

# AI 驱动 软件研发 全面进入数字化时代

中国·深圳 11.24-25

AI+  
software  
Development  
Digital  
summit



# 面向复杂问题的智能自主设计探索

胡杏 中国科学院计算技术研究所研究员

# 科技生态圈峰会 + 深度研习



—1000+ 技术团队的选择



K+全球软件研发行业创新峰会

会议时间: 2024.05.24-25



K+全球软件研发行业创新峰会

会议时间: 2024.09.20-21



AI+ 软件研发数字峰会

会议时间: 2023.11.24-25



AI+ 软件研发数字峰会

会议时间: 2024.07.19-20



AI+ 软件研发数字峰会

会议时间: 2024.11.15-16

# ▶ 演讲嘉宾



## 胡杏

中国科学院计算技术研究所 研究员

---

博士生导师，主要研究方向为高效安全的智能计算系统以及基于智能技术的计算系统自动化构建。多次担任MICRO、DAC等体系结构和人工智能旗舰会议的程序委员会委员，获得国家级和中国科学院级青年人才计划。

# 目录

## CONTENTS

1. 智能自主设计：从计算到设计
2. 智能自主设计的方法和挑战
3. 极致开放的智能自主设计：开放场景编程
4. 展望

## **PART 01**

# **智能自主设计背景：从计算到设计**

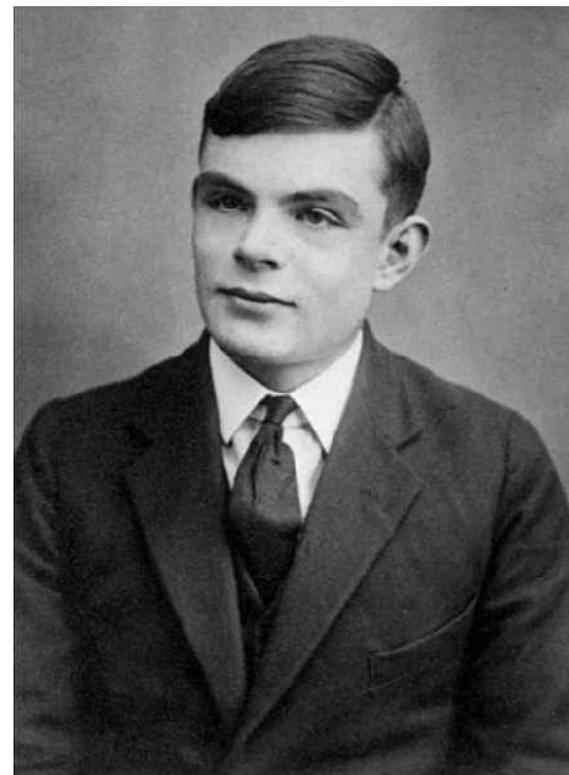
# ▶ 计算问题理论基础

## 丘奇图灵论题

Alonzo Church: Lambda演算

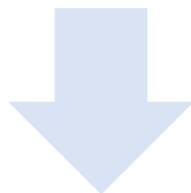


Allen Turing: 图灵机



可有效计算的问题

1. 有限指令
2. 有限步骤结束, 得到正确的答案



函数演算  
过程

侧重语言和逻辑

=

图灵机  
计算过程

侧重物理和机器

# 解决可计算问题的过程

可有效计算的问题

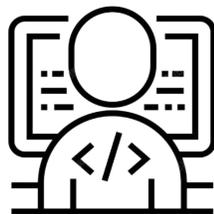
问题描述 I/O数据输入 验证条件



问题刻画

定义自然语言描述的功能和设计约束集合 $X$ 、验证条件 $R(x, y)$

基于Machine OP的程序



程序设计

构建映射 $F: X \rightarrow Y$ ,  
使得 $\forall x \in X, R(x, F(x))$ 成立  
**可执行、可验证、可理解、可修正**

物理可执行的计算空间  
(Machine OP)



物理计算

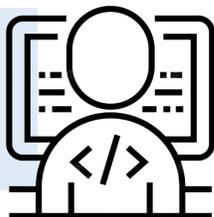
可执行的操作集合 $Y$

# ▶ 从计算到设计

可有效计算的问题

问题描述 I/O数据输入 验证条件

基于Machine OP的程序



物理可执行的计算空间  
(Machine OP)

是否可以  
智能自主设计?



