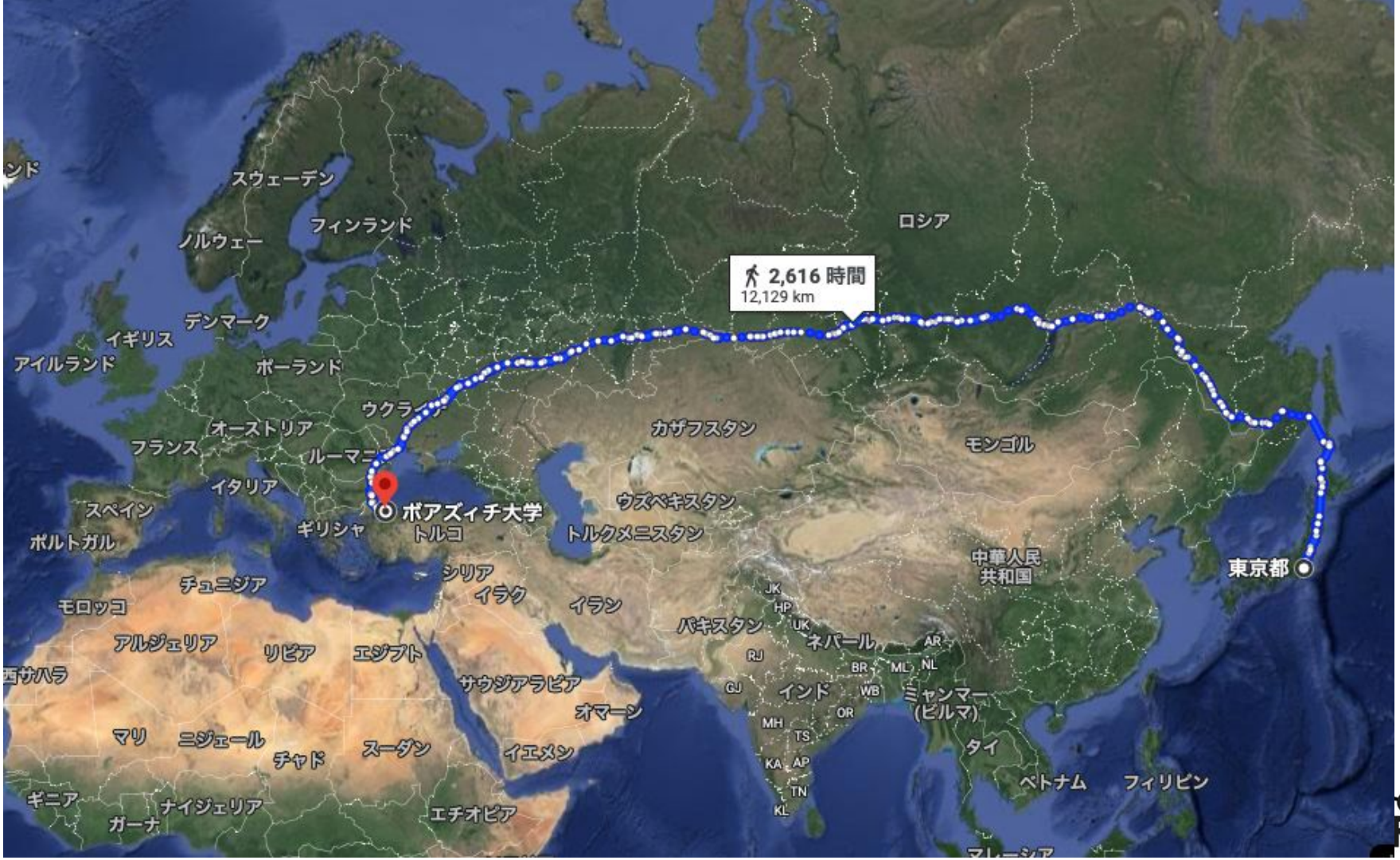
Eighth International Workshop on Symbolic-Neural Learning (SNL2024)

June 26-27, 2024
Miraikan hall, Odaiba Miraikan 7F (Tokyo, Japan)

DeepSym: A Neuro-Symbolic Approach for Symbol Emergence and Planning

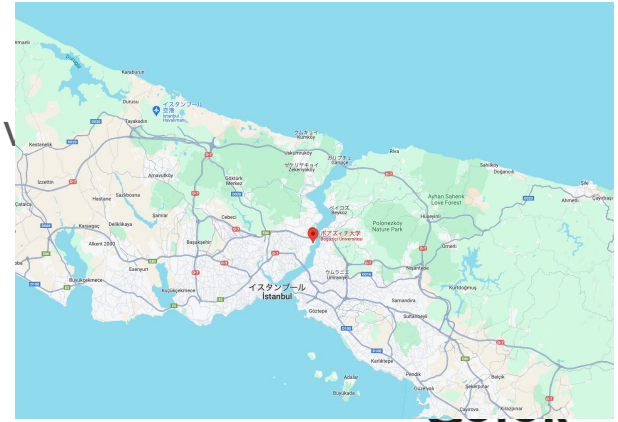
Emre Ugur
Bogazici University
Istanbul, Turkey

COLORS


2,616 時間
12,129 km
ボアズィチ大学
東京都
スウェーデン
フィンランド
ノルウェー
ロシア
デンマーク
イギリス
アイルランド
ポーランド
ウクライナ
オーストリア
フランス
ルーマニア
カザフスタン
モンゴル
イタリア
スペイン
ポルトガル
ギリシャ
トルコ
ウズベキスタン
トルクメニスタン
中華人民共和国
シリア
イラク
イラン
チュニジア
モロッコ
アルジェリア
リビア
エジプト
西サハラ
サウジアラビア
オマーン
パキスタン
ネパール
インド
ミャンマー(ビルマ)
タイ
ベトナム
フィリピン
マレーシア
マリ
ニジェール
チャド
スーダン
イエメン
ギニア
ナイジェリア
ガーナ
エチオピア
JK
HP
UK
AR
RJ
BR
ML
NL
GJ
WB
OR
MH
TS
KA
AP
TN
KL

Bogazici University - General

- One of the top ranking public universities
- Robert College (1863), 1971 integrated into Turkish Univ
- The official educational language is English
- 30 PhD and 47 Msc programs

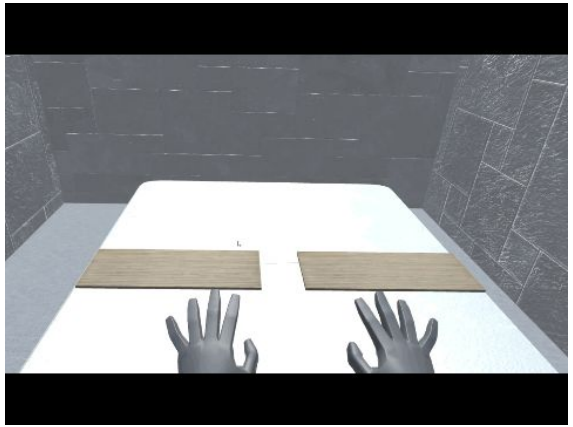


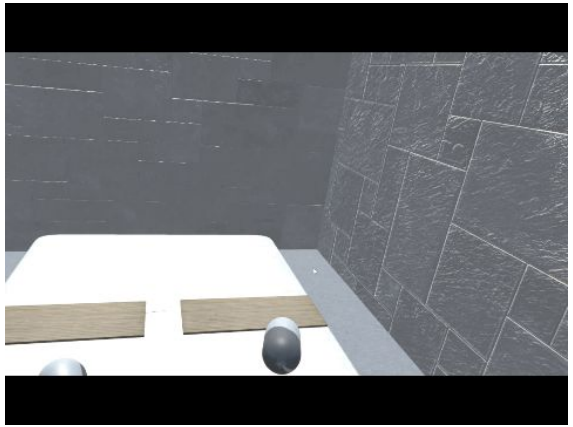
Cognition, Robotics and Learning Lab
http://colors.cmpe.boun.edu.tr/

Cognitive Science Track

- Embodied perception
- Affordances
- Tool Use
- Body Representation
- Time perception
- Event perception

Figure 6. Tool-holding rubber hand mechanism enabling vertical mobility



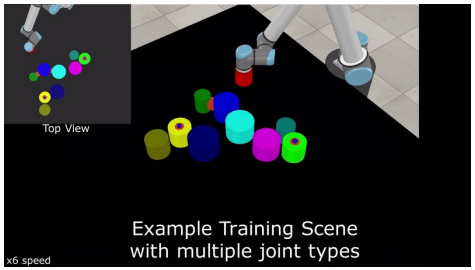


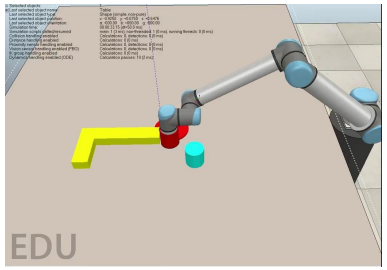
Emre Ugur, Bogazici University, Istanbul

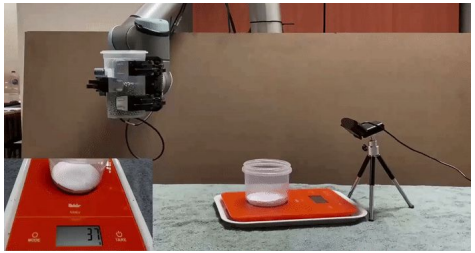
Predictive Robotic Manipulation Learning Track

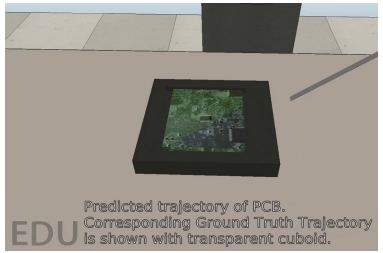
- Learn complex actions from demonstrations and via trial-and-error
- Detect affordances, predict actions along with their parameters
  - Complex multi-modal trajectories, non-linear environment-parameter relationship
- Predict the effect of its actions, before and during their execution.

Predictive Robotic Manipulation Learning Track









- Learn complex actions from demonstrations and via trial-and-error
- Detect affordances, predict actions along with their parameters
  - Complex multi-modal trajectories, non-linear environment-parameter relationship
- Predict the effect of its actions, before and during their execution.

Emre Ugur, Bogazici University, Istanbul

Cognitive and Developmental Robotics Track
Neuro-Symbolic Robotics

- Safe
- General purpose
- Advanced sensorimotor skills
- Social communication and learning
- High-level understanding & reasoning
- Language and planning

Hard-coding or ML not feasible.

DeepSym: Discovering symbols for planning in robotic

- *Standard encoding language for "classical" planning tasks*
  - Planning Domain Definition Language (PDDL)

- Components of a PDDL planning task:
  - Objects: Things in the world that interest us.
  - Predicates: Properties of objects, relations btw. objects: true or false.
  - Actions: Ways of changing the state of the world.

  - Initial state: The state of the world that we start in.
  - Goal: Things that we want to be true.

Minimal PDDL Example

Domain Definition (light_domain.pddl):

```
(define domain light)
(:predicates
  (light-off)
  (light-on)
)

(:action turn-on-light
  :precondition (light-off)
  :effect (and (light-on) (not (light-off)))
)
```

Problem Definition (light_domain.pddl):

```
(define problem light_problem)
(:domain light)

(:init
  (light-off)
)

(:goal (light-on))
```

Use an off-the-shelf AI planner (PDDL solver)

- **turn-on-light**

Minimal PDDL Example

Domain Definition

```
(:predicates ….)

(:action pick
  :parameters (?obj, ?loc)
  :precondition
     (and (robot-at ?loc)
     (object-at ?obj ?loc)
     (graspable ?obj)
  :effect
     (and (holding ?obj)
     (not (object-at ?obj ?loc)
     (not graspable ?obj)))
)

(:action place ….)
(:action carry ….)
```

Problem Definition

```
(define problem robot_problem)
(:domain robot)
(:init
  (robot-at location1)
  (object-at object1 location1)
  (graspable object1)
)

(:goal (and (object-at object1 location2)))
```

Use an off-the-shelf AI planner (PDDL solver)

- **pick (object1 location1)**
- **carry (location1 location2)**
- **place (object1 location2)**

Minimal Probabilistic PDDL (PPDDL) Example

Domain Definition

```
(:predicates ….)
(:action pick
:parameters (?obj ?loc)
:precondition (and (robot-at ?loc) (object-at
?obj ?loc) (graspable ?obj) )

:effect (probabilistic
0.9 (and (holding ?obj) (not (object-at ?obj
?loc)) (not (graspable ?obj)))
0.1 (and (not (holding ?obj))
; The robot fails to pick the object
(object-at ?obj ?loc)
; The object remains at the location
(graspable ?obj))
; The object remains graspable ) )

(:action place ….)  (:action carry ….)
```

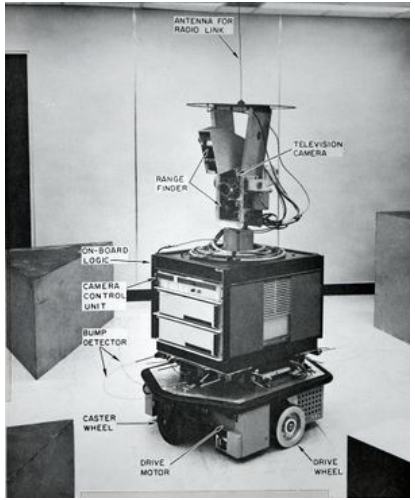
Problem Definition

```
(define problem robot_problem)
(:domain robot)
(:init
  (robot-at location1)
  (object-at object1 location1)
  (graspable object1)
)

(:goal (and (object-at object1 location2)))
```

Use an off-the-shelf AI planner (PDDL solver)

- **pick (object1 location1)**
- **carry (location1 location2)**
- **place (object1 location2)**

AI planners since 1970's



Initial World Model

(∀x∀y∀z)[CONNECTS(x,y,z)⇒CONNEC
CONNECTS(DOOR1,ROOM1,ROOM5)
CONNECTS(DOOR2,ROOM2,ROOM5)
CONNECTS(DOOR3,ROOM3,ROOM5)
CONNECTS(DOOR4,ROOM4,ROOM5)
LOCINROOM(f,ROOM4)
AT(BOX1,a)
AT(BOX2,b)
AT(BOX3,c)
AT(LIGHTSWITCH1,d)
ATROBOT(e)
TYPE(BOX1,BOX)
TYPE(BOX2,BOX)
TYPE(BOX3,BOX)
TYPE(D4,DOOR)
TYPE(D3,DOOR)
TYPE(D2,DOOR)
TYPE(D1,DOOR)

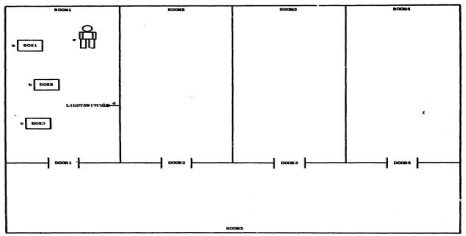
Operators

goto1(m): Robot goes to coordinate location *m*.
Preconditions:
(ONFLOOR) ∧ (∃x)[INROOM(ROBOT,x) ∧ LOCINROOM(m,x)]
Delete list: ATROBOT($),NEXTTO(ROBOT,$)
Add list: ATROBOT(*m*)

goto2(m): Robot goes next to item *m*.
Preconditions:
(ONFLOOR) ∧ {(∃x)[INROOM(ROBOT,x) ∧ INROOM(m,x)] ∨ (∃x)(∃y)
[INROOM(ROBOT,x) ∧ CONNECTS(m,x,y)]}
Delete list: ATROBOT($),NEXTTO(ROBOT,$)
Add list: NEXTTO(ROBOT,*m*)

pushto(m,n): robot pushes object *m* next to item *n*
Precondition:
PUSHABLE(*m*) ∧ ONFLOOR ∧ NEXTTO(ROBOT,*m*) ∧ {(∃x)[INROOM(*m*,x)
∧ INROOM(*n*,x)] ∨ (∃x,∃y)[INROOM(*m*,x) ∧ CONNECTS(*n*,x,y)]}
Delete list: AT ROBOT ($) NEXTTO (ROBOT $) NEXTTO ($,*m*)
AT (*m*$) NEXTTO (*m*$)
Add list: NEXTTO(*m*,*n*)
NEXTTO(*n*,*m*)
NEXTTO(ROBOT,*m*)

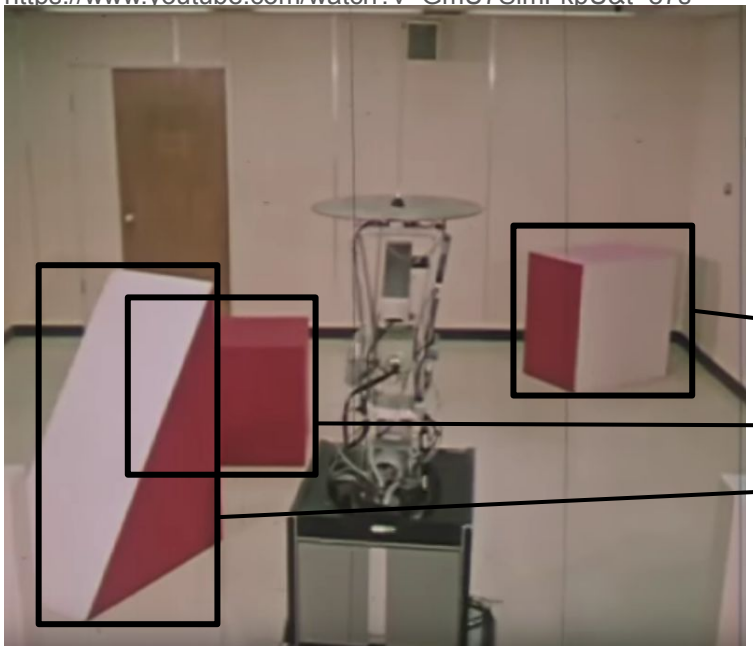
Turn on the lightswitch
Push three boxes together
Go to a location in another room



Fikes, Richard E., and Nils J. Nilsson. "STRIPS: A new approach to the application of theorem proving to problem solving." *Artificial intelligence* 2.3-4 (1971): 189-208.
Kuipers, B.; Feigenbaum, E. A.; Hart, P. E.; and Nilsson, N. J. 2017. Shakey: From Conception to History. AI Magazine 38(1): 88–103.

