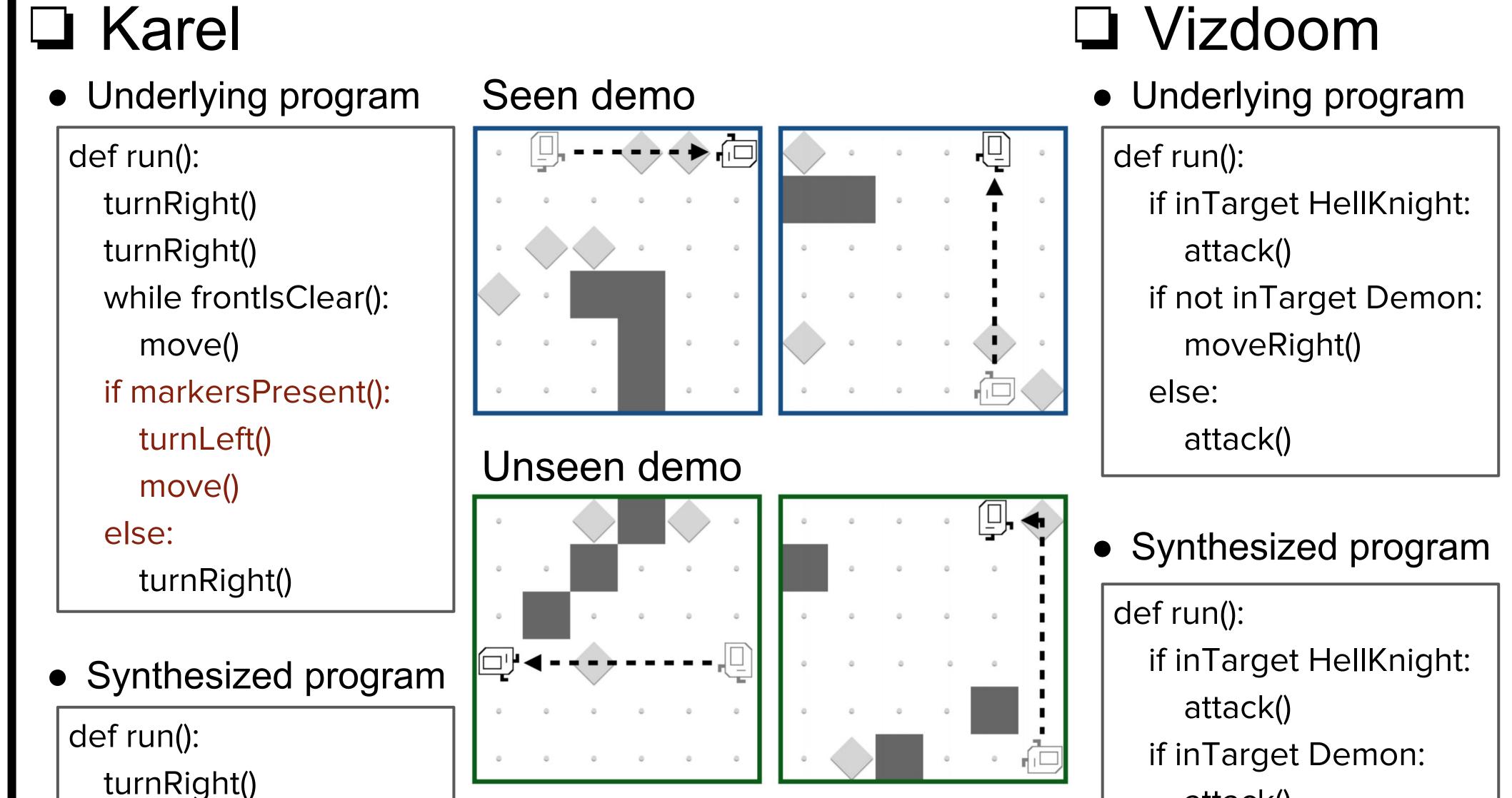Average Pooling → Z → Relation Network → $v_{summary}$

- *Summarizer module*:
  - Inferring actions and underlying conditions from demonstrations
- *Multi-task loss*:
  - Learning representation for inferring action / perception from demonstrations
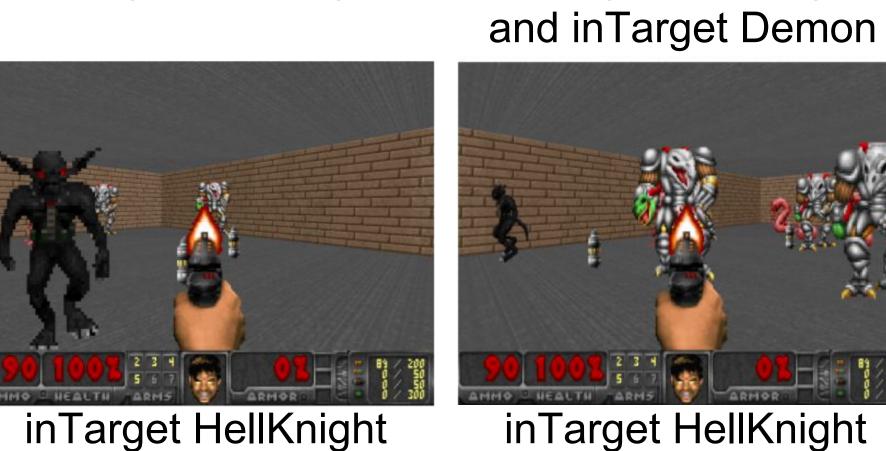
Qualitative Results

❑ Karel
- Underlying program

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

Seen demo

Unseen demo

- Synthesized program

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

❑ Vizdoom
- Underlying program

```
def run():
    if inTarget HellKnight:
        attack()
    if not inTarget Demon:
        moveRight()
    else:
        attack()
```

- Synthesized program

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

Demo 1 | Demo 2

inTarget HellKnight | inTarget HellKnight and inTarget Demon

inTarget HellKnight → attack() | inTarget HellKnight → attack()