可有效计算的问题

问题描述 I/O数据输入 验证条件



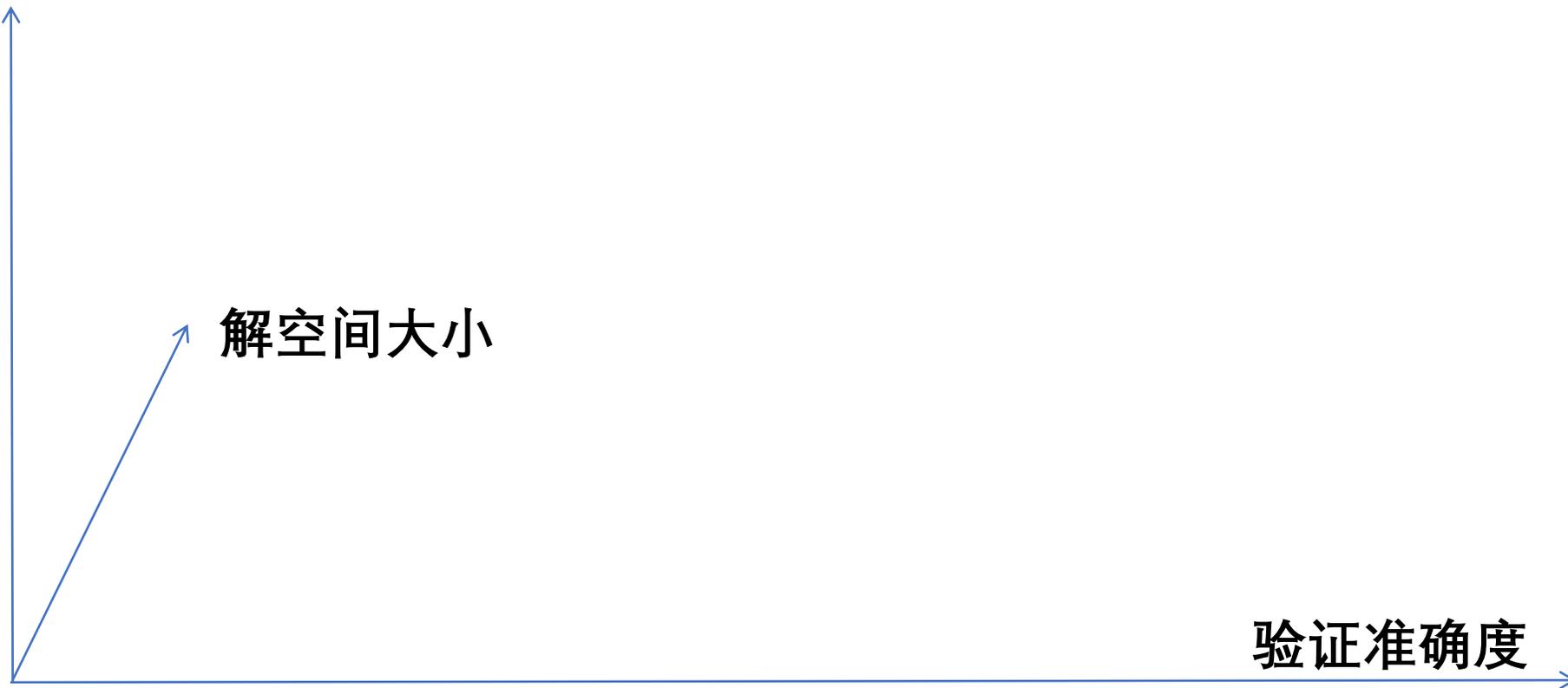
DIGITALAGENT

生成基于machine  
OP的程序

Machine可执行的OP空间

## 来自现实场景任务的挑战

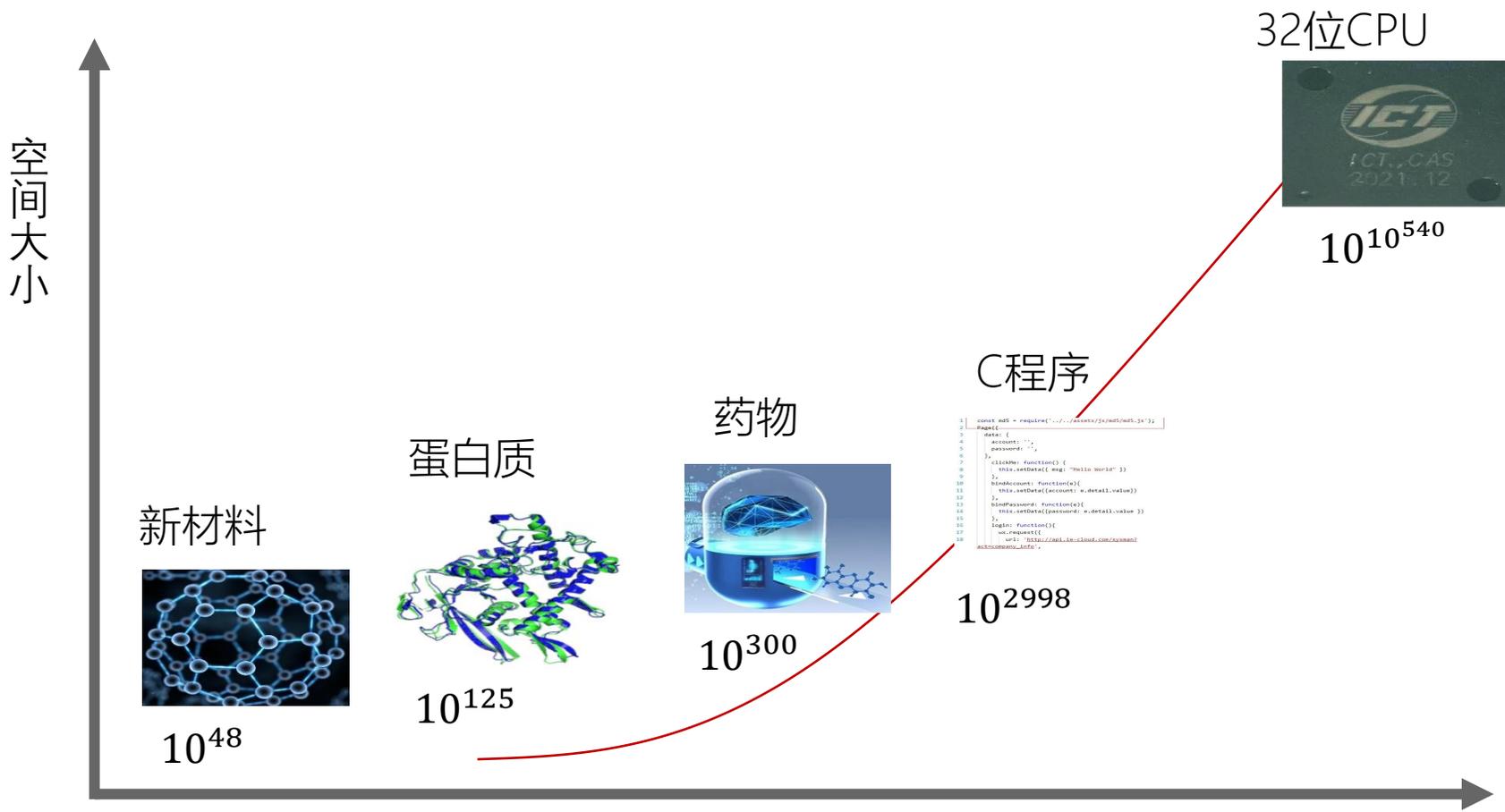
描述抽象程度



评价任务复杂性的几个维度

# 智能自主设计的挑战：来自现实场景中的任务

## 超大的解空间



# ▶ 智能自主设计的挑战：来自现实场景中的任务

## 超高的准确度要求



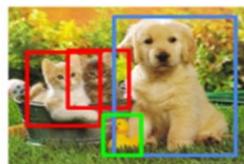
需要运行10亿次测试（每条测试有1万条指令）才能保证 Intel P4 CPU芯片**99.9999999999%**的准确率

VS



CAT

图像分类  
~90%



CAT, DOG, DUCK

目标检测  
~80%



语音识别  
~90%



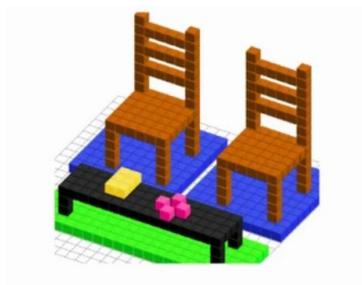
问答系统  
~90%

# 智能自主设计的挑战：来自现实场景中的任务

## 开放多样的抽象输入

	A	B	C
3	Phone Number	Formatted Number	
4	123456789	123-456-789	
5	789474763		
6	992784639		
7			
8			
9			
10			
11			
12			