Sensor - Symbol mapping

Shakey: Experiments in Robot Planning and Learning (1972)
https://www.youtube.com/watch?v=GmU7SimFkpU&t=87s



Requires pre-defining all symbols

classifiers

Initial World Model
(∀x∀y∀z)[CONNECTS(x,y,z)⇒CONNEC
CONNECTS(DOOR1,ROOM1,ROOM5)
CONNECTS(DOOR2,ROOM2,ROOM5)
CONNECTS(DOOR3,ROOM3,ROOM5)
CONNECTS(DOOR4,ROOM4,ROOM5)
LOCINROOM(f,ROOM4)
AT(BOX1,a)
AT(BOX2,b)
AT(BOX3,c)
AT(LIGHTSWITCH1,d)
ATROBOT(e)
TYPE(BOX1,BOX)
TYPE(BOX2,BOX)
TYPE(BOX3,BOX)
TYPE(D4,DOOR)
TYPE(D3,DOOR)
TYPE(D2,DOOR)
TYPE(D1,DOOR)

Classify sensor data into symbols

- Manually
- Learning

Pre-defining symbols??

- depends on agent, agent's capabilities, environment, other agents. …

… Difficult to provide a general set of symbols...

Possible to provide symbols in extremely constrained settings.

- It breaks quickly in slightly different environments.

Pre-defining symbols, learning sensor - symbol learning Neuro-symbolic approaches today

THE NEURO-SYMBOLIC CONCEPT LEARNER: INTERPRETING SCENES, WORDS, AND SENTENCES FROM NATURAL SUPERVISION

Jiayuan Mao
MIT CSAIL and IIIS, Tsinghua University
mjy14@mails.tsinghua.edu.cn

Chuang Gan
MIT-IBM Watson AI Lab
ganchuang@csail.mit.edu

Pushmeet Kohli
Deepmind
pushmeet@google.com

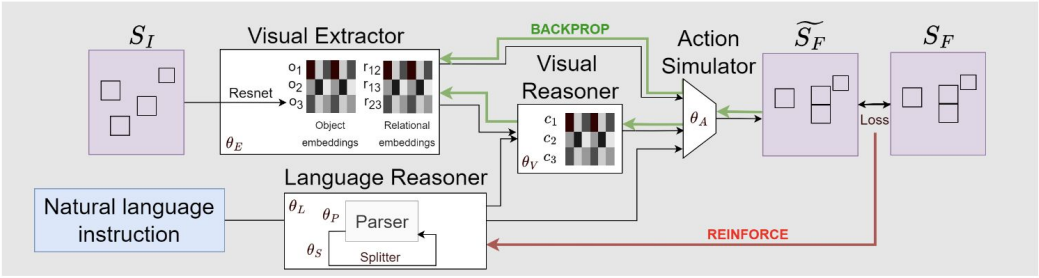
Joshua B. Tenenbaum
MIT BCS, CBMM, CSAIL
jbt@mit.edu

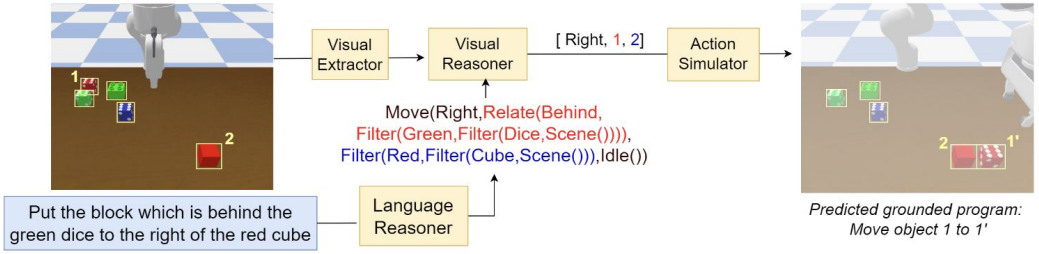
Jiajun Wu
MIT CSAIL
jiajunwu@mit.edu

2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Learning Neuro-symbolic Programs for Language Guided Robot Manipulation

Namasivayam K*[1], Himanshu Singh*[1], Vishal Bindal*[1], Arnav Tuli[1], Vishwajeet Agrawal#[2], Rahul Jain#[2], Parag Singla[1] and Rohan Paul[1]
[1]Affiliated with IIT Delhi. [2]Work done when at IIT Delhi. * and # denote equal contribution.





Predicted grounded program: Move object 1 to 1'

Pre-defining symbols, learning sensor - symbol learning
Neuro-symbolic approaches today

Do As I Can, Not As I Say:
Grounding Language in Robotic Affordances

[1] Michael Ahn*, Anthony Brohan*, Noah Brown*, Yevgen Chebotar*, Omar Cortes*, Byron David*, Chelsea Finn*, Chuyuan Fu†, Keerthana Gopalakrishnan*, Karol Hausman*, Alex Herzog†, Daniel Ho†, Jasmine Hsu*, Julian Ibarz*, Brian Ichter*, Alex Irpan*, Eric Jang*, Rosario Jauregui Ruano*, Kyle Jeffrey*, Sally Jesmonth*, Nikhil J Joshi*, Ryan Julian*, Dmitry Kalashnikov*, Yuheng Kuang*, Kuang-Huei Lee*, Sergey Levine*, Yao Lu*, Linda Luu*, Carolina Parada*, Peter Pastor†, Jornell Quiambao*, Kanishka Rao*, Jarek Rettinghouse*, Diego Reyes*, Pierre Sermanet*, Nicolas Sievers*, Clayton Tan*, Alexander Toshev*, Vincent Vanhoucke*, Fei Xia*, Ted Xiao*, Peng Xu*, Sichun Xu*, Mengyuan Yan†, Andy Zeng*

*Robotics at Google, †Everyday Robots

RT-1: Robotics Transformer for Real-World Control at Scale

[1] Anthony Brohan*, Noah Brown*, Justice Carbajal*, Yevgen Chebotar*, Joseph Dabis*, Chelsea Finn*, Keerthana Gopalakrishnan*, Karol Hausman*, Alex Herzog†, Jasmine Hsu*, Julian Ibarz*, Brian Ichter*, Alex Irpan*, Tomas Jackson*, Sally Jesmonth*, Nikhil J Joshi*, Ryan Julian*, Dmitry Kalashnikov*, Yuheng Kuang*, Isabel Leal*, Kuang-Huei Lee‡, Sergey Levine*, Yao Lu*, Utsav Malla*, Deeksha Manjunath*, Igor Mordatch‡, Ofir Nachum‡, Carolina Parada*, Jodilyn Peralta*, Emily Perez*, Karl Pertsch*, Jornell Quiambao*, Kanishka Rao*, Michael Ryoo*, Grecia Salazar*, Pannag Sanketi*, Kevin Sayed*, Jaspiar Singh*, Sumedh Sontakke‡, Austin Stone*, Clayton Tan*, Huong Tran*, Vincent Vanhoucke*, Steve Vega*, Quan Vuong*, Fei Xia*, Ted Xiao*, Peng Xu*, Sichun Xu*, Tianhe Yu*, Brianna Zitkovich*

*Robotics at Google, †Everyday Robots, ‡Google Research, Brain Team

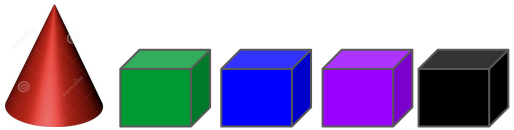
RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich

Google DeepMind. Authors listed in alphabetical order, with contributions listed in Appendix A.

Pre-defining symbols, learning sensor - symbol learning
Neuro-symbolic approaches today

Is this the right way (relying on LLMs)?

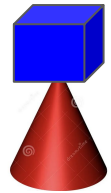


User : There is a red cone, a green cube, a blue cube, a purple cube, and a **black** cube in the current scene.
Possible actions:
a) Put the purple cube next to the red cone
b) Put the blue cube on top of the red cone
c) Put the purple cube next to the red cone
…..
h) Put the green cube in front of the blue cube

How can I built the **highest possible tower**?

ChatGPT : To build the highest possible tower, we need to consider the stability and balance of the objects.

The best action would be to put the blue cube on top of the red cone (action b). This is because **the red cone can provide a stable base for the cube, and the cube can sit securely on top of the cone**.



B. Celik, A. Ahmetoglu, E. Ugur, E. Oztop, Developmental Scaffolding with Large Language Models, ICDL 2023.

GPT-4 Technical Report

5 Limitations

Despite its capabilities, GPT-4 has similar limitations as earlier GPT models. Most importantly, it still is not fully reliable (it "hallucinates" facts and makes reasoning errors). Great care should be taken when using language model outputs, particularly in high-stakes contexts, with the exact protocol (such as human review, grounding with additional context, or avoiding high-stakes uses altogether) matching the needs of specific applications. See our System Card for details.

> GPT-4 generally lacks knowledge of events that have occurred after the vast majority of its pre-training data cuts off in September 2021[10], and does not learn from its experience. It can sometimes make simple reasoning errors which do not seem to comport with competence across so many domains, or be overly gullible in accepting obviously false statements from a user. It can fail at hard problems the same way humans do, such as introducing security vulnerabilities into code it produces.

GPT-4 can also be confidently wrong in its predictions, not taking care to double-check work when it's likely to make a mistake. Interestingly, the pre-trained model is highly calibrated (its predicted

Symbol grounding vs. symbol discovery

Predefined symbols and rules
with human bias

GROUND

Robot's continuous
sensorimotor experience

Emerging symbols and rules

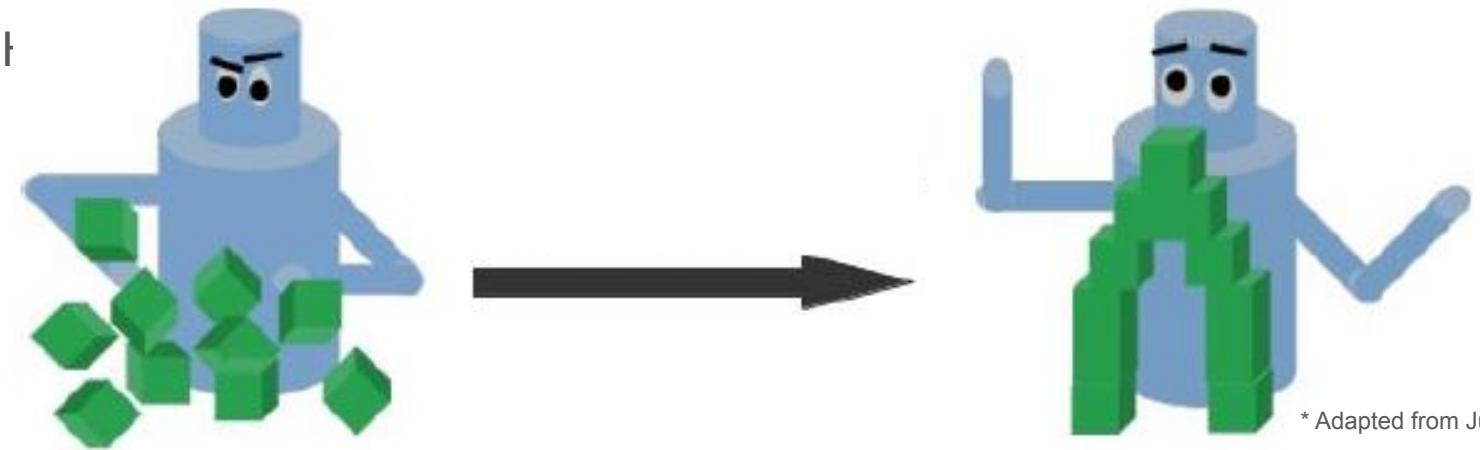
DISCOVER

Robot's continuous
sensorimotor experience

Taniguchi, T., **Ugur**, E., Hoffmann, M., Jamone, L., Nagai, T., Rosman, B., ... & **Wörgötter**, F. (2018). Symbol emergence in cognitive developmental systems: a survey. IEEE transactions on Cognitive and Developmental Systems, 11(4), 494-516.

Instead of pre-defining symbols, the robots can discover symbols themselves

Discover useful! (affordance) symbols



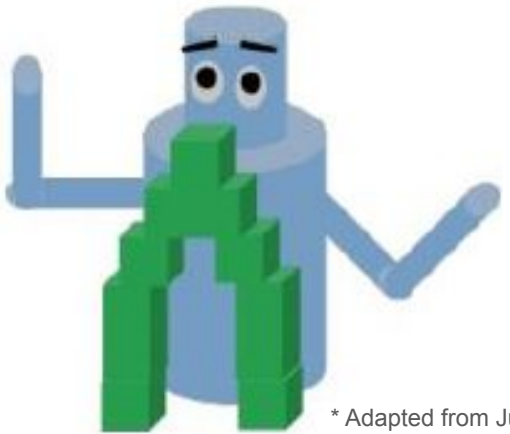
* Adapted from Justus Piater

Taniguchi, T., Ugur, E. et al. (2018). Symbol emergence in cognitive developmental systems: a survey. IEEE TCDS, 11(4), 494-516.
Yamanobe, Natsuki, et al. "A brief review of affordance in robotic manipulation research." Advanced Robotics 31.19-20 (2017): 1086-1101.