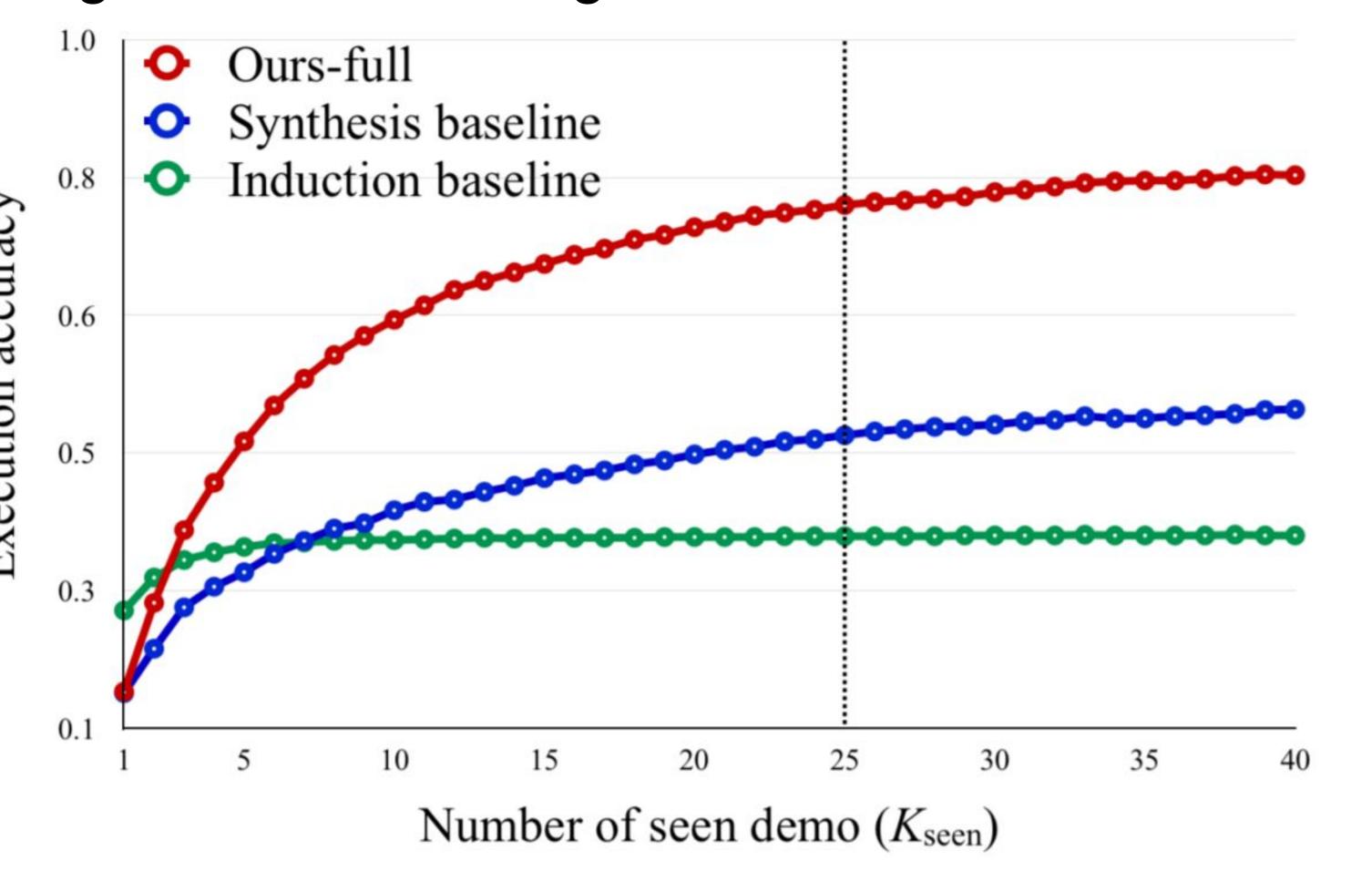not inTarget Demon → moveRight() | inTarget Demon → attack()

Experimental Results

❑ Advantage of explicit modeling of program
❑ Effect of summarizer / multi-task loss
- Karel

Methods	Execution	Program	Sequence
Induction baseline	62.8% (69.1%)	-	-
Synthesis baseline	64.1%	42.4%	35.7%
+ summarizer (ours)	68.6%	45.3%	38.3%
+ multi-task loss (ours-full)	**72.1%**	**48.9%**	**41.0%**

- Vizdoom

Methods	Execution	Program	Sequence
Induction baseline	35.1% (60.6%)	-	-
Synthesis baseline	48.2%	39.9%	33.1%
Ours-full	**78.4%**	**62.5%**	**53.2%**

❑ Generalization to more / less demonstrations
- Training with 25 demos, generalization to more / less demos

Ours-full / Synthesis baseline / Induction baseline
Execution accuracy vs Number of seen demo (K_{seen})

❑ Effect of summarizer with more demonstrations
- Advantage of summarizer is bigger with more demonstrations

Methods	k=3	k=5	k=10
Synthesis baseline	58.5%	60.1%	64.1%
+ summarizer (ours)	**60.6%**	**63.1%**	**68.6%**
Improvement	2.1%	3.0%	4.5%

❑ If-else experiment - inferring condition
- Single IFELSE statement with 1 condition, 2 different actions
- Inferring correct condition is essential

Methods	Execution	Program	Sequence
Induction baseline	26.5% (83.1%)	-	-
Synthesis baseline	59.9%	44.4%	36.1%
Ours-full	**89.4%**	**69.1%**	**58.8%**

Neural Program Synthesis from Diverse Demonstration Videos

Shao-Hua Sun*[1], Hyeonwoo Noh*[2], Sriram Somasundaram[1], Joseph Lim[1]
shaohuas@usc.edu shgusdngogo@postech.ac.kr   sriramso@usc.edu   limjj@usc.edu

USC[1]
POSTECH[2]
POHANG UNIVERSITY OF SCIENCE AND T...

Problem Statement

Q: Could we infer underlying program of a behavior from diverse demonstration videos?

Underlying program

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

Diverse Demonstrations:
Executions with different initial states

Demo 1   Demo 2   Demo 3

Objective: Inferring underling program from demonstrations

❏ Our approach: Neural Program Synthesis

Demonstration videos → Neural Program Synthesizer → DEF run { turnRight turnRight WHILE (frontIsClear) {..

Program as a token sequence

Experimental Setting

❏ Environments
- Common DSL with different action / perception
  - DSL (domain specific language) including IF, WHILE, IFELSE, REPEAT, binary perceptions and NOT operator

  - *Karel*:
    - Movement
    - Put / pick marker
  - *Vizdoom*:
    - Movement
    - Perceive / kill monster

❏ Induction baseline (Few-shot imitation)
- Implicit modeling of programs by learning to imitate

Implicit model of program → Induction baseline → Environment ← New initial state
Demonstration videos   move turnRight move move turnLeft putMarker ...
                                                      Take actions

❏ Synthesis baseline
- Program synthesis without summarizer / multi-task loss

Proposed Model

❏ Overall architecture

Demo 1 → Demo Encoder
Demo 2 → Demo Encoder
Demo k → Demo Encoder
→ Summarizer Module → v_{summary} → Program Decoder

def run() move() ··· else move()   Program

```
def run():
    move()
    if leftIsClear():
        turnLeft()
    REPEAT R=5:
        turnRight()
    if MarkersPresent():
        pickupMarker()
    else:
        move()
```

v_{demo}^1, v_{demo}^2, v_{demo}^k

Action Decoder
```
move()   turnLeft()  ···  pickupMarker()
move()   turnLeft()   ···   move()
move()   turnRight()  ···  pickupMarker()
```

Perception Decoder
```
frontIsClear()    yes
rightIsClear()    no
MarkersPresent()  yes
```

❏ Summarizer module

- *Summarizer module*:
  - Inferring actions and underlying conditions from demonstrations
- *Multi-task loss*:
  - Learning representation for inferring action / perception from demonstrations

Evaluation Result

❏ Evaluation Metric
- Execution accuracy: accuracy of execution result on unseen initial states
- Program accuracy: accuracy based on program behavior space measured by enumerating program syntax
- Sequence accuracy: accuracy of synthesized program token sequence

❏ If-else experiment - inferring condition

Methods	Execution	Program	Sequence
Induction baseline	26.5% (83.1%)	-	-
Synthesis baseline	59.9%	44.4%	36.1%
Ours-full	**89.4%**	**69.1%**	**58.8%**

❏ Inferring conditions from diverging demonstrations

Sequence of programs - qualitative result - by sri

❏ Generalization to more / less demonstrations

Ours-full
Synthesis baseline
Induction baseline

Execution accuracy vs. Number of seen demo (K_{seen})

Neural Program Synthesis from Diverse Demonstration Videos

USC[1]
POSTEC[2]
POHANG UNIVERSITY OF SCIENCE AND T...