add two chairs 5 spaces apart



炒一盘番茄鸡蛋



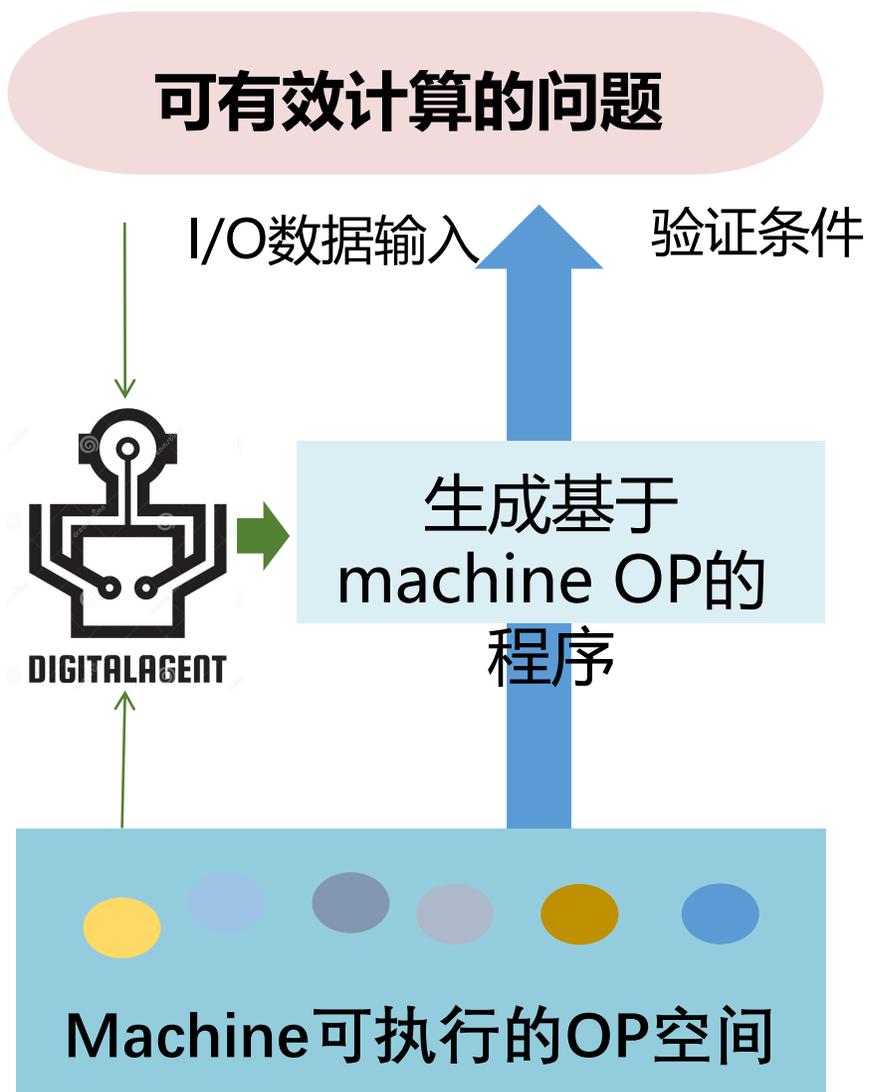
从用户的**输入输出样例**中生成程序

$B = A[:3] + '-' + A[3:6] + '-' + A[6:9]$

从用户的**自然语言描述**中生成程序

从与**环境的交互**中生成程序

# ▶ 传统程序合成的限制



传统程序合成的成功依赖于

**明确且有意义的约束**

一般要求可以通过SMT验证

**Heuristic搜索算法**

自顶向下、自底向上、剪枝、  
搜索空间表示.....

**精心设计的DSL/Grammar**

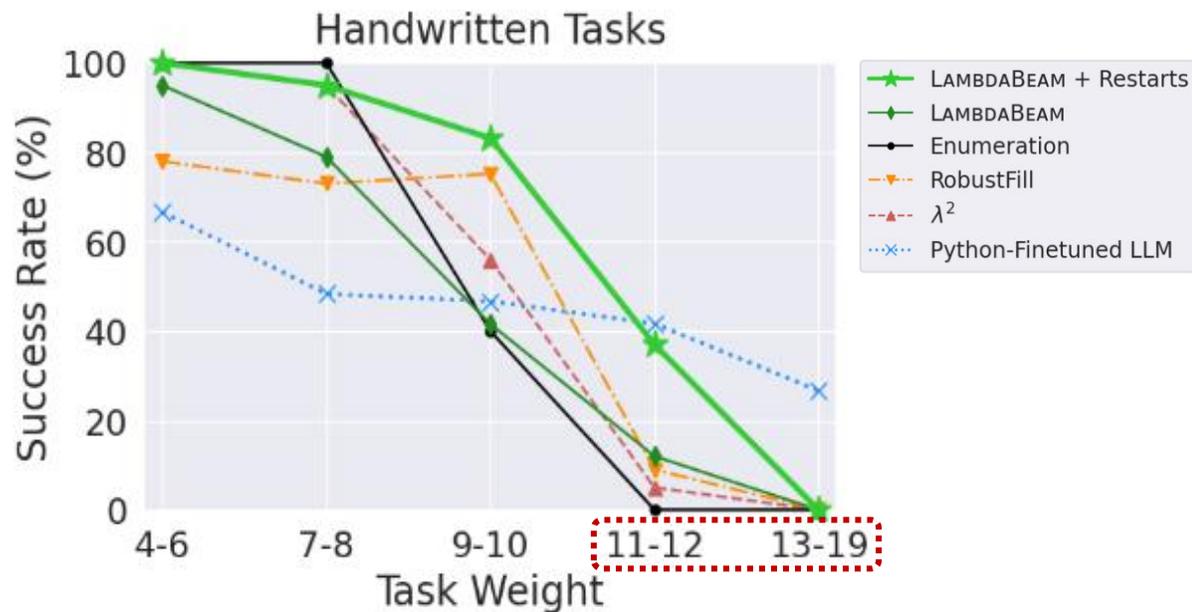
DSL设计与程序的表达能力和  
搜索空间大小紧密相关

# ▶ 传统程序合成的限制

传统程序合成的程序规模远远小于实际场景里的问题规模

Program	IO example
<code>s ← [int]</code>	<i>Input:</i> <code>[1 2 4 5 7]</code>
<code>b ← REVERSE s</code>	
<code>c ← ZIPWITH (-) b s</code>	<i>Output:</i> <code>9</code>
<code>d ← FILTER (&gt;0) c</code>	
<code>e ← SUM d</code>	