Instead of pre-defining symbols, the robots can discover symbols themselves

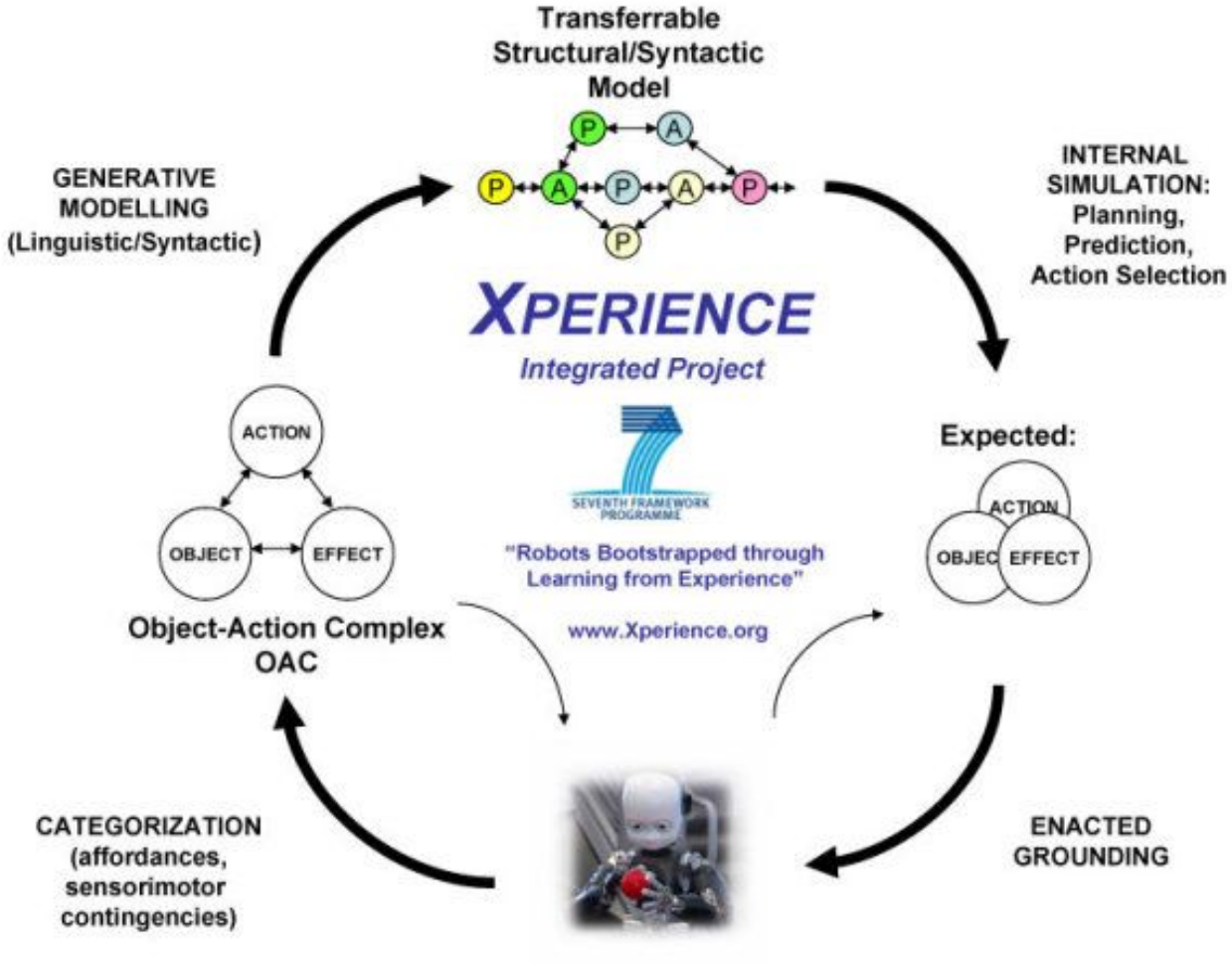
Discover useful! (affordance) symbols

- High-level cognition: Reasoning, planning, communication, language



* Adapted from Justus Piater

Taniguchi, T., Ugur, E. et al. (2018). Symbol emergence in cognitive developmental systems: a survey. IEEE TCDS, 11(4), 494-516.
Yamanobe, Natsuki, et al. "A brief review of affordance in robotic manipulation research." Advanced Robotics 31.19-20 (2017): 1086-1101.


Transferrable
Structural/Syntactic
Model
P
A
GENERATIVE
MODELLING
(Linguistic/Syntactic)
INTERNAL
SIMULATION:
Planning,
Prediction,
Action Selection
XPERIENCE
Integrated Project
SEVENTH FRAMEWORK
PROGRAMME
"Robots Bootstrapped through
Learning from Experience"
www.Xperience.org
ACTION
OBJECT
EFFECT
Object-Action Complex
OAC
Expected:
ACTION
OBJEC
EFFECT
CATEGORIZATION
(affordances,
sensorimotor
contingencies)
ENACTED
GROUNDING

Symbol discovery for (long-horizon) robotic planning

Constructing Symbolic Representations for High-Level Plann

George Konidaris, Leslie Pack Kaelbling and **Tomas Lozano-Perez**
MIT Computer Science and Artificial Intelligence Laboratory

Partition dataset s.t. subgoal property is satisfied, and learn action precondition & effect densities.
Learned symbols are sound & complete.

(ICRA 2015)

Bottom-Up Learning of Object Categories, Action Effects and Logical Rules: From Continuous Manipulative Exploration to Symbolic Planning

Emre Ugur and Justus Piater

Partition object features to predict effect categories

Learning Portable Representations for High-Level

Steven James [1] **Benjamin Rosman** [1] **George Konidaris** [2]

Learn object-centric representations

Learning Neuro-Symbolic Skills for Bilevel Planning

Tom Silver, Ashay Athalye
Joshua B. Tenenbaum, Tomás Lozano-Pérez, Leslie Pack Kaelbling
MIT Computer Science and Artificial Intelligence Laboratory
{tslvr, ashay, jbt, tlp, lpk}@mit.edu

Learn symbols via program synthesis that directly increase planning performance

Table of Contents - Methods

- DeepSym: Discovering symbols via predictive encoder-decoder network with binary bottleneck layer
  - Allows symbol-based rule learning and planning
  - Limitation 1: Planning only with one or two objects
    - to allow multi-object planning:
      - DeepSym with Attention
      - DeepSym with Graph Neural Networks
  - Limitation 2: Actions are already discrete
    - to discover motion primitives:
      - Generative systems (Conditional Neural Processes) with Mixture of Experts and Winner-take-all mechanism

Journal of Artificial Intelligence Research 75 (2022) 709-745 Submitted 03/2022; published 11/2022

DeepSym: Deep Symbol Generation and Rule Learning for Planning from Unsupervised Robot Interaction

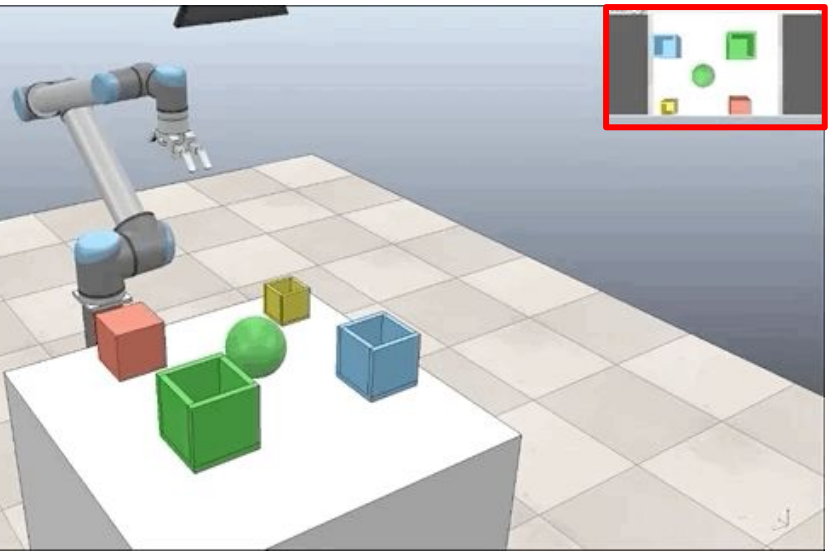
Alper Ahmetoglu ALPER.AHMETOGLU@BOUN.EDU.TR
M. Yunus Seker YUNUS.SEKER1@BOUN.EDU.TR
Department of Computer Engineering
Bogazici University, Istanbul, Turkey

Justus Piater JUSTUS.PIATER@UIBK.AC.AT
Department of Computer Science
Universität Innsbruck, Austria

Erhan Oztop ERHAN.OZTOP@OZYEGIN.EDU.TR
Osaka University, Osaka, Japan
Ozyegin University, Istanbul, Turkey

Emre Ugur EMRE.UGUR@BOUN.EDU.TR
Department of Computer Engineering
Bogazici University, Istanbul, Turkey

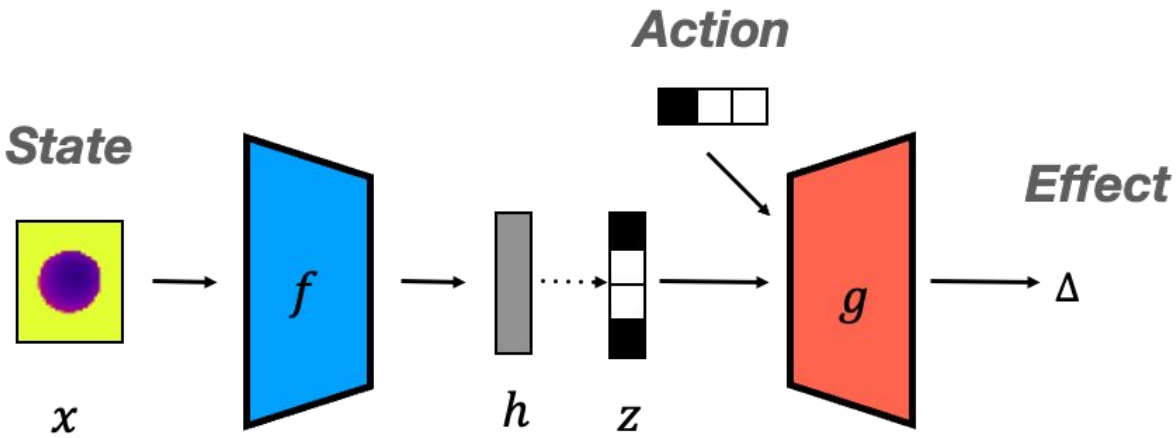
Discover discrete symbols from continuous experience



```
(:action stack
:parameters (?obj1 ?obj2)
:precondition (and (graspable obj1?)
(object-at ?obj1 ?loc) (object-at ?obj2
?loc) (rollable ?obj1) (insertable ?obj2)
)

:effect (probabilistic

0.9 (and (not object-at ?obj1 ?loc) (not
object-at ?obj1 ?loc)
(object-at ?obj3 ?loc) (not graspable
?obj3) (insertable ?obj3))

0.1 (and (not (graspable ?obj1))
```

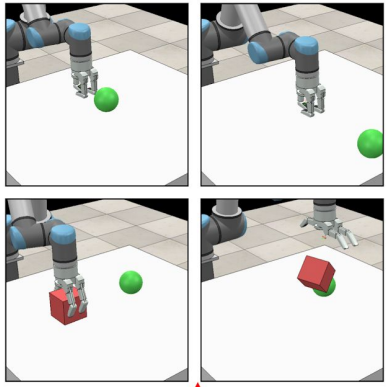
1. Learn object **symbols** that encode **effects** of **actions** with **neural nets**.
   a. Symbols encode affordances, i.e. (object, action, effect) relations
2. Use the learned symbols for **planning**.

DeepSym: Deep Symbol Generation and Rule Learning



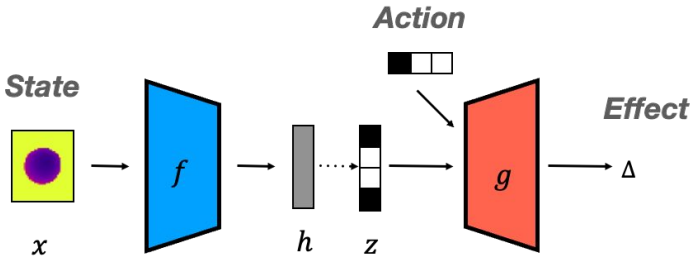
1. Learn object **symbols** that encode **effects** of **actions** with **neural nets**.
   a. Symbols encode affordances, i.e. (object, action, effect) relations
2. Use the learned symbols for **planning**.

(I) Interaction with objects with pre-defined actions



Low-level continuous object and effect observations

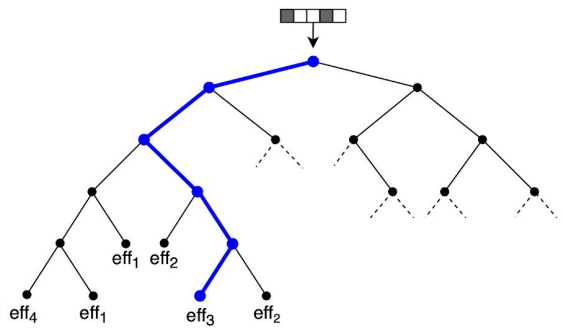
(II) Symbol formation (discovery of object and effect categories)



Generated plans for verification

discrete object and effect observations

```
(:action stack23
:parameters (?below ?above)
:precondition (and (not (stacked)) (not (inserted)) (pickloc ?above)
                   (stackloc ?below) (objtype3 ?below) (objtype1 ?above)
                   (relation1 ?below ?above))
:effect (and (probabilistic
    0.578 (and (inserted) (instack ?above)
               (stackloc ?above) (not (stackloc ?below)))
    0.422 (and (stacked) (inserted) (instack ?above)
               (stackloc ?above) (not (stackloc ?below)))
    0.000 (roll1)
    0.000 (tumble1)
    0.000 (roll2)
    0.000 (tumble2))
    (not (pickloc ?above))))
```

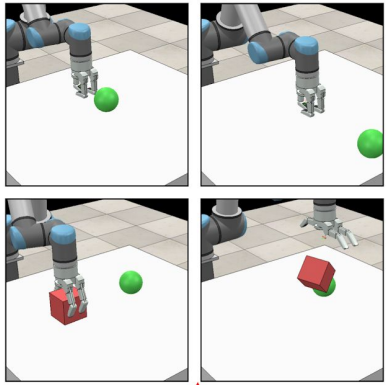


Probabilistic rules

(IV) Translation of rules to PPDDL operators

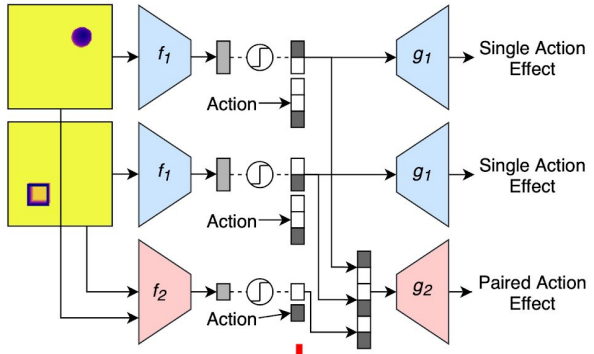
(III) Decision tree learning

(I) Interaction with objects with pre-defined actions

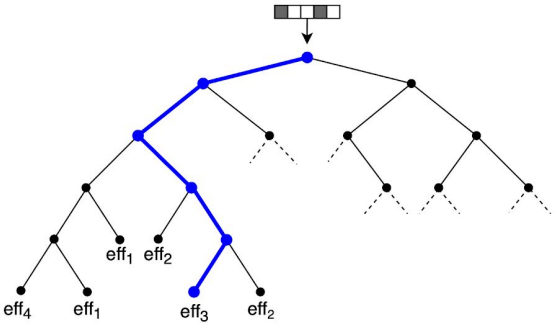


(II) Symbol formation (discovery of object and effect categories)