Shao-Hua Sun[*1], Hyeonwoo Noh[*2], Sriram Somasundaram[1], Joseph Lim[1]
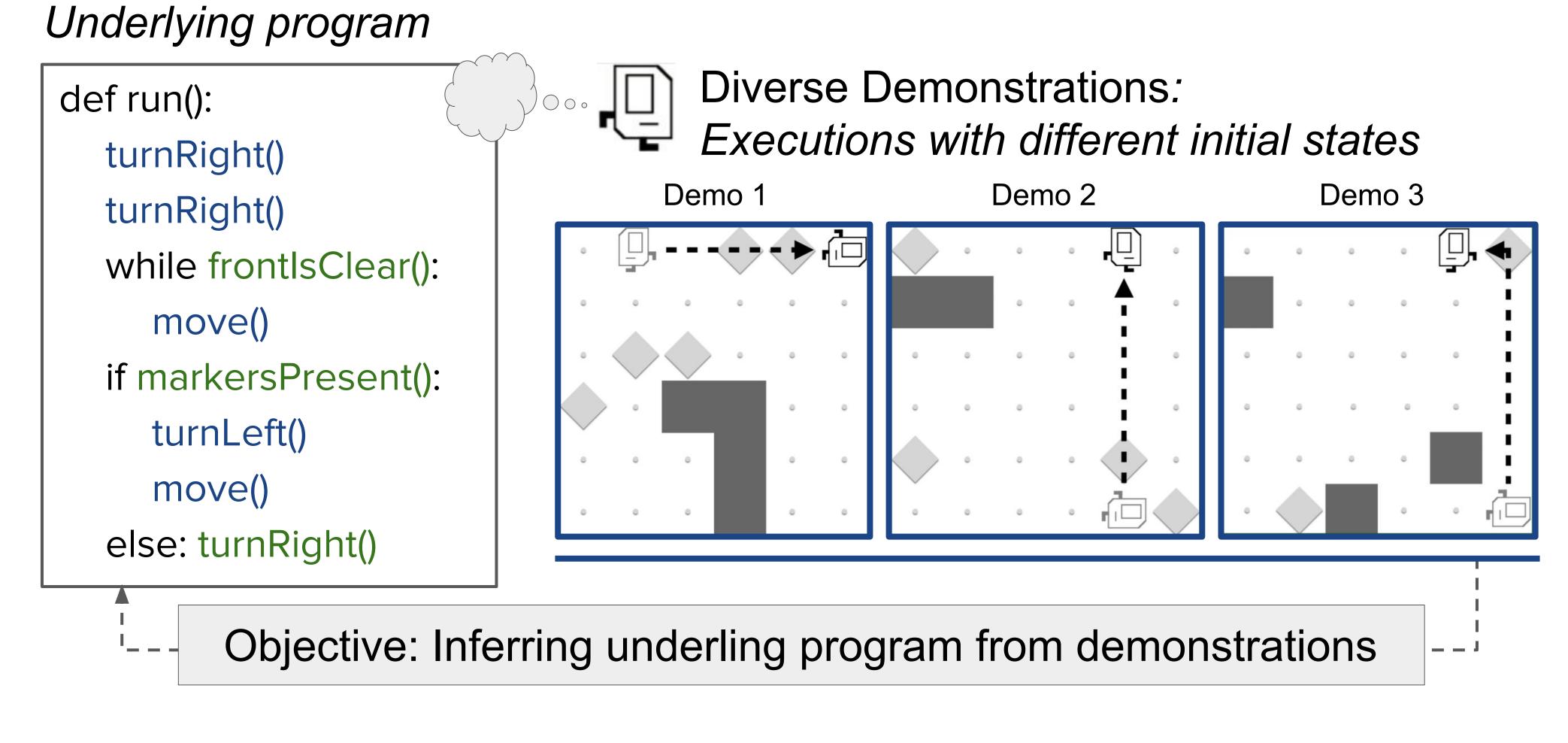
shaohuas@usc.edu   shgusdngogo@postech.ac.kr   sriramso@usc.edu   limjj@usc.edu