List processing 样例



生成正确率与语法树大小的关系 [Shi'23]

## PART 02

# 智能自主设计的方法和挑战

# ▶ 智能新时代的两大里程碑

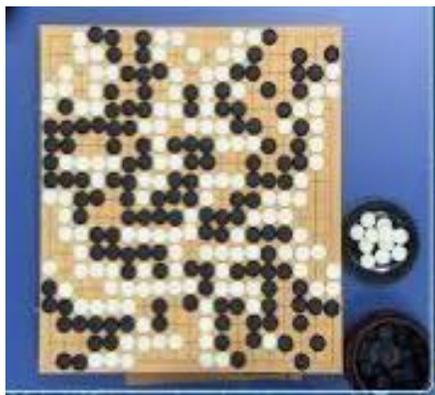
## 深度强化学习



AlphaGo

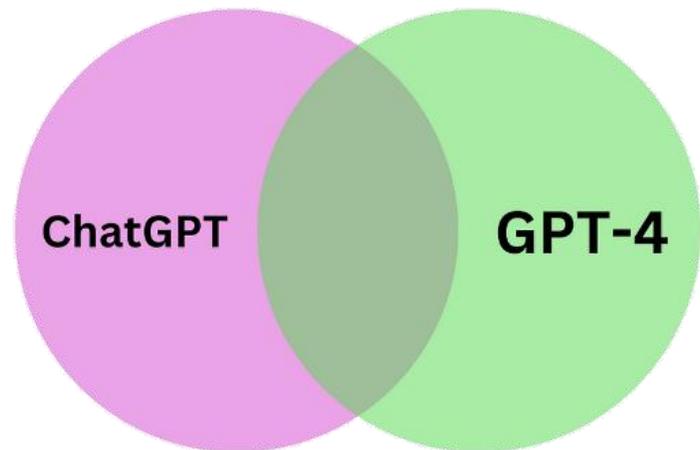
AlphaFold

AlphaTensor



专精

## 大模型

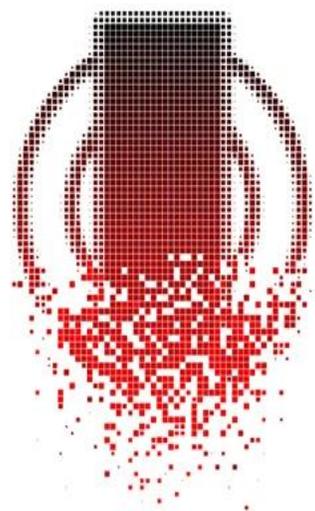
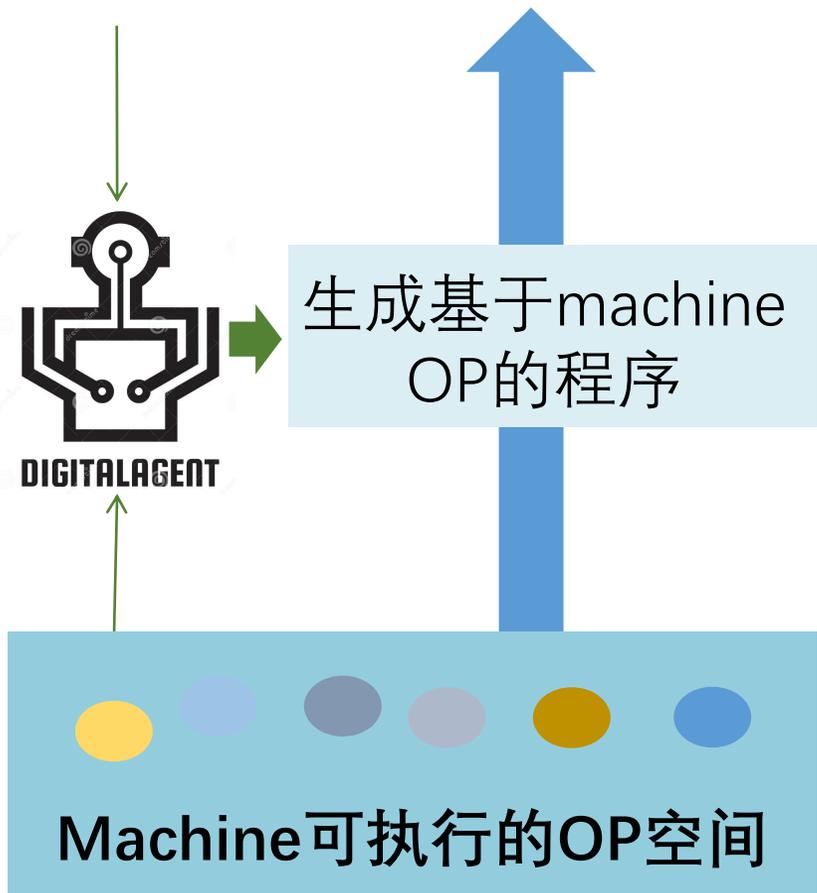