Low-level continuous object and effect observations



High-level discrete object and effect observations



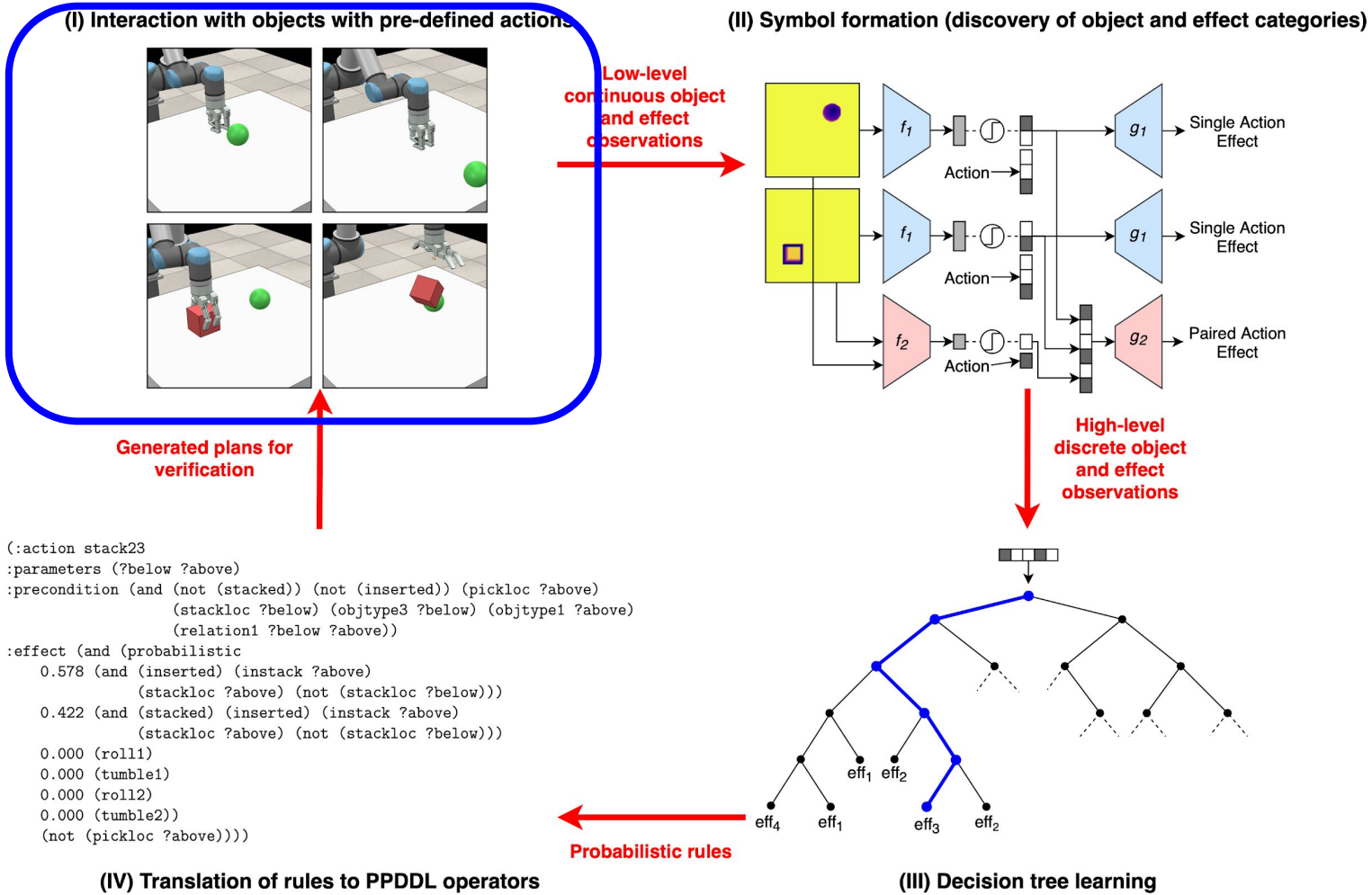
(III) Decision tree learning

Probabilistic rules

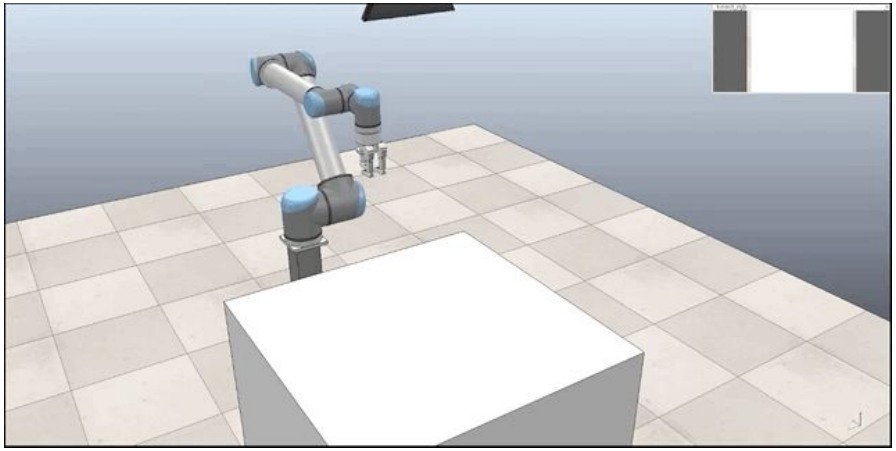
(IV) Translation of rules to PPDDL operators

Generated plans for verification

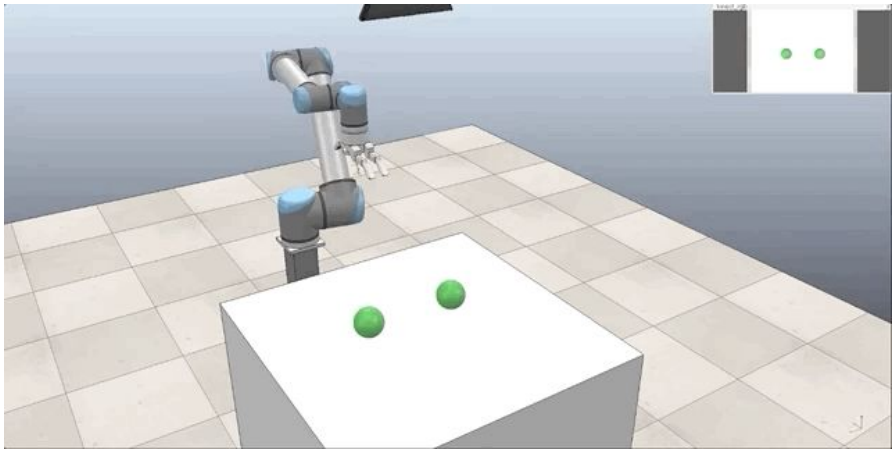
```
(:action stack23
:parameters (?below ?above)
:precondition (and (not (stacked)) (not (inserted)) (pickloc ?above)
                   (stackloc ?below) (objtype3 ?below) (objtype1 ?above)
                   (relation1 ?below ?above))
:effect (and (probabilistic
    0.578 (and (inserted) (instack ?above)
               (stackloc ?above) (not (stackloc ?below)))
    0.422 (and (stacked) (inserted) (instack ?above)
               (stackloc ?above) (not (stackloc ?below)))
    0.000 (roll1)
    0.000 (tumble1)
    0.000 (roll2)
    0.000 (tumble2))
    (not (pickloc ?above))))
```


(I) Interaction with objects with pre-defined actions
Low-level continuous object and effect observations
(II) Symbol formation (discovery of object and effect categories)
f_1
Action
g_1
Single Action Effect
f_1
Action
g_1
Single Action Effect
f_2
Action
g_2
Paired Action Effect
Generated plans for verification
High-level discrete object and effect observations
(:action stack23
:parameters (?below ?above)
:precondition (and (not (stacked)) (not (inserted)) (pickloc ?above)
(stackloc ?below) (objtype3 ?below) (objtype1 ?above)
(relation1 ?below ?above))
:effect (and (probabilistic
0.578 (and (inserted) (instack ?above)
(stackloc ?above) (not (stackloc ?below)))
0.422 (and (stacked) (inserted) (instack ?above)
(stackloc ?above) (not (stackloc ?below)))
0.000 (roll1)
0.000 (tumble1)
0.000 (roll2)
0.000 (tumble2))
(not (pickloc ?above))))
eff_1 eff_2
eff_4 eff_1 eff_3 eff_2
Probabilistic rules
(IV) Translation of rules to PPDDL operators
(III) Decision tree learning

Interaction with objects with pre-defined actions



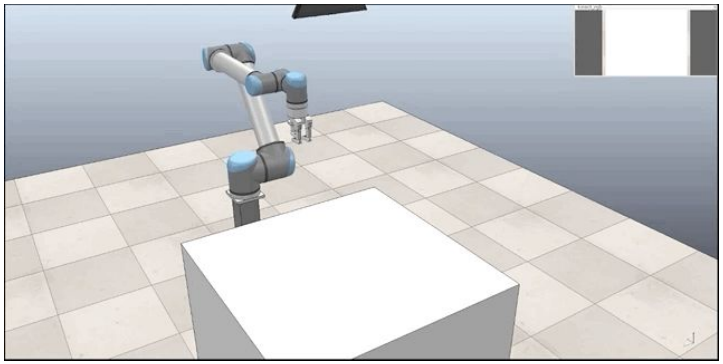
Single-object interactions (Push action)

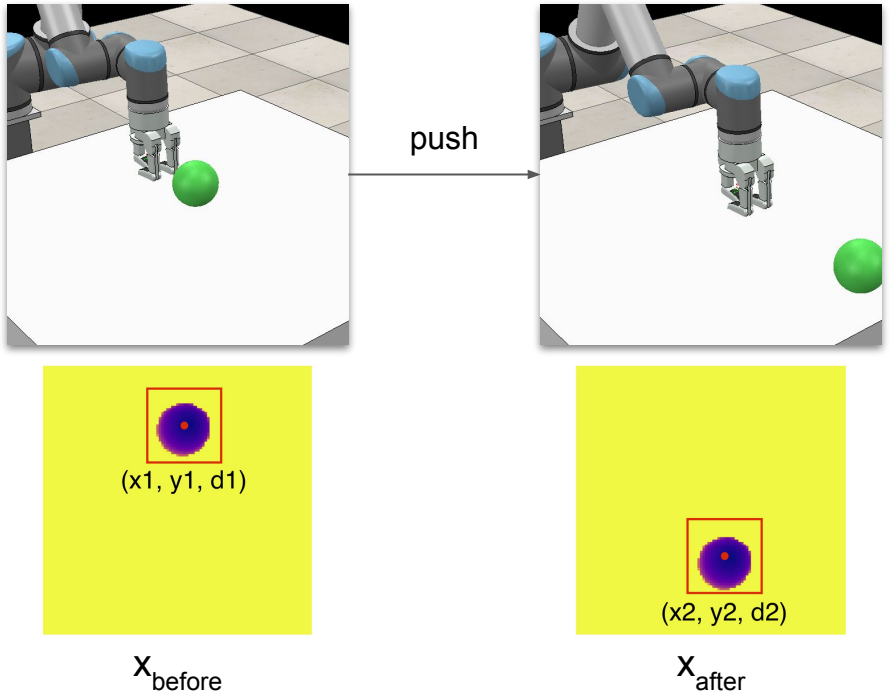


Paired-object interactions (Stack action)

Step (I): Interaction with objects with pre-defined actions

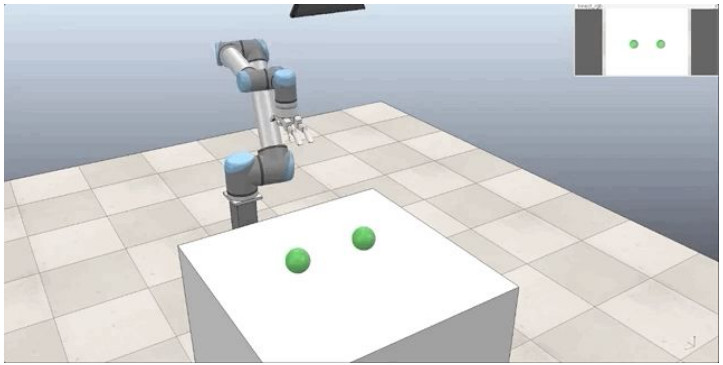
- Input: Depth image of the table.
- 3 actions: [push-front, push-left, push-top]
- Effect: (Δx, Δy, Δd, ΔF)
- 5 objects: [sphere, cube, vertical cylinder, horizontal cylinder, cup]

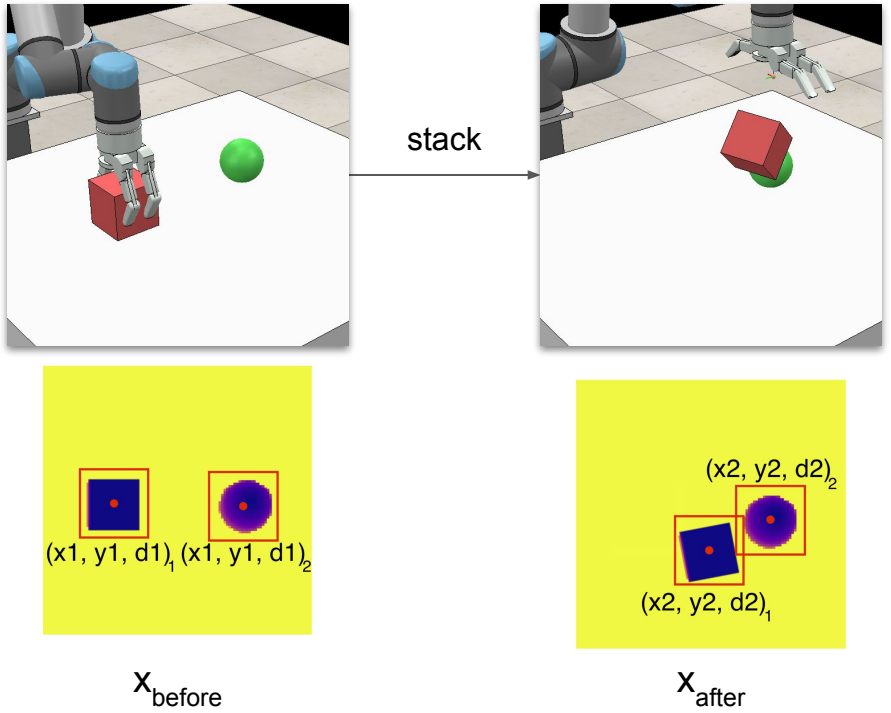




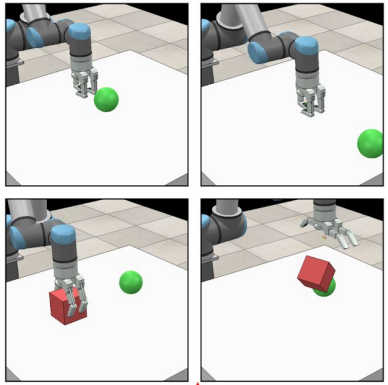
Step (I): Interaction with objects with pre-defined actions

- Input: Depth image of the table.
- Single action: [stack]
- Effect: $(\Delta x, \Delta y, \Delta d)_1$, $(\Delta x, \Delta y, \Delta d)_2$
- 5 objects: [sphere, cube, vertical cylinder, horizontal cylinder, cup]



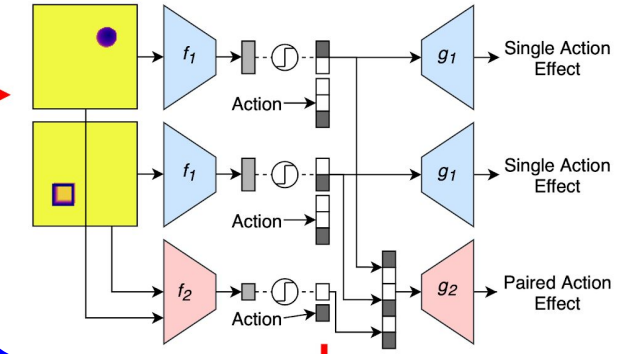


(I) Interaction with objects with pre-defined actions



(II) Symbol formation (discovery of object and effect categories)

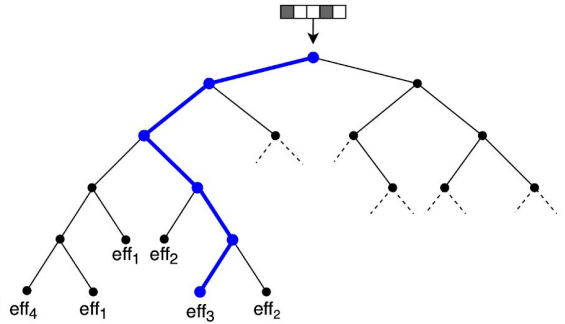
Low-level continuous object and effect observations



Generated plans for verification

High-level discrete object and effect observations

```
(:action stack23
:parameters (?below ?above)
:precondition (and (not (stacked)) (not (inserted)) (pickloc ?above)
                   (stackloc ?below) (objtype3 ?below) (objtype1 ?above)
                   (relation1 ?below ?above))
:effect (and (probabilistic
    0.578 (and (inserted) (instack ?above)
               (stackloc ?above) (not (stackloc ?below)))
    0.422 (and (stacked) (inserted) (instack ?above)
               (stackloc ?above) (not (stackloc ?below)))
    0.000 (roll1)
    0.000 (tumble1)
    0.000 (roll2)
    0.000 (tumble2))
    (not (pickloc ?above))))
```



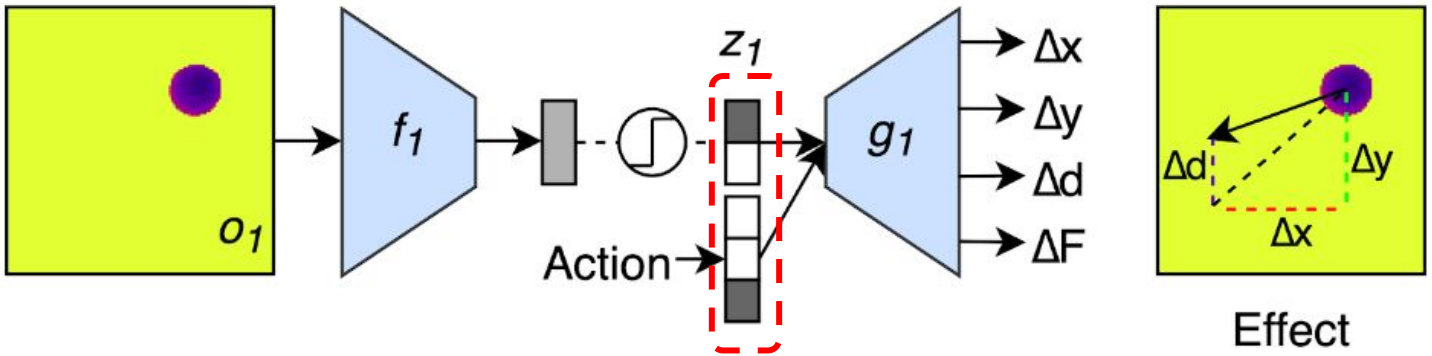
Probabilistic rules

(IV) Translation of rules to PPDDL operators

(III) Decision tree learning

Step (II): Symbol formation (single objects)