Problem Statement

Q: Could we infer underlying program of a behavior from diverse demonstration videos?
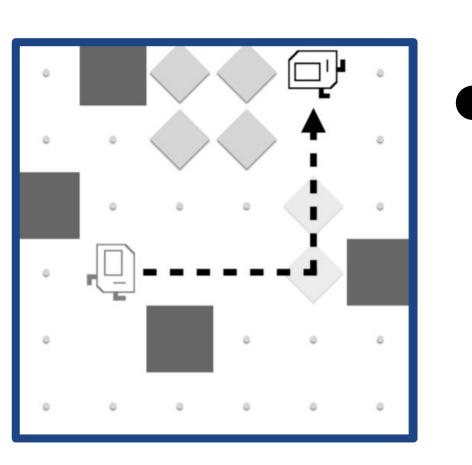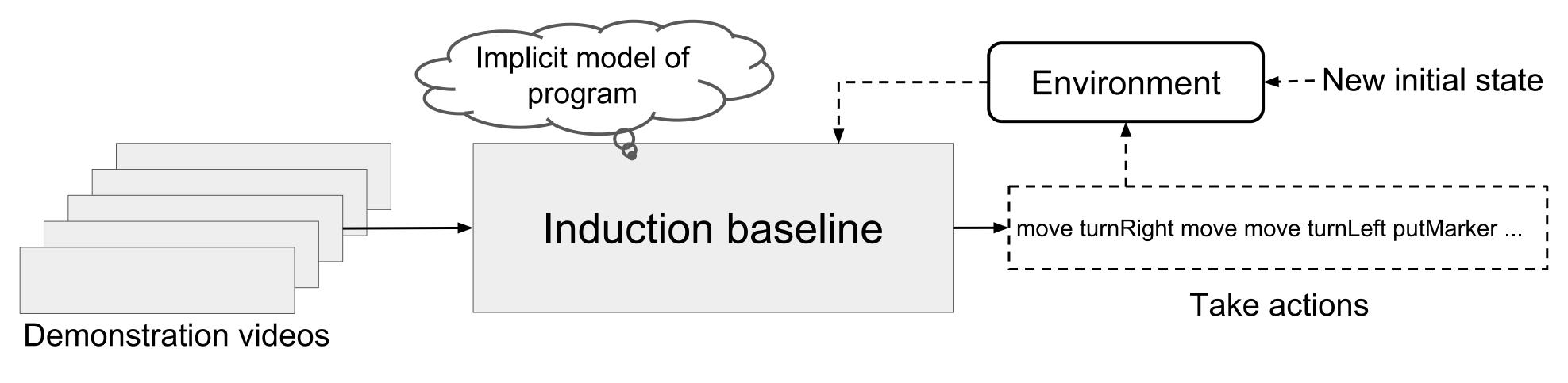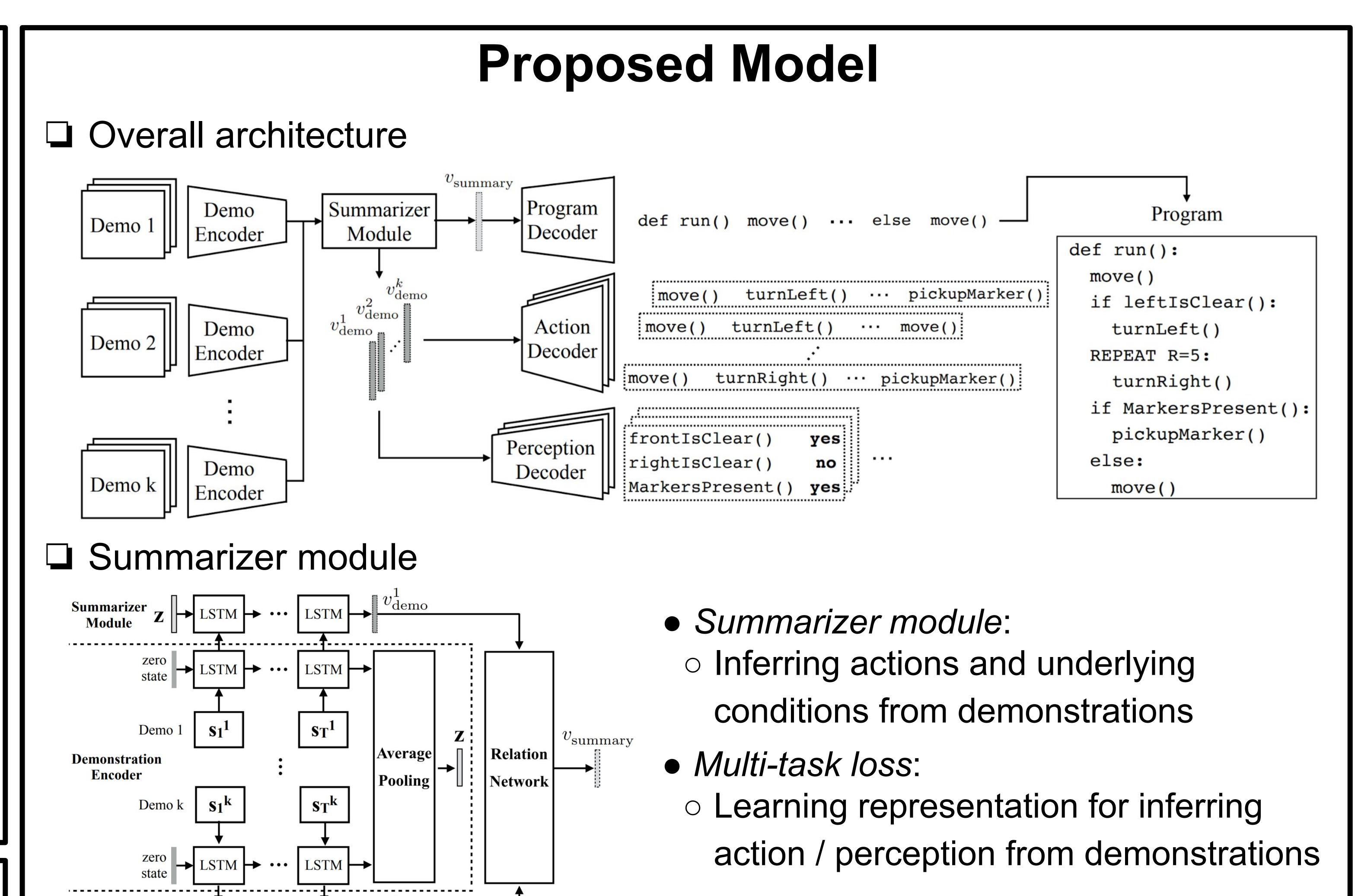