大众

# 智能自主设计的范式

可有效计算的问题

问题描述 I/O数据输入 验证条件



## 自顶向下

问题空间表示转换  
解决描述复杂度问题

问题空间语义分解  
减小搜索空间



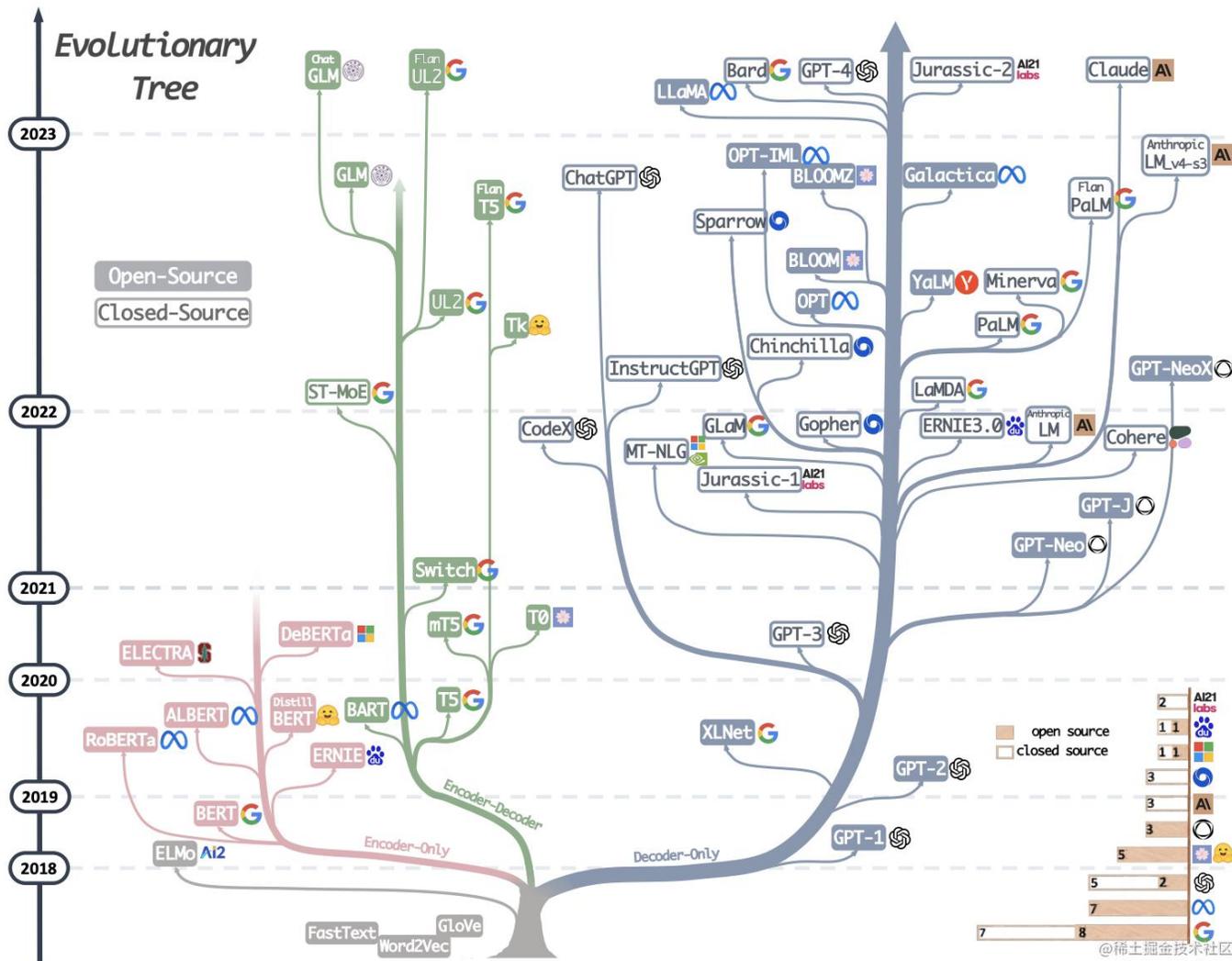
## 自底向上

搓工具  
构造功能更复杂、  
更多样的可计算载体

快搜索  
优化搜索方法寻找  
更有价值的搜索空间



# 智能自主设计的范式——自顶向下

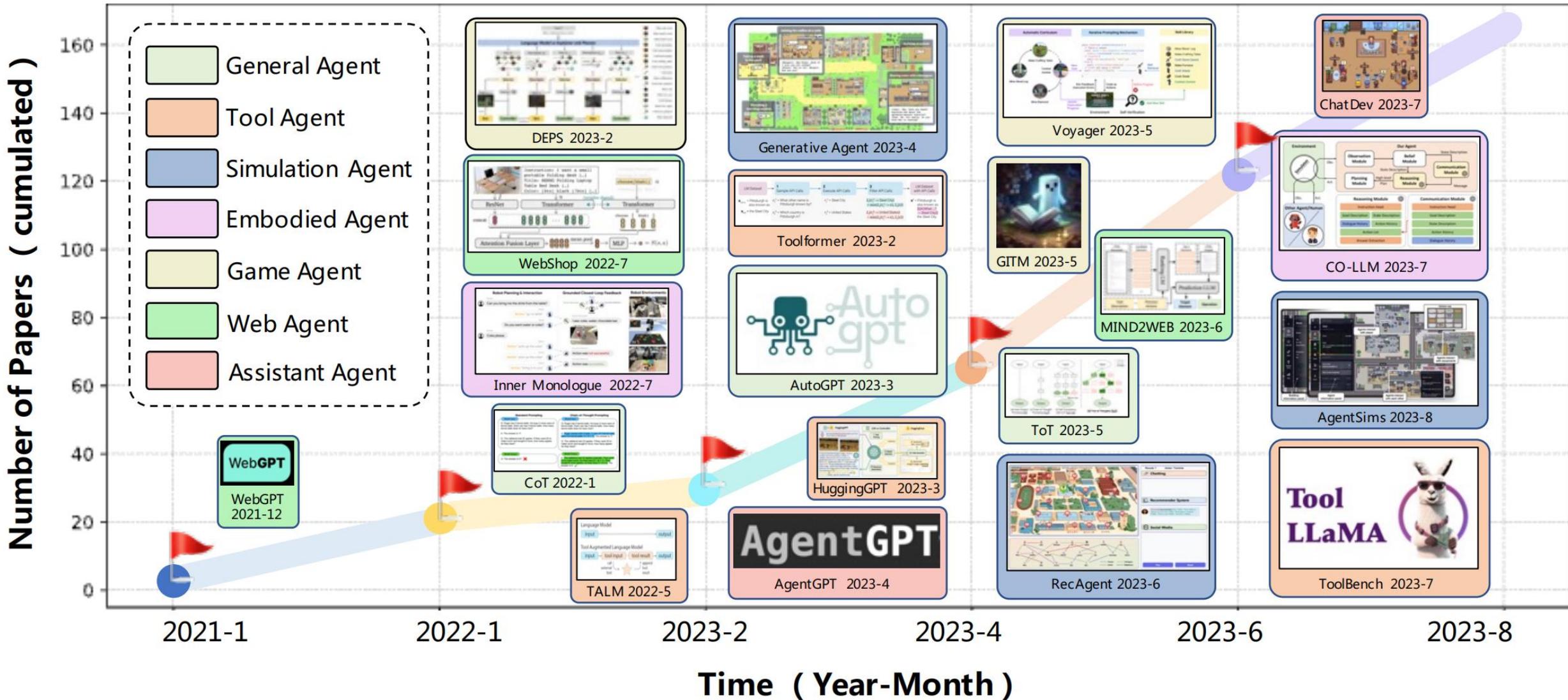


能自顶向下来解决问题的核心驱动：LLM！

语言表征学习

人类通识经验

# 基于LLMs的智能体设计



## ▶ 智能自主设计的范式——自顶向下

大模型擅长任务分解，尤其是流程类的任务分解

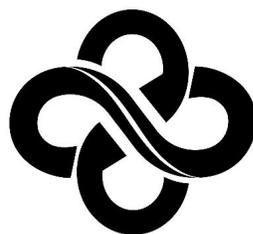


Andrej Karpathy ✓

@karpathy



Next frontier of prompt engineering imo: "AutoGPTs". 1 GPT call is just like 1 instruction on a computer. They can be strung together into programs. Use prompt to define I/O device and tool specs, define the cognitive loop, page data in and out of context window, .run().