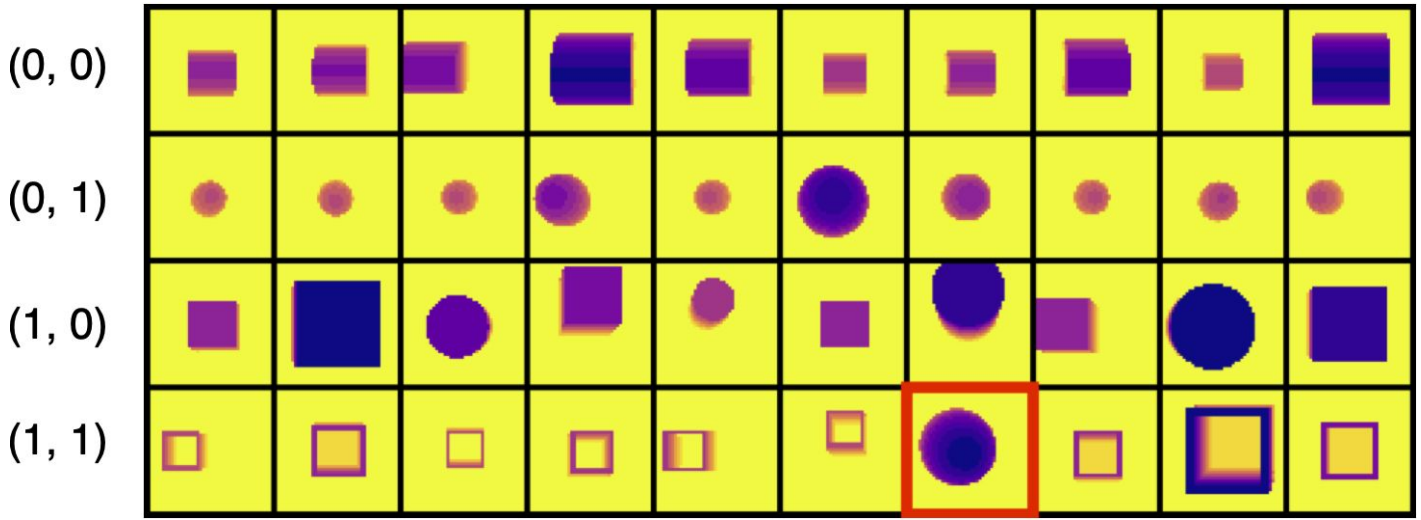
- A differentiable end-to-end deep neural network.
- Binarized hidden units with straight-through estimator (STE) for backprop.
- Predicts the effect of an action.
- Latent layer combines info from observation, action, and effect.



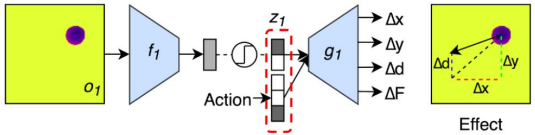
[push-front, push-left, or push-top]

Step (II): Symbol formation - example

- Object categories found with two binary units (four possible categories).
- Categories can be interpreted as `rollable`, `unstable`, `stable`, `insertable`.
- Unsupervised, only based on effect.



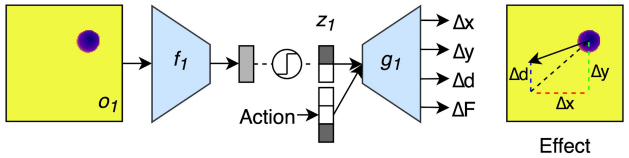
Step (II): Symbol formation (single objects)



	Our method			
Obj\Cat	(0, 0)	(0, 1)	(1, 0)	(1, 1)
Sphere	0.03	**0.97**	0.0	0.0
Cube	0.0	0.0	**0.97**	0.01
V. Cylinder	0.01	0.0	**0.99**	0.0
H. Cylinder	**0.86**	0.03	0.10	0.0
Cup	0.0	0.0	0.02	**0.98**

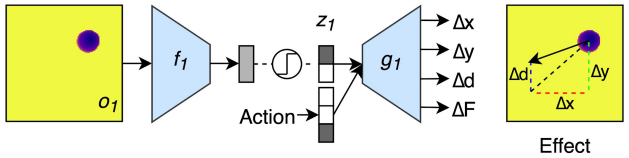
Step (II): Symbol formation (paired objects)

- Symbols are discovered from single object interactions (i.e. push from different sides).



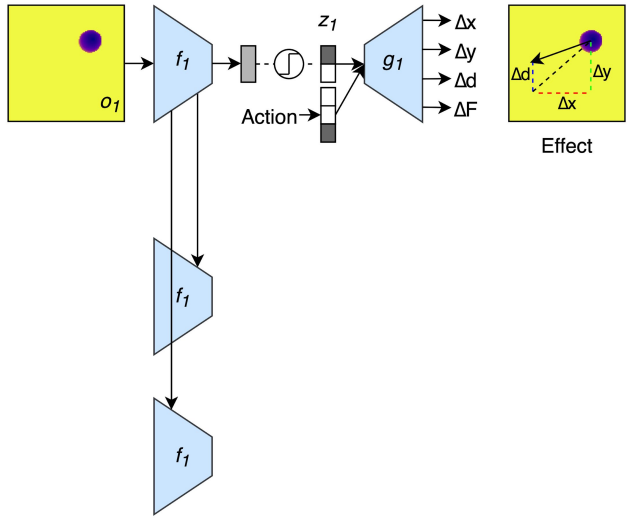
Step (II): Symbol formation (paired objects)

- Symbols are discovered from single object interactions (i.e. push from different sides).
- Use these symbols in further learning.



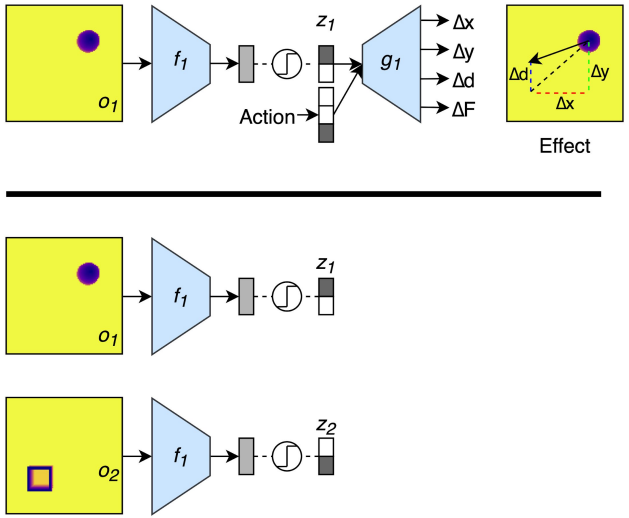
Step (II): Symbol formation (paired objects)

- Symbols are discovered from single object interactions (i.e. push from different sides).
- Use these symbols in further learning.
- Freeze the previously learned encoder.



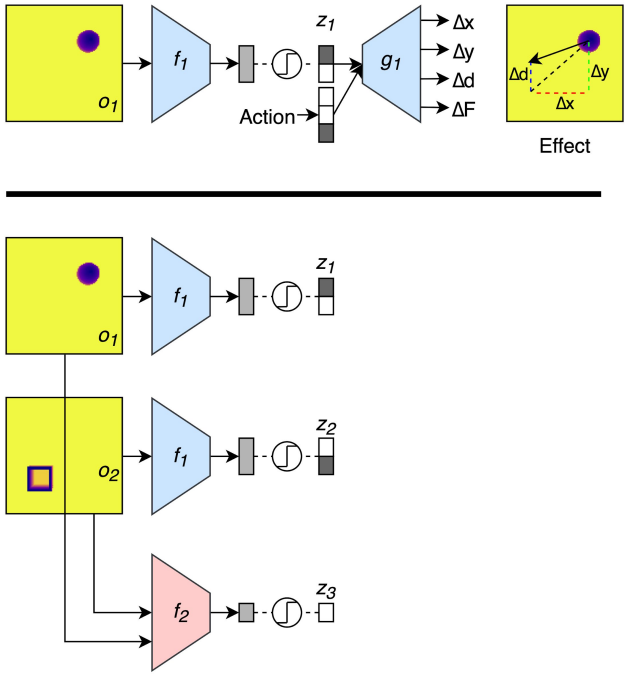
Step (II): Symbol formation (paired objects)

- Symbols are discovered from single object interactions (i.e. push from different sides).
- Use these symbols in further learning.
- Freeze the previously learned encoder.



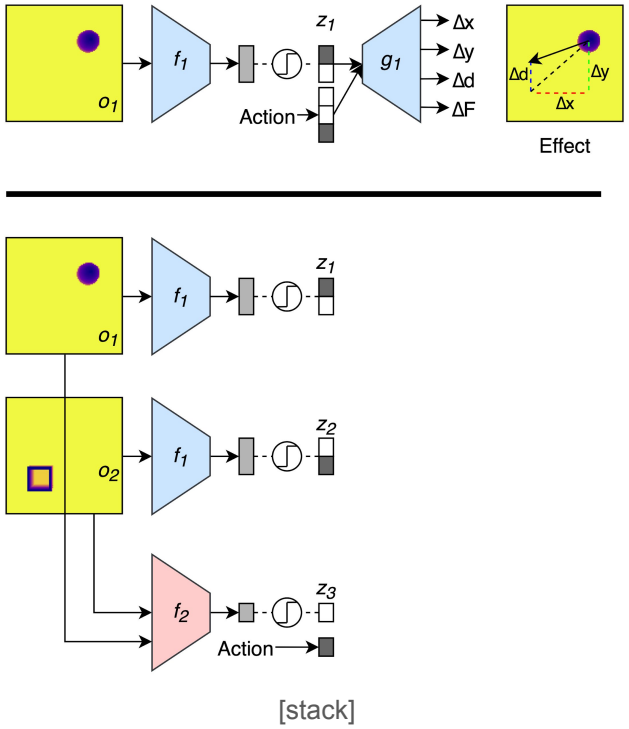
Step (II): Symbol formation (paired objects)

- Symbols are discovered from single object interactions (i.e. push from different sides).
- Use these symbols in further learning.
- Freeze the previously learned encoder.
- A new encoder for paired-object categories.



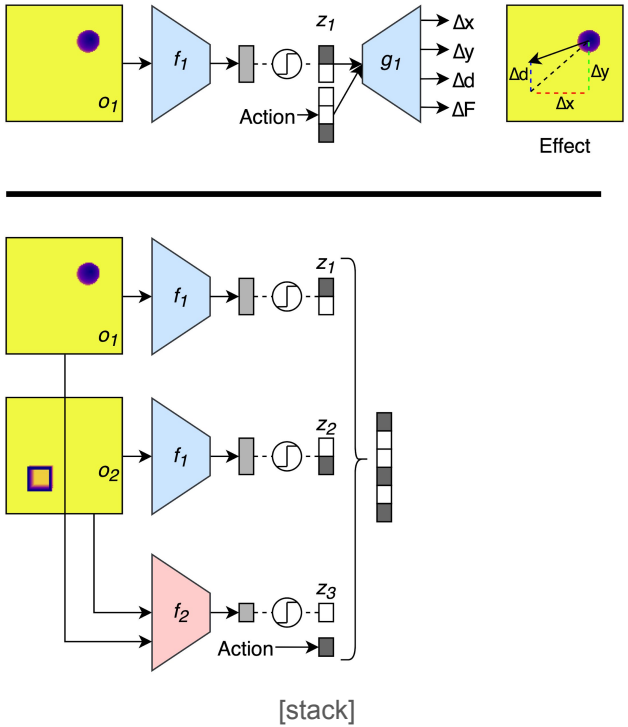
Step (II): Symbol formation (paired objects)

- Symbols are discovered from single object interactions (i.e. push from different sides).
- Use these symbols in further learning.
- Freeze the previously learned encoder.
- A new encoder for paired-object categories.
- Concatenate the action.



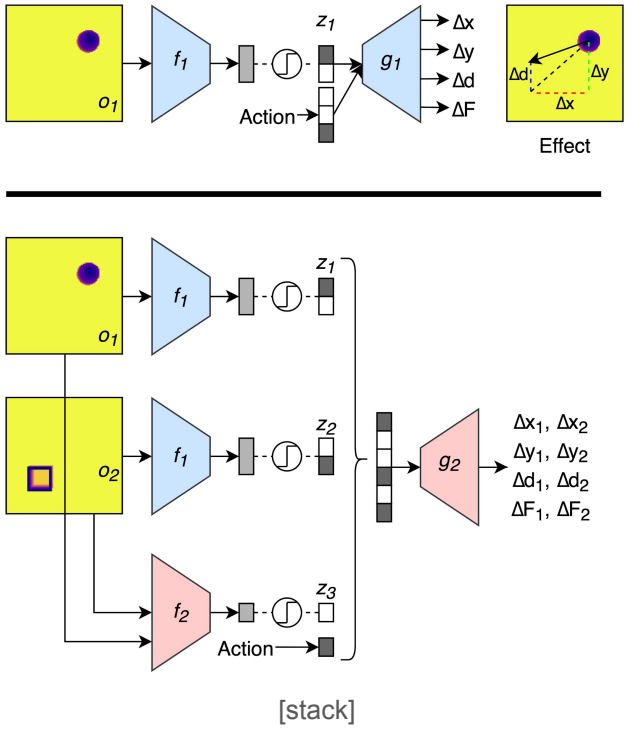
Step (II): Symbol formation (paired objects)

- Symbols are discovered from single object interactions (i.e. push from different sides).
- Use these symbols in further learning.
- Freeze the previously learned encoder.
- A new encoder for paired-object categories.
- Concatenate the action.



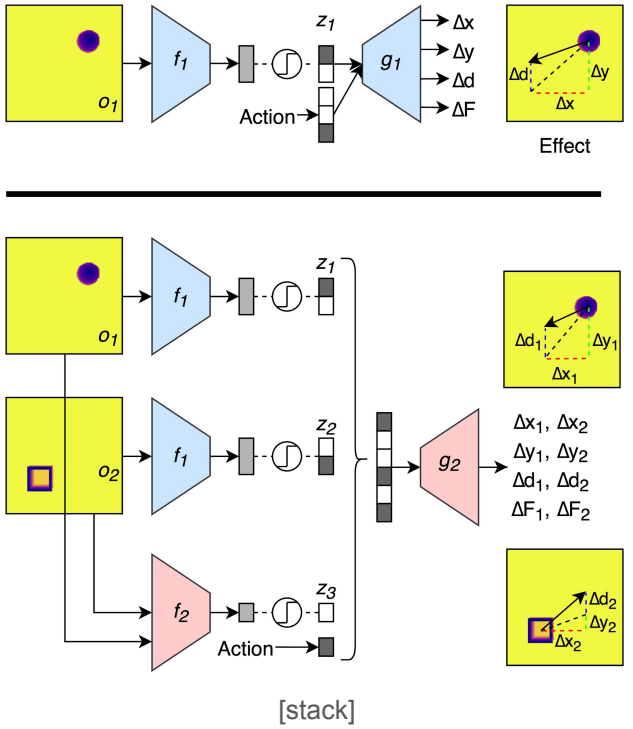
Step (II): Symbol formation (paired objects)

- Symbols are discovered from single object interactions (i.e. push from different sides).
- Use these symbols in further learning.
- Freeze the previously learned encoder.
- A new encoder for paired-object categories.
- Concatenate the action.
- Predict the effect.