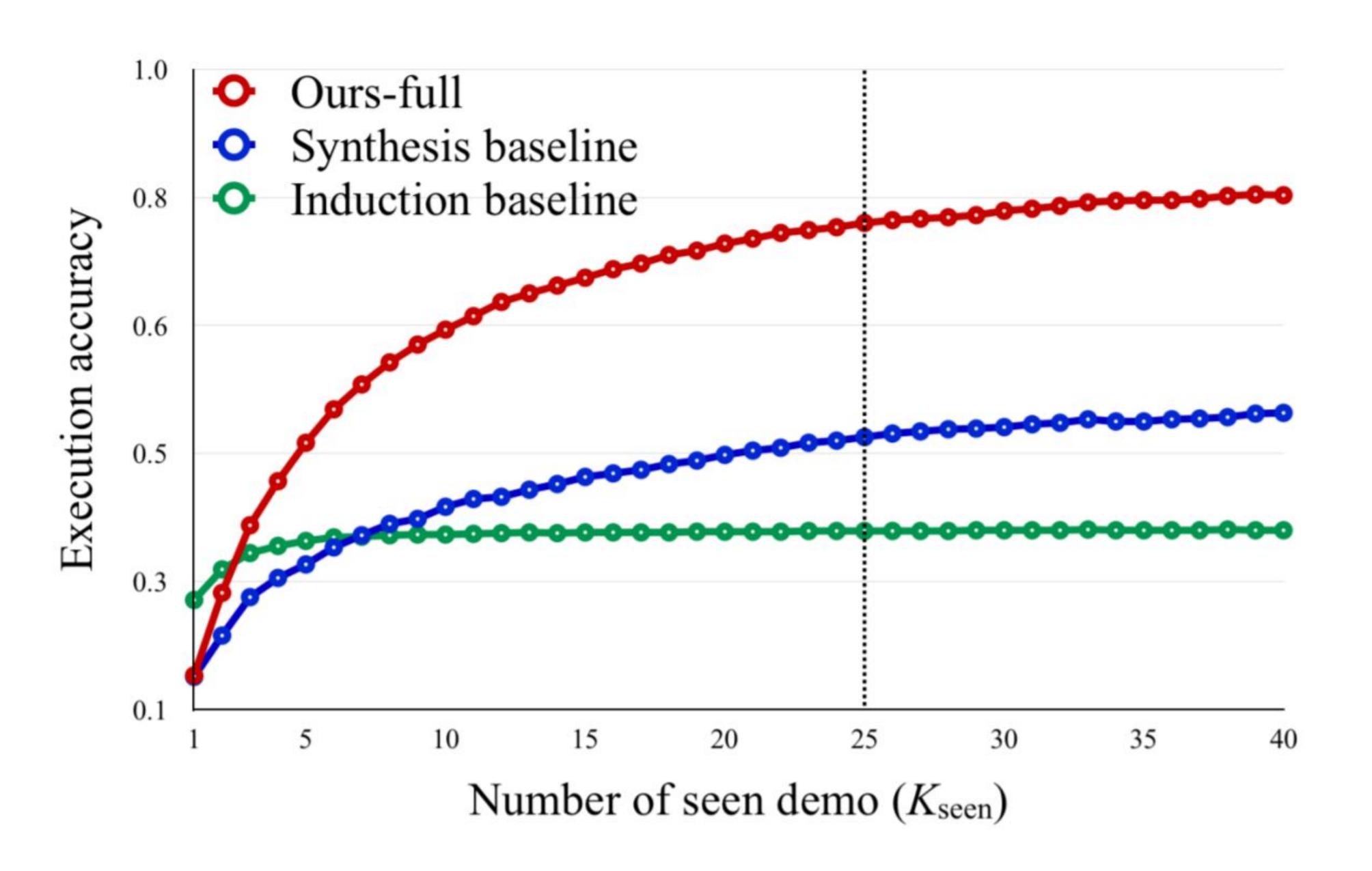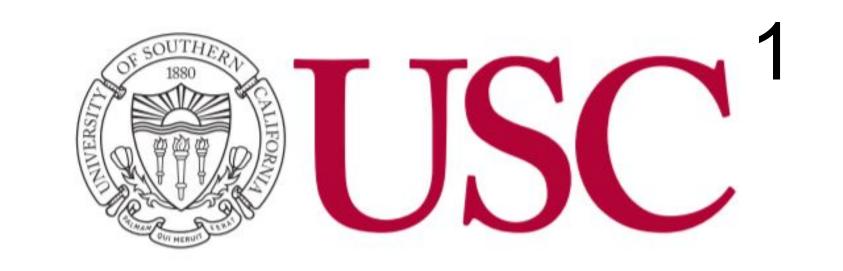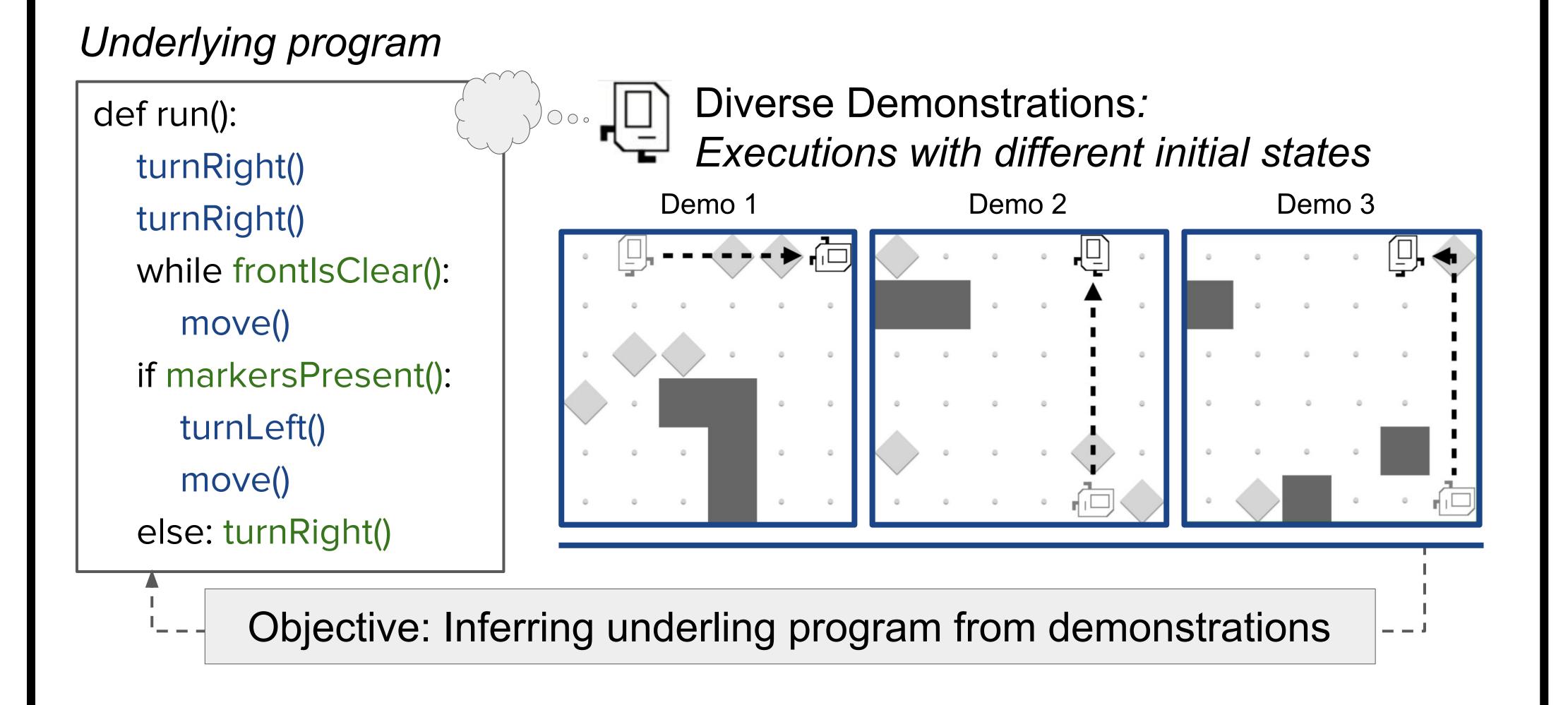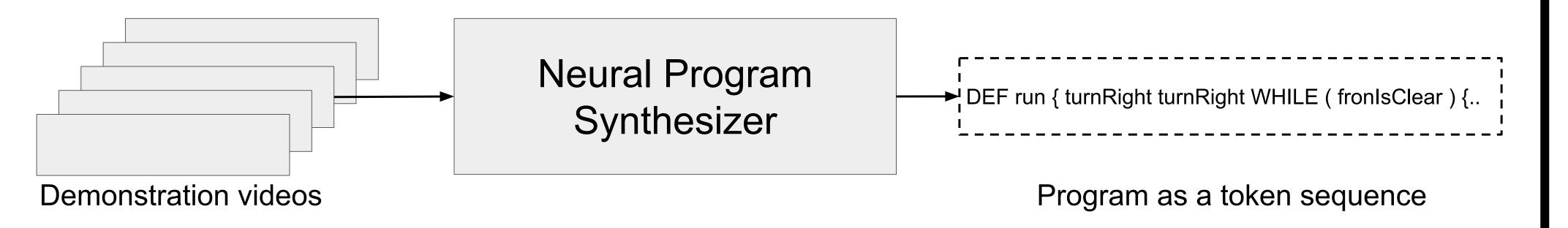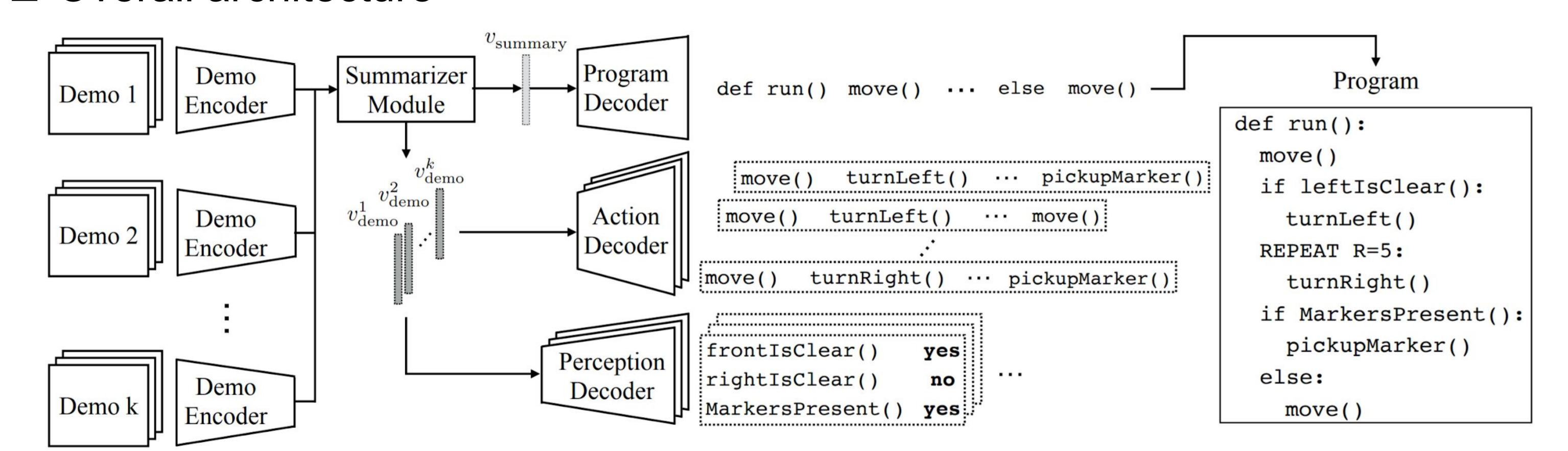
Underlying program

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

Diverse Demonstrations:
Executions with different initial states

Demo 1   Demo 2   Demo 3

Objective: Inferring underling program from demonstrations

❏ Our approach: Neural Program Synthesis

Demonstration videos → Neural Program Synthesizer → DEF run { turnRight turnRight WHILE (fronIsClear) {..
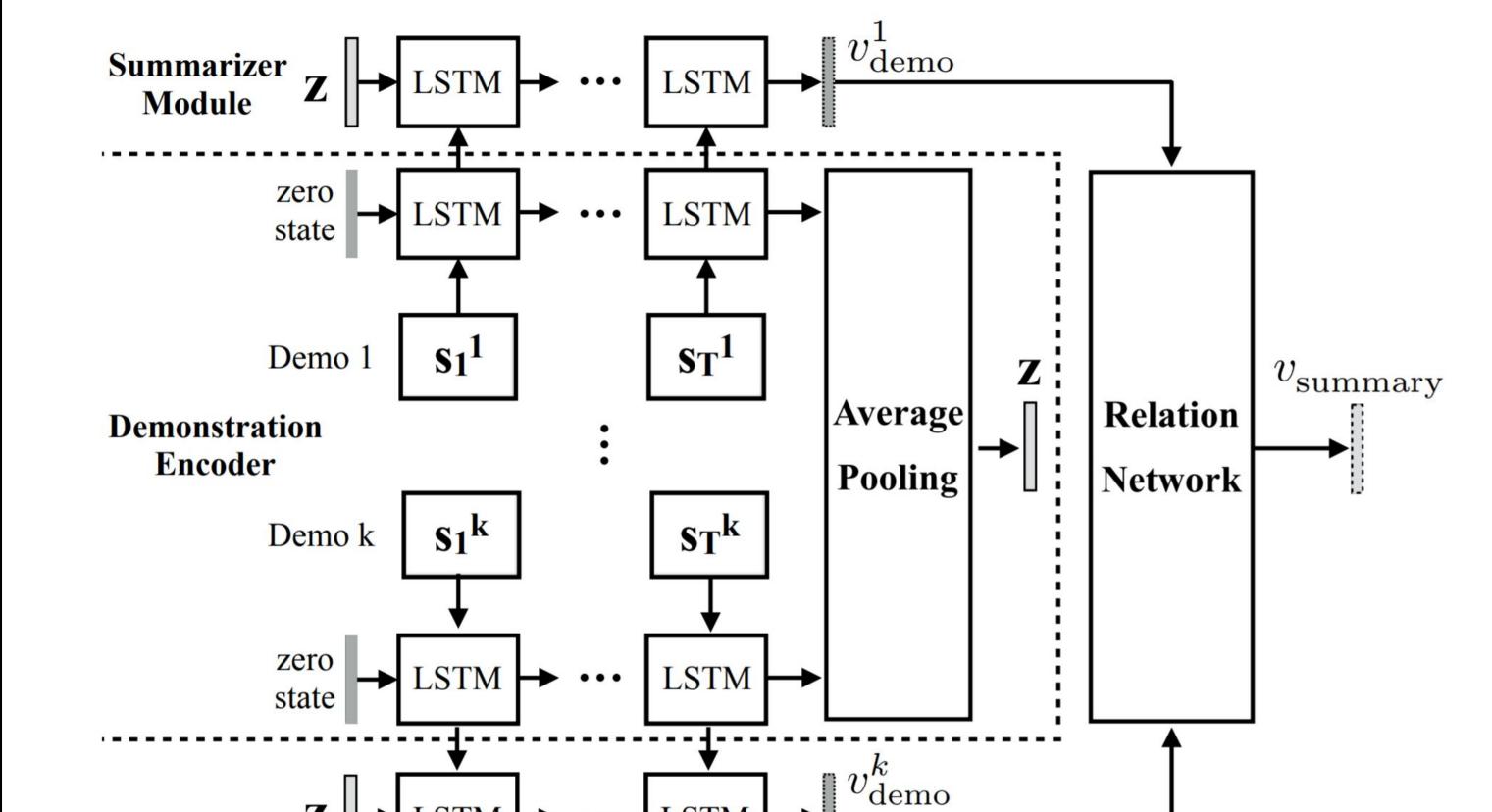
Program as a token sequence

❏ Contribution
- Program synthesis for explicit modeling of underlying programs
- Architecture for summarizing diverse demonstrations
- Auxiliary objectives enhancing program synthesis

Experimental Setting

- *Karel*:
  - Movement
  - Put / pick marker
- *Vizdoom*:
  - Movement
  - Perceive / kill monster

❏ Induction baseline (Few-shot imitation)
- Implicit modeling of programs by learning to imitate

Demonstration videos → Induction baseline → move turnRight move move turnLeft putMarker ...

Implicit model of program → Environment -- New initial state

Take actions

❏ Synthesis baseline
- Program synthesis without summarizer / multi-task loss

Proposed Model

❏ Overall architecture

Demo 1 → Demo Encoder → Summarizer Module → $v_{summary}$ → Program Decoder

Demo 2 → Demo Encoder → Action Decoder

Demo k → Demo Encoder → Perception Decoder

v_{demo}^1 v_{demo}^2 v_{demo}^k

```
def run()  move()  ···  else  move()        Program
```

```
move()   turnLeft()  ···  pickupMarker()
move()   turnLeft()  ···  move()
move()   turnRight() ···  pickupMarker()
```

```
frontIsClear()    yes
rightIsClear()    no      ···
MarkersPresent()  yes
```

```
def run():
    move()
    if leftIsClear():
        turnLeft()
    REPEAT R=5:
        turnRight()
    if MarkersPresent():
        pickupMarker()
    else:
        move()
```