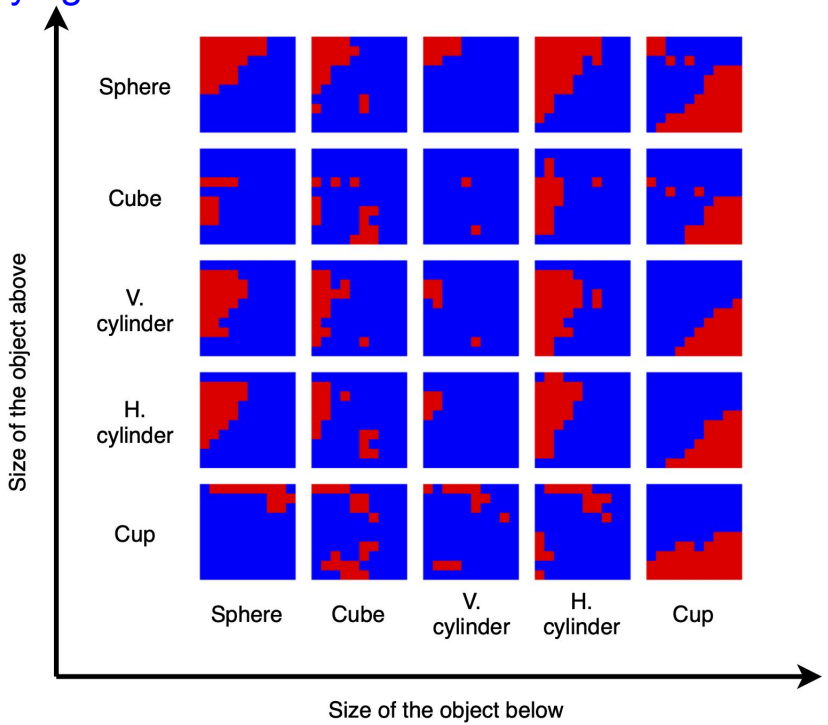
Step (II): Symbol formation (paired objects)

- Symbols are discovered from single object interactions (i.e. push from different sides).
- Use these symbols in further learning.
- Freeze the previously learned encoder.
- A new encoder for paired-object categories.
- Concatenate the action.
- Predict the effect.



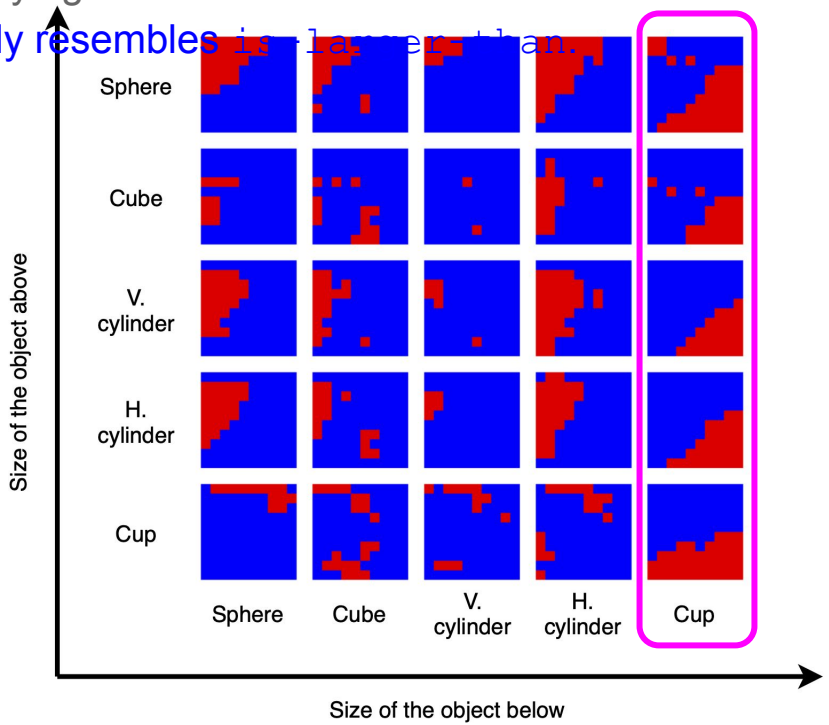
Step (II): Symbol formation (paired objects)

- One dimensional paired-object categories for varying sizes.

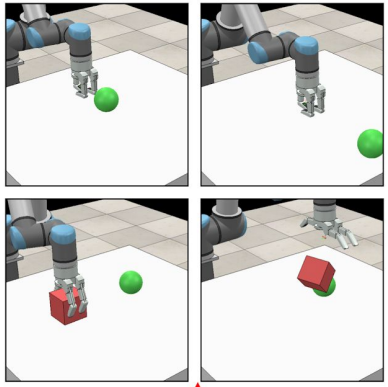


Step (II): Symbol formation

- One dimensional paired-object categories for varying sizes.
- When the below object is cup, the relation loosely resembles *is larger than*.

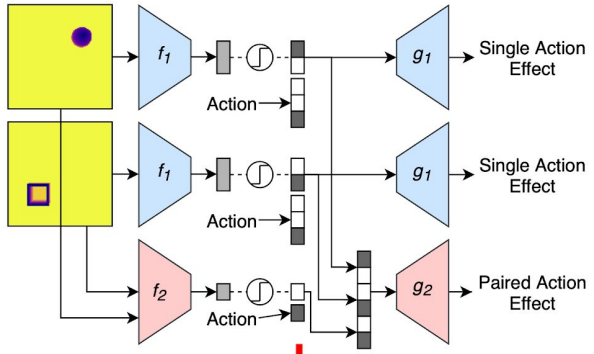


(I) Interaction with objects with pre-defined actions

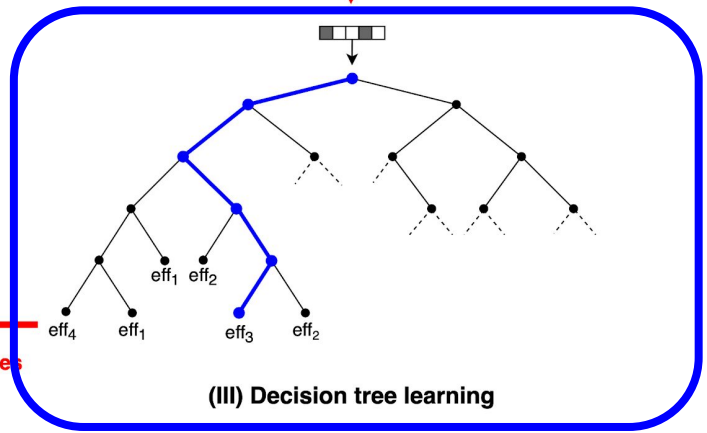


(II) Symbol formation (discovery of object and effect categories)

Low-level continuous object and effect observations



High-level discrete object and effect observations



(III) Decision tree learning

Probabilistic rules

Generated plans for verification

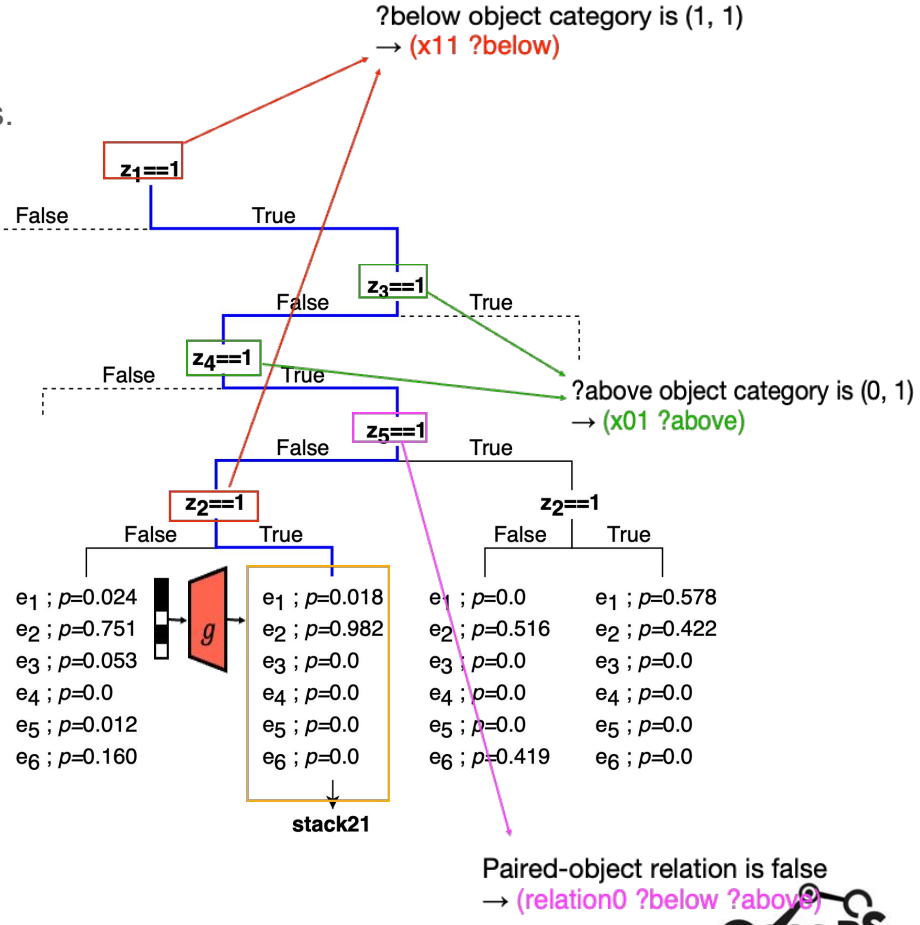
```
(:action stack23
:parameters (?below ?above)
:precondition (and (not (stacked)) (not (inserted)) (pickloc ?above)
                   (stackloc ?below) (objtype3 ?below) (objtype1 ?above)
                   (relation1 ?below ?above))
:effect (and (probabilistic
    0.578 (and (inserted) (instack ?above)
               (stackloc ?above) (not (stackloc ?below)))
    0.422 (and (stacked) (inserted) (instack ?above)
               (stackloc ?above) (not (stackloc ?below)))
    0.000 (roll1)
    0.000 (tumble1)
    0.000 (roll2)
    0.000 (tumble2))
    (not (pickloc ?above))))
```

(IV) Translation of rules to PPDDL operators

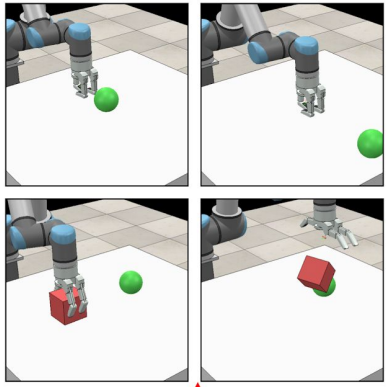
Step (III): Decision tree learning

- Transform the learned symbols into probabilistic rules.
- **Input:** object categories concatenated with action
- **Output:** generated effect (the decoder's output)
- 5 different symbols -> 2^5 possible effect categories
- Cluster effect categories for duplicate removal

```
(:action stack21
:parameters (?below ?above)
:precondition (and (not (aux-height) (not (aux-instack))
(pickloc ?above) (stackloc ?below)
(x11 ?below) (x01 ?above)
(relation0 ?below ?above))
:effect (and (probabilistic
0.018 (and (e1) (aux-height)
(stackloc ?above) (not (stackloc ?below)))
0.982 (and (e2) (aux-instack) (aux-height)
(stackloc ?above) (not (stackloc ?below)))
0.000 (e3)
0.000 (e4)
0.000 (e5)
0.000 (e6)
(not (pickloc ?above))))
(:action increaseheight1
:precondition (and (aux-height) (H0))
:effect (and (not (H0)) (H1) (not (aux-height))))
(:action increasestack1
:precondition (and (aux-instack) (S0))
:effect (and (not (S0)) (S1) (not (aux-instack))))
(:action makebase
:parameters (?obj)
:precondition (not (base))
:effect (and (base) (aux-height) (aux-instack)
(not (pickloc ?obj)) (stackloc ?obj)))
```

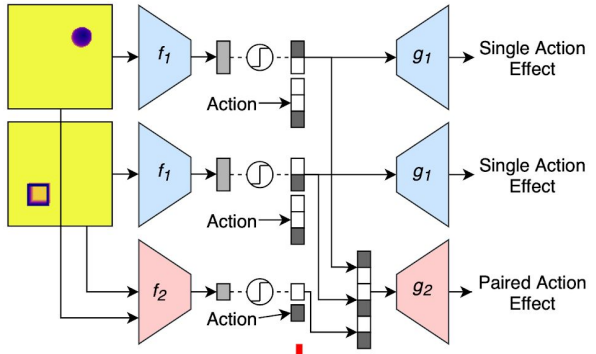


(I) Interaction with objects with pre-defined actions

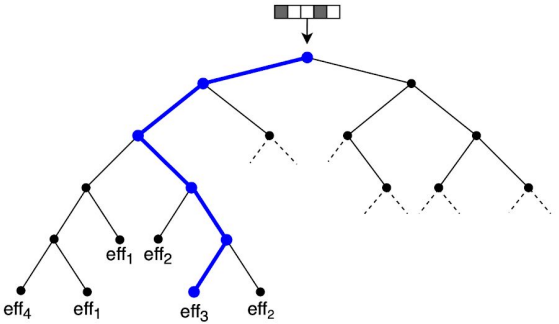


(II) Symbol formation (discovery of object and effect categories)

Low-level continuous object and effect observations



High-level discrete object and effect observations



Probabilistic rules

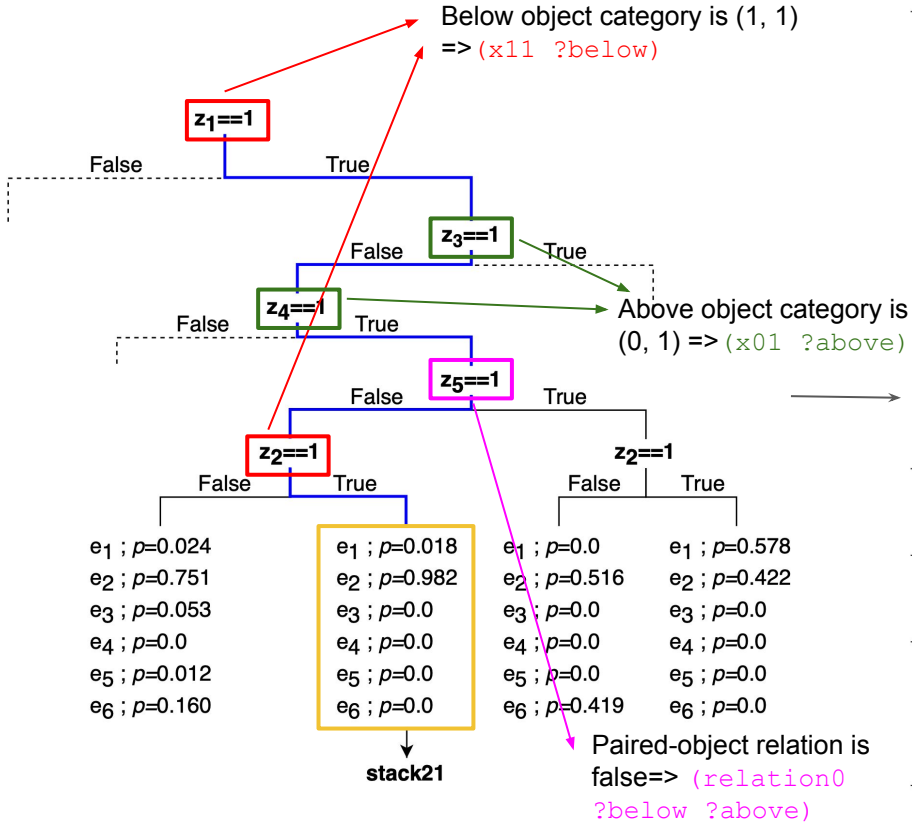
(III) Decision tree learning

Generated plans for verification

```
(:action stack23
:parameters (?below ?above)
:precondition (and (not (stacked)) (not (inserted)) (pickloc ?above)
                   (stackloc ?below) (objtype3 ?below) (objtype1 ?above)
                   (relation1 ?below ?above))
:effect (and (probabilistic
    0.578 (and (inserted) (instack ?above)
               (stackloc ?above) (not (stackloc ?below)))
    0.422 (and (stacked) (inserted) (instack ?above)
               (stackloc ?above) (not (stackloc ?below)))
    0.000 (roll1)
    0.000 (tumble1)
    0.000 (roll2)
    0.000 (tumble2))
    (not (pickloc ?above))))
```

(IV) Translation of rules to PPDDL operators

Step (IV): Translation of rules to PPDDL operators




```
(:action stack21
:parameters (?below ?above)
:precondition (and (not (aux-height)) (not (aux-instack))
(pickloc ?above) (stackloc ?below)
(x11 ?below) (x01 ?above)
(relation0 ?below ?above))
:effect (and (probabilistic
0.018 (and (e1) (aux-height)
(stackloc ?above) (not (stackloc ?below)))
0.982 (and (e2) (aux-instack) (aux-height)
(stackloc ?above) (not (stackloc ?below)))
0.000 (e3)
0.000 (e4)
0.000 (e5)
0.000 (e6)
(not (pickloc ?above))))
```