❏ Summarizer module

Summarizer Module: **z** → LSTM ··· LSTM → v_{demo}^1

zero state → LSTM ··· LSTM

Demonstration Encoder: Demo 1 → s_1^1 ··· s_T^1

Demo k → s_1^k ··· s_T^k

zero state → LSTM ··· LSTM

z → LSTM ··· LSTM → v_{demo}^k

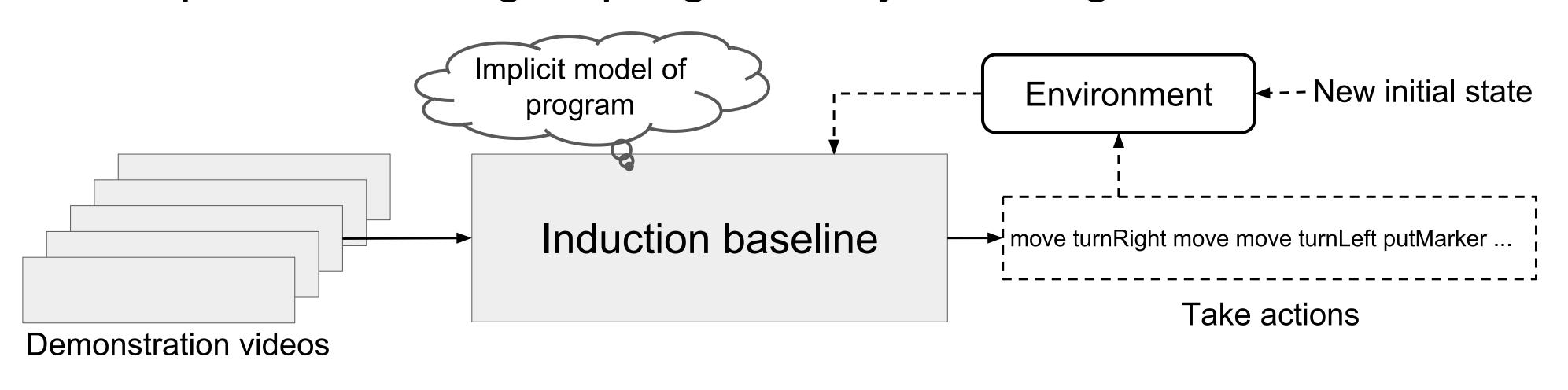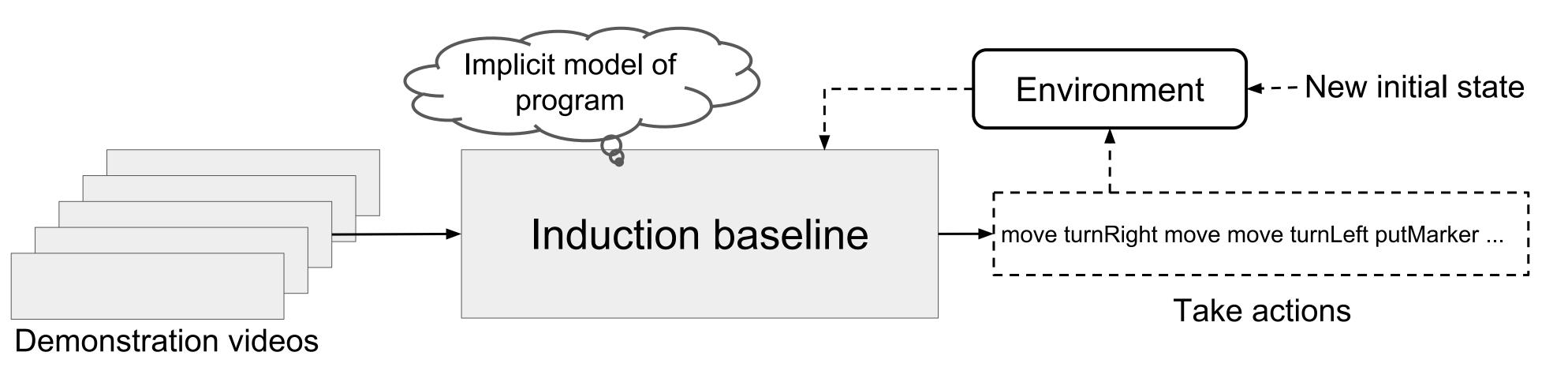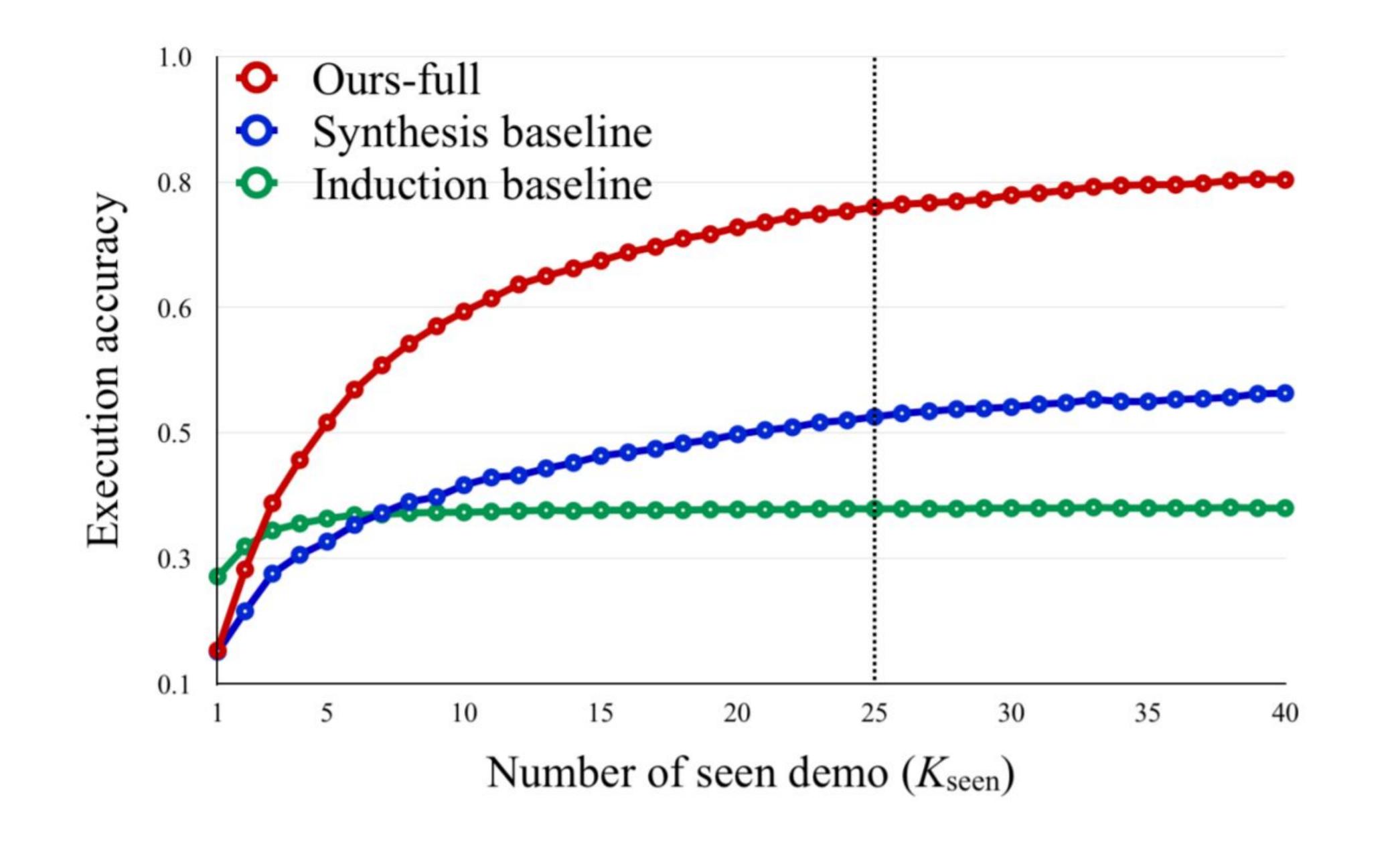Average Pooling → **Z** → Relation Network → $v_{summary}$

- *Summarizer module*:
  - Inferring actions and underlying conditions from demonstrations
- *Multi-task loss*:
  - Learning representation for inferring action / perception from demonstrations

➢ Effect of summarizer module:

Methods	k=3	k=5	k=10
Synthesis baseline	58.5%	60.1%	64.1%
+ summarizer (ours)	**60.6%**	**63.1%**	**68.6%**
Improvement	2.1%	3.0%	4.5%

❏ Inferring conditions from diverging demonstrations

Sequence of programs - qualitative result - by sri

Baselines

❏ Induction baseline (Few-shot imitation)
- Implicit modeling of programs by learning to imitate

Implicit model of program → Environment -- New initial state

Demonstration videos → Induction baseline → move turnRight move move turnLeft putMarker ...

Take actions
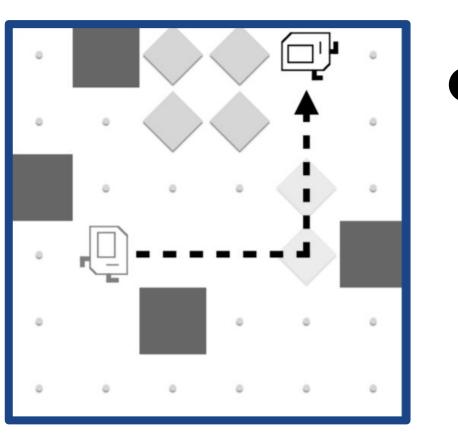
❏ Synthesis baseline
- Program synthesis without summarizer / multi-task loss

❏ If-else experiment - inferring condition

Methods	Execution	Program	Sequence
Induction baseline	26.5% (83.1%)	-	-
Synthesis baseline	59.9%	44.4%	36.1%
Ours-full	**89.4%**	**69.1%**	**58.8%**

❏ Generalization to more / less demonstrations

- Ours-full
- Synthesis baseline
- Induction baseline

Execution accuracy vs. Number of seen demo (K_{seen})

Neural Program Synthesis from Diverse Demonstration Videos

Shao-Hua Sun[*1], Hyeonwoo Noh[*2], Sriram Somasundaram[1], Joseph Lim[1]
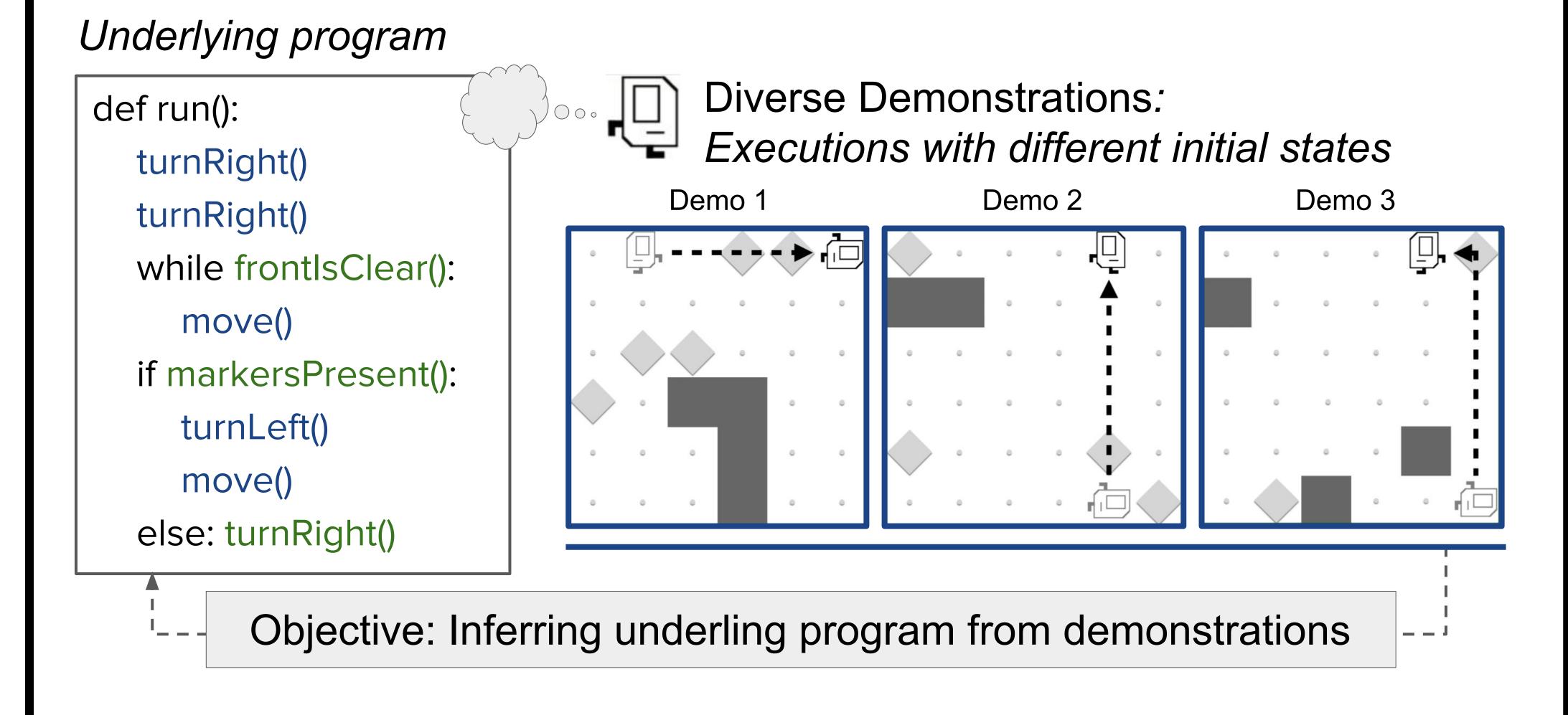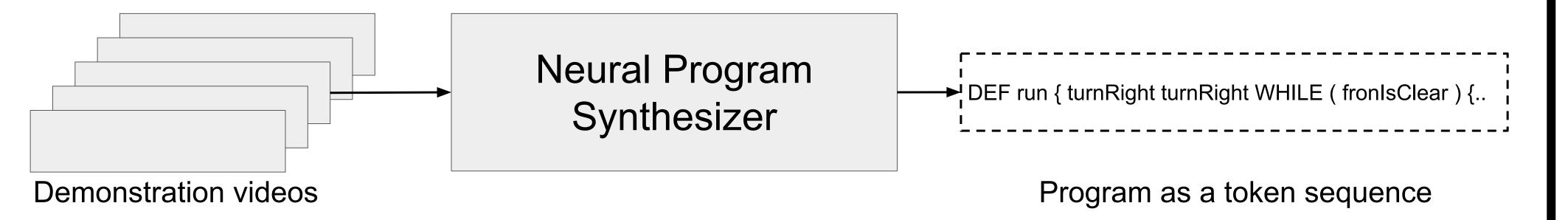shaohuas@usc.edu shgusdngogo@postech.ac.kr sriramso@usc.edu limjj@usc.edu

Problem Statement

Q: Could we infer underlying program of a behavior from diverse demonstration videos?

Underlying program

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

Diverse Demonstrations:
Executions with different initial states

Demo 1   Demo 2   Demo 3

Objective: Inferring underling program from demonstrations

❏ Our approach: Neural Program Synthesis

Demonstration videos → Neural Program Synthesizer → DEF run { turnRight turnRight WHILE (fronIsClear) {..

Program as a token sequence

❏ Contribution
- Program synthesis for explicit modeling of underlying programs
- Architecture for summarizing diverse demonstrations
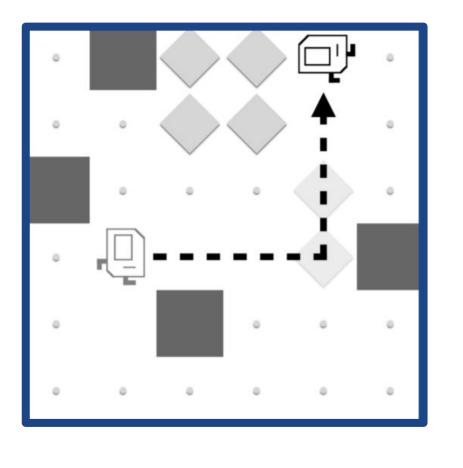- Auxiliary objectives enhancing program synthesis

Experimental Setting

Proposed Model

❏ Overall architecture


```
def run():
    move()
    if leftIsClear():
        turnLeft()
    REPEAT R=5:
        turnRight()
    if MarkersPresent():
        pickupMarker()
    else:
        move()
```

❏ Summarizer module

❏ Inferring conditions from diverging demonstrations

Sequence of programs - qualitative result - by sri

Baselines

❏ Induction baseline

❏ Synthesis baseline

❏ Proposed model

❏ If-else experiment - inferring condition

Methods	Execution	Program	Sequence
Induction baseline	26.5% (83.1%)	-	-
Synthesis baseline	59.9%	44.4%	36.1%
Ours-full	**89.4%**	**69.1%**	**58.8%**

❏ Generalization to more / less demonstrations