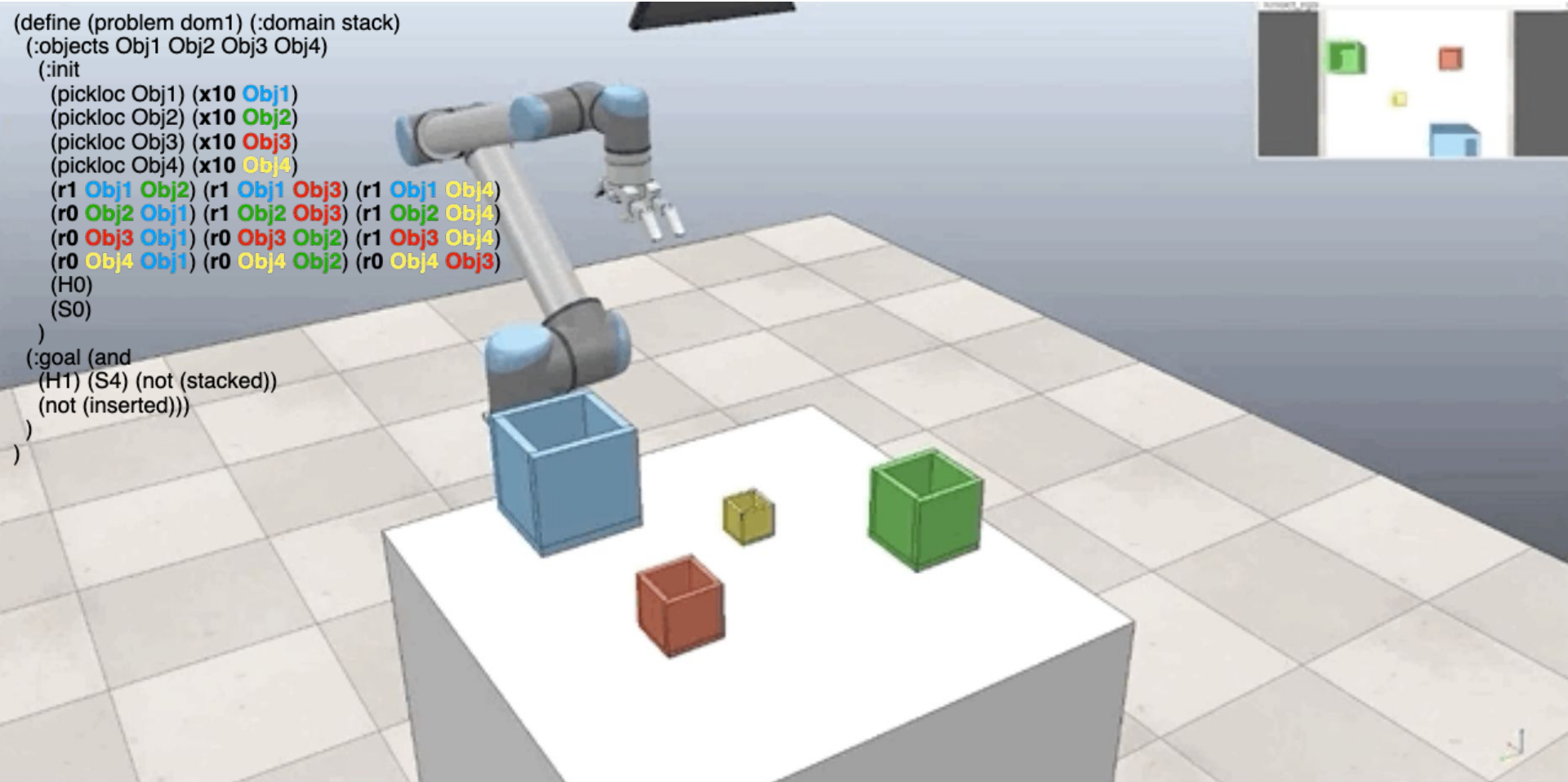
```
(:action increaseheight1
:precondition (and (aux-height) (H0))
:effect (and (not (H0)) (H1) (not (aux-height))))
```

```
(:action increasestack1
:precondition (and (aux-instack) (S0))
:effect (and (not (S0)) (S1) (not (aux-instack))))
```

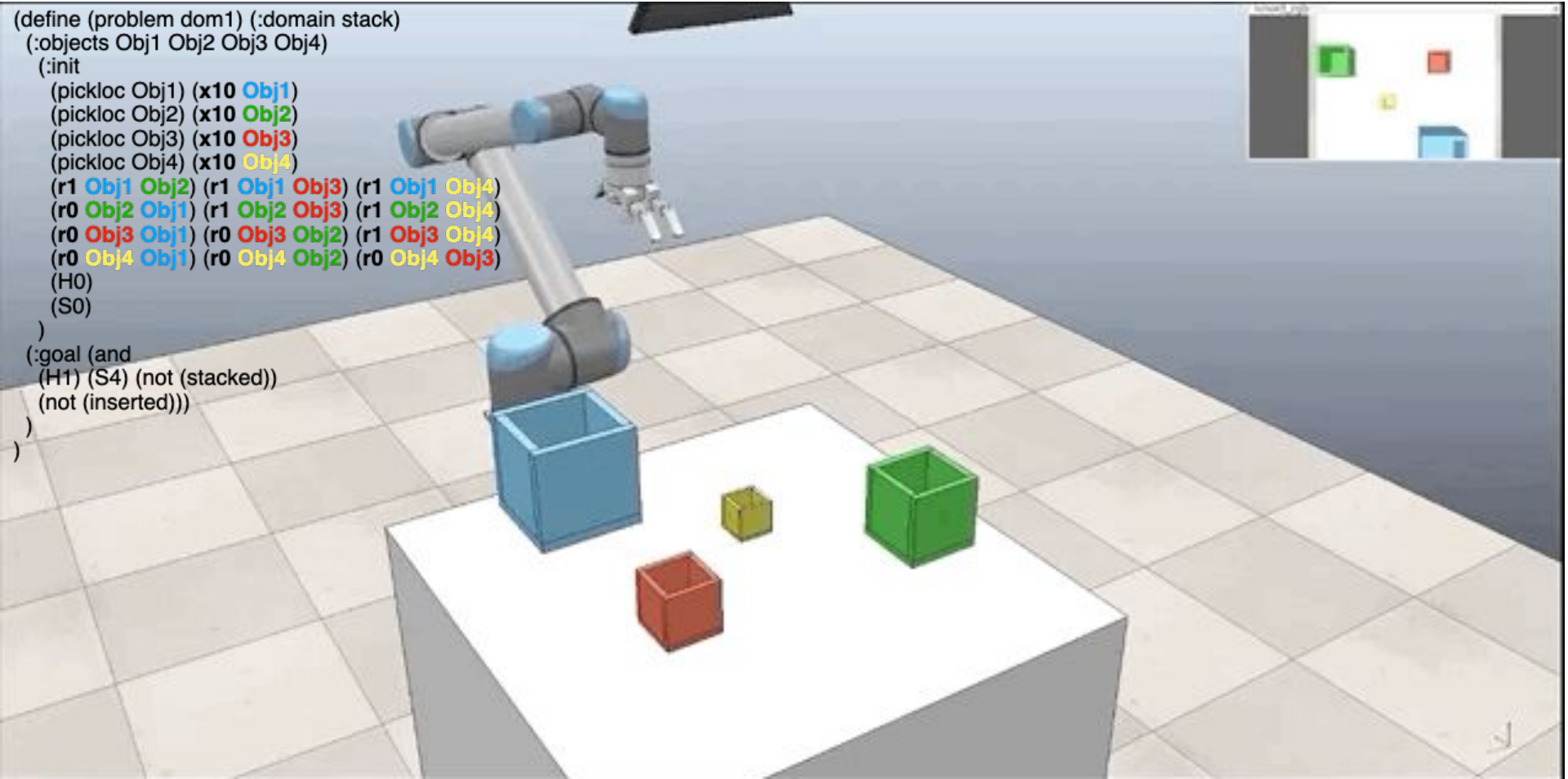
```
(:action makebase
:parameters (?obj)
:precondition (not (base))
:effect (and (base) (aux-height) (aux-instack)
(not (pickloc ?obj)) (stackloc ?obj)))
```

Objective: A tower with height=1 using 4 objects

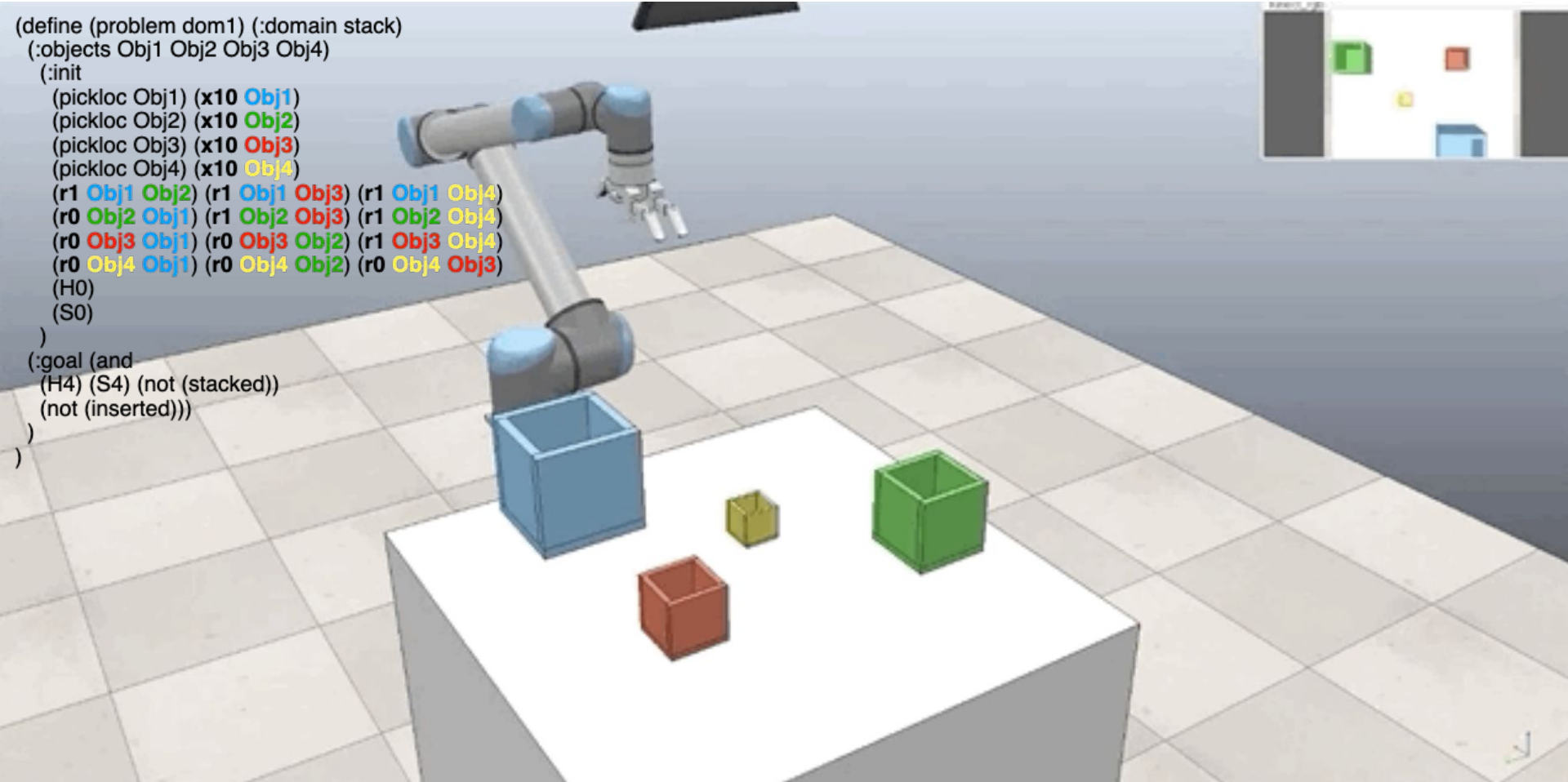
```
(define (problem dom1) (:domain stack)
  (:objects Obj1 Obj2 Obj3 Obj4)
    (:init
      (pickloc Obj1) (x10 Obj1)
      (pickloc Obj2) (x10 Obj2)
      (pickloc Obj3) (x10 Obj3)
      (pickloc Obj4) (x10 Obj4)
      (r1 Obj1 Obj2) (r1 Obj1 Obj3) (r1 Obj1 Obj4)
      (r0 Obj2 Obj1) (r1 Obj2 Obj3) (r1 Obj2 Obj4)
      (r0 Obj3 Obj1) (r0 Obj3 Obj2) (r1 Obj3 Obj4)
      (r0 Obj4 Obj1) (r0 Obj4 Obj2) (r0 Obj4 Obj3)
      (H0)
      (S0)
    )
  (:goal (and
    (H1) (S4) (not (stacked))
    (not (inserted)))
  )
)
```



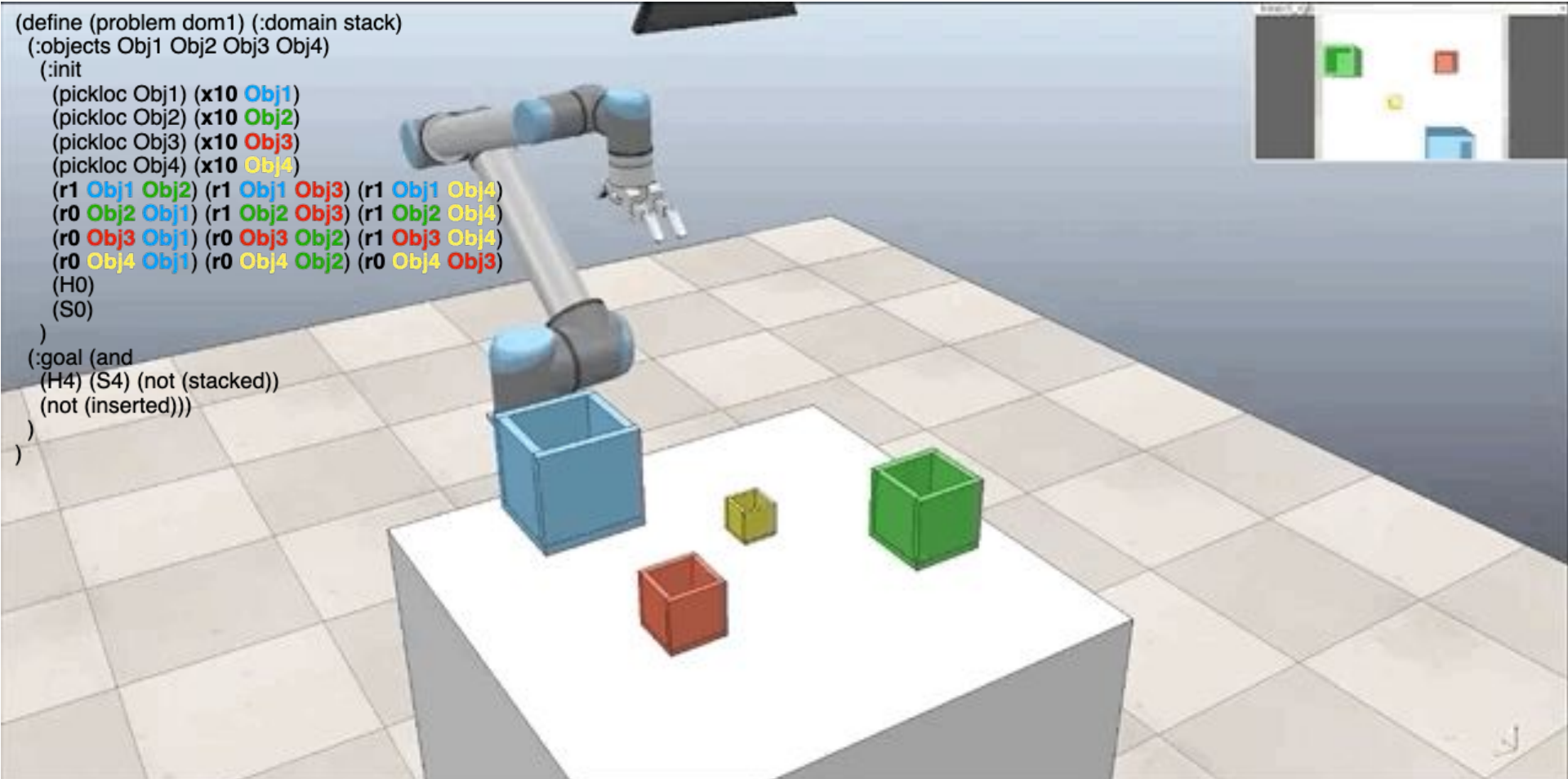
Objective: A tower with height=1 using 4 objects



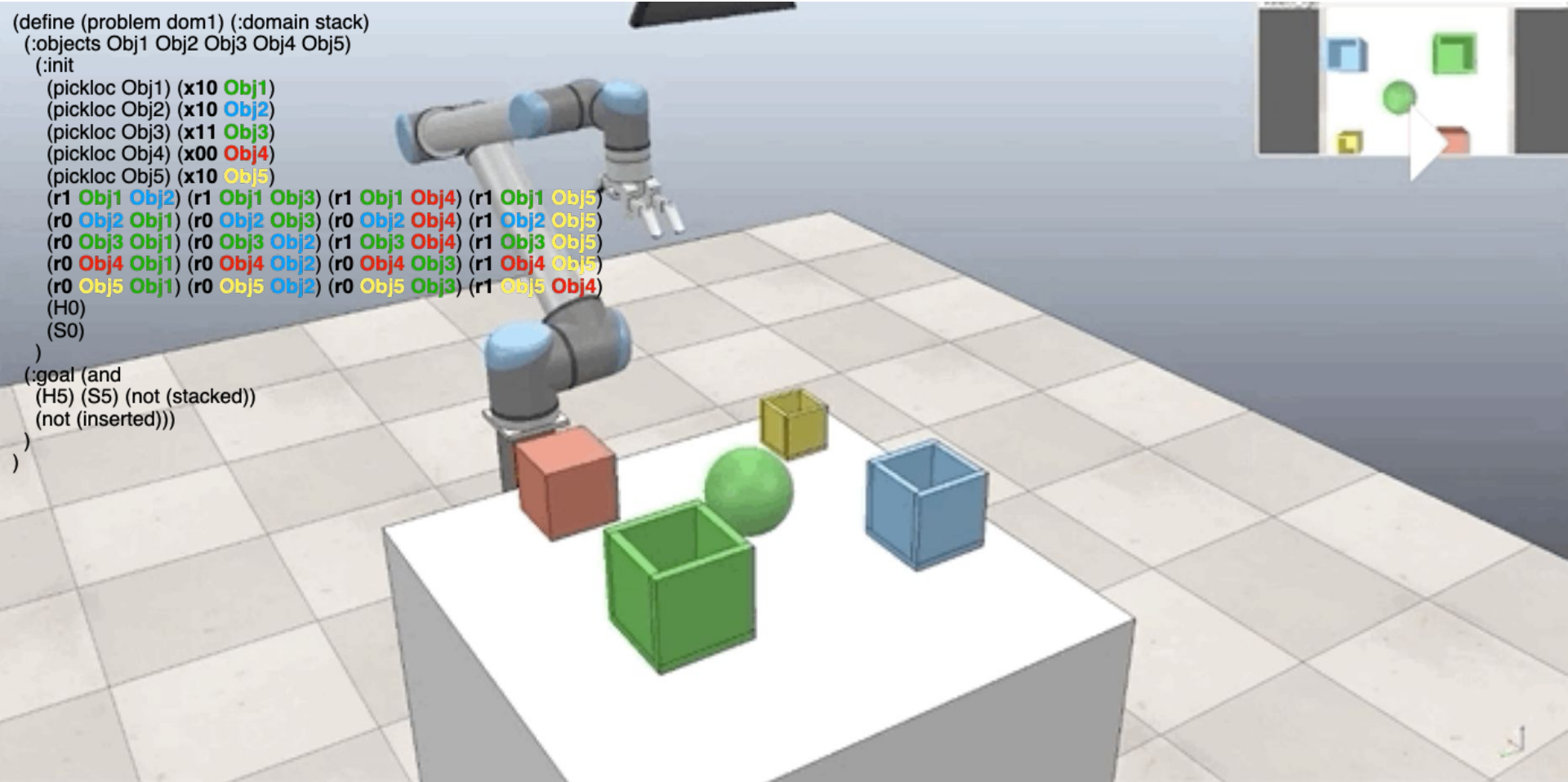
Objective: A tower with height=4 using 4 objects



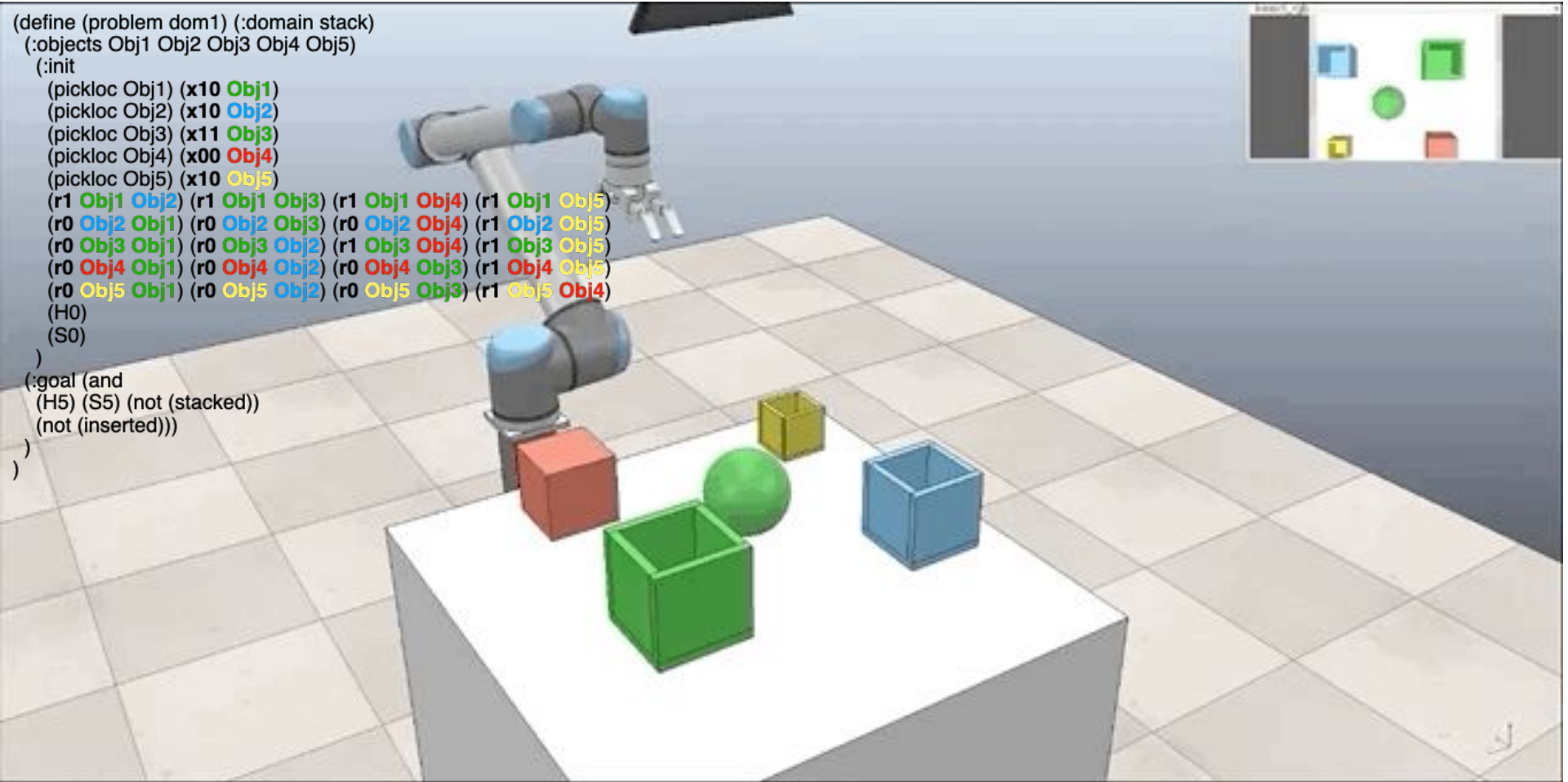
Objective: A tower with height=4 using 4 objects

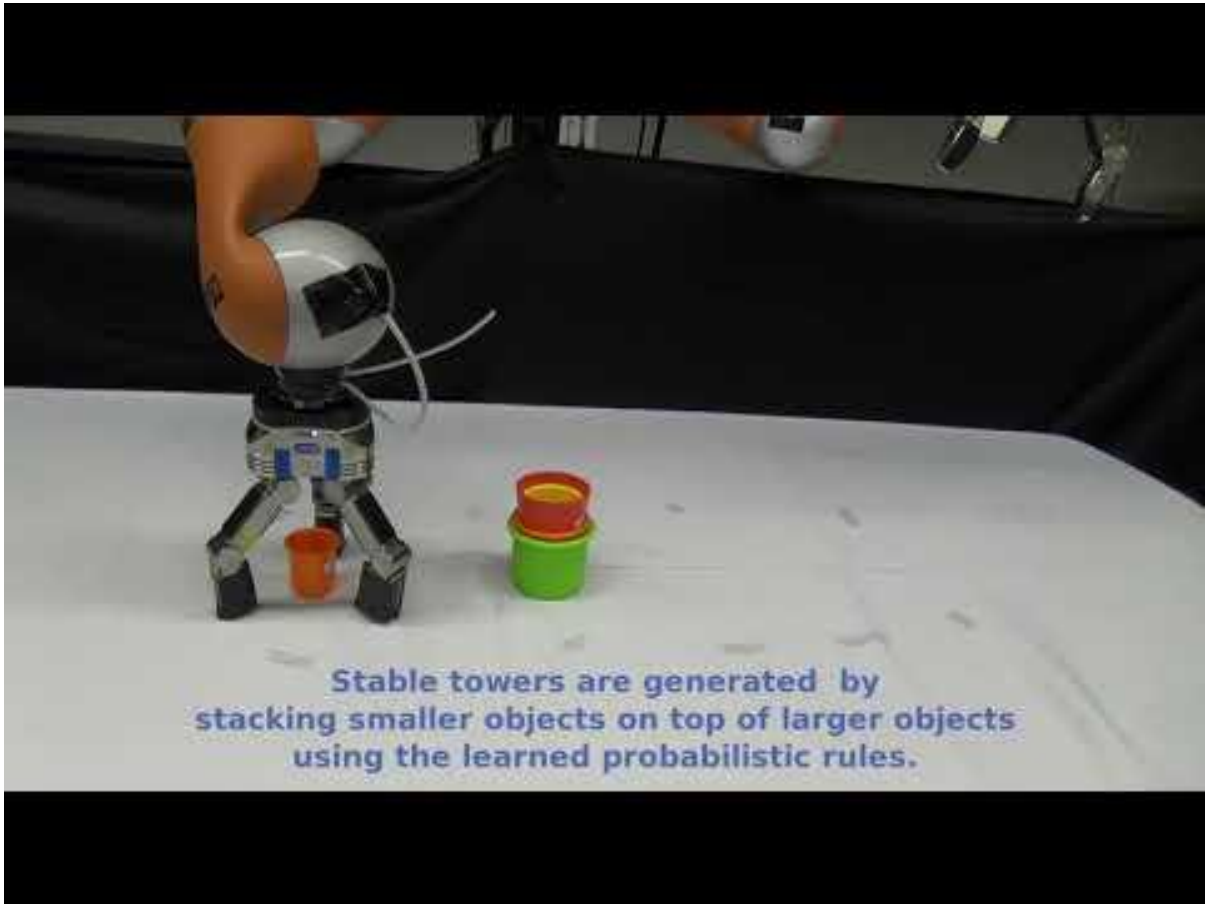


Objective: A tower with height=5 using 5 objects



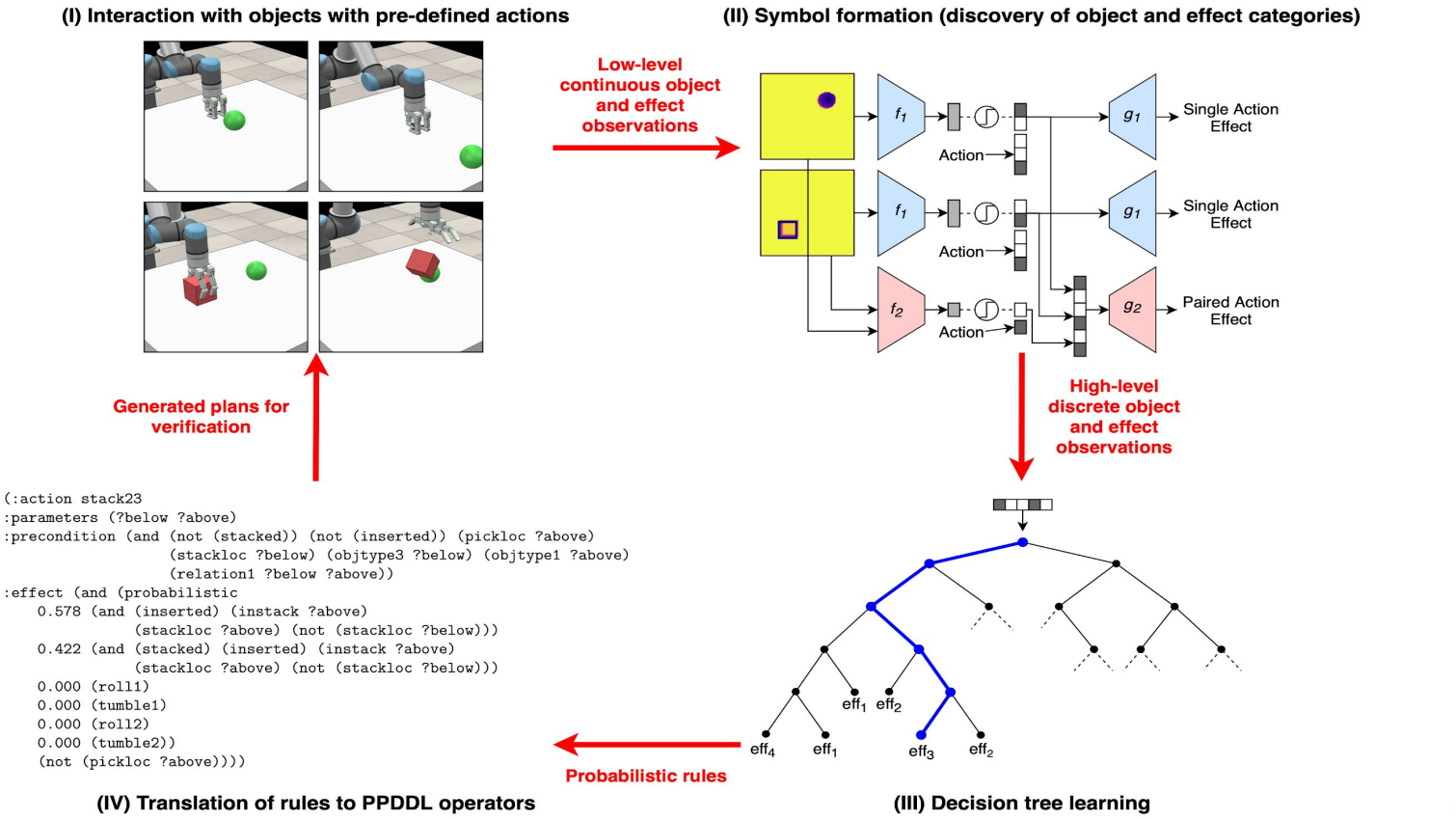
Objective: A tower with height=5 using 5 objects





E. Ugur, J. Piater, **Bottom-Up Learning of Object Categories, Action Effects and Logical Rules: From Continuous Manipulative Exploration to Symbolic Planning**, IEEE Intl. Conf. on Robotics and Automation **(ICRA)**, pp. 2627-2633, 2015.
E. Ugur, J. Piater, **Refining discovered symbols with multi-step interaction experience**, IEEE-RAS Intl. Conf. on Humanoid Robotics , pp. 1007-1012, 2015.

What are the limitations of this architecture?



What are the limitations of this architecture?

1. Learn symbols of one or two objects
    - Require to discover symbols for **multiple objects** and relations
2. Assume pre-defined discrete actions
    - Require to find **discrete action symbols** in continuous motor space.