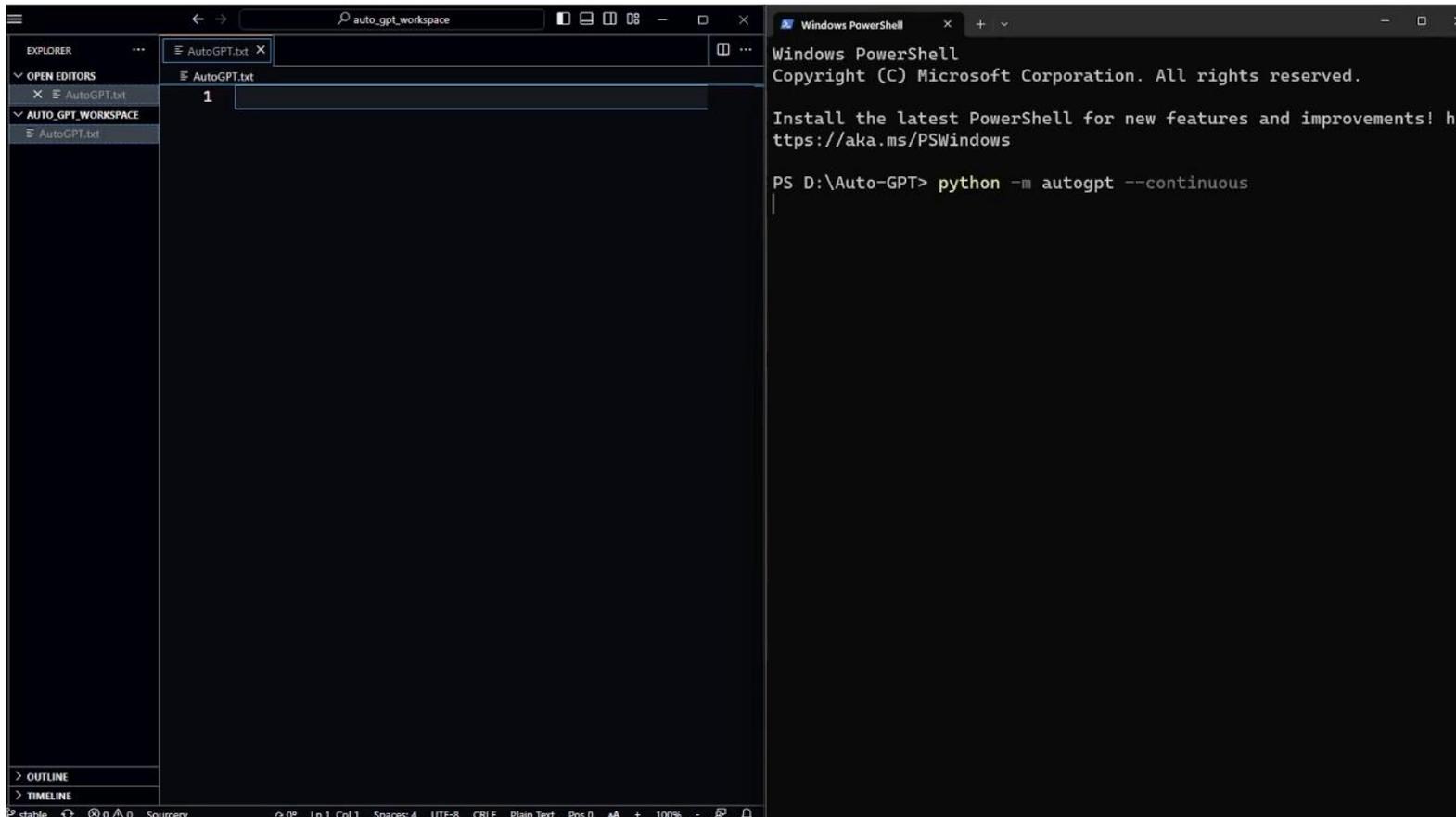
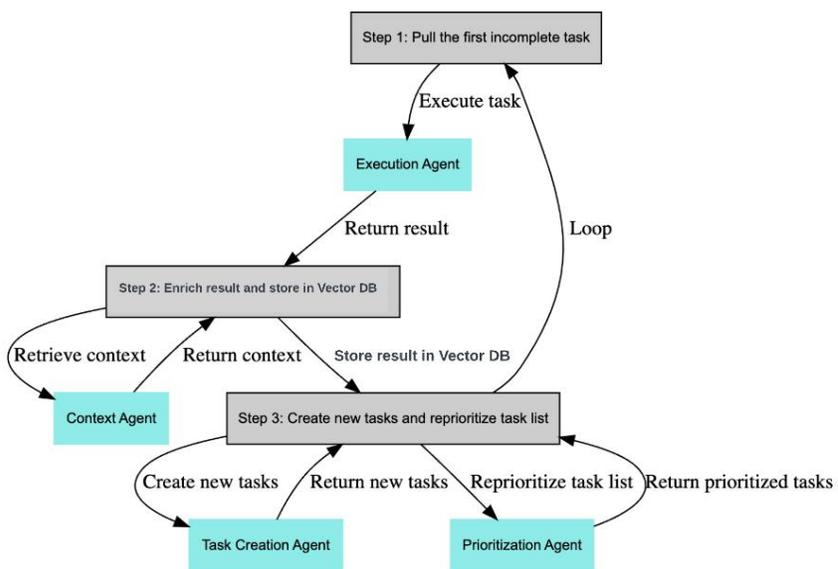


以AutoGPT为代表的大模型解决方案

# 智能自主设计的范式——自顶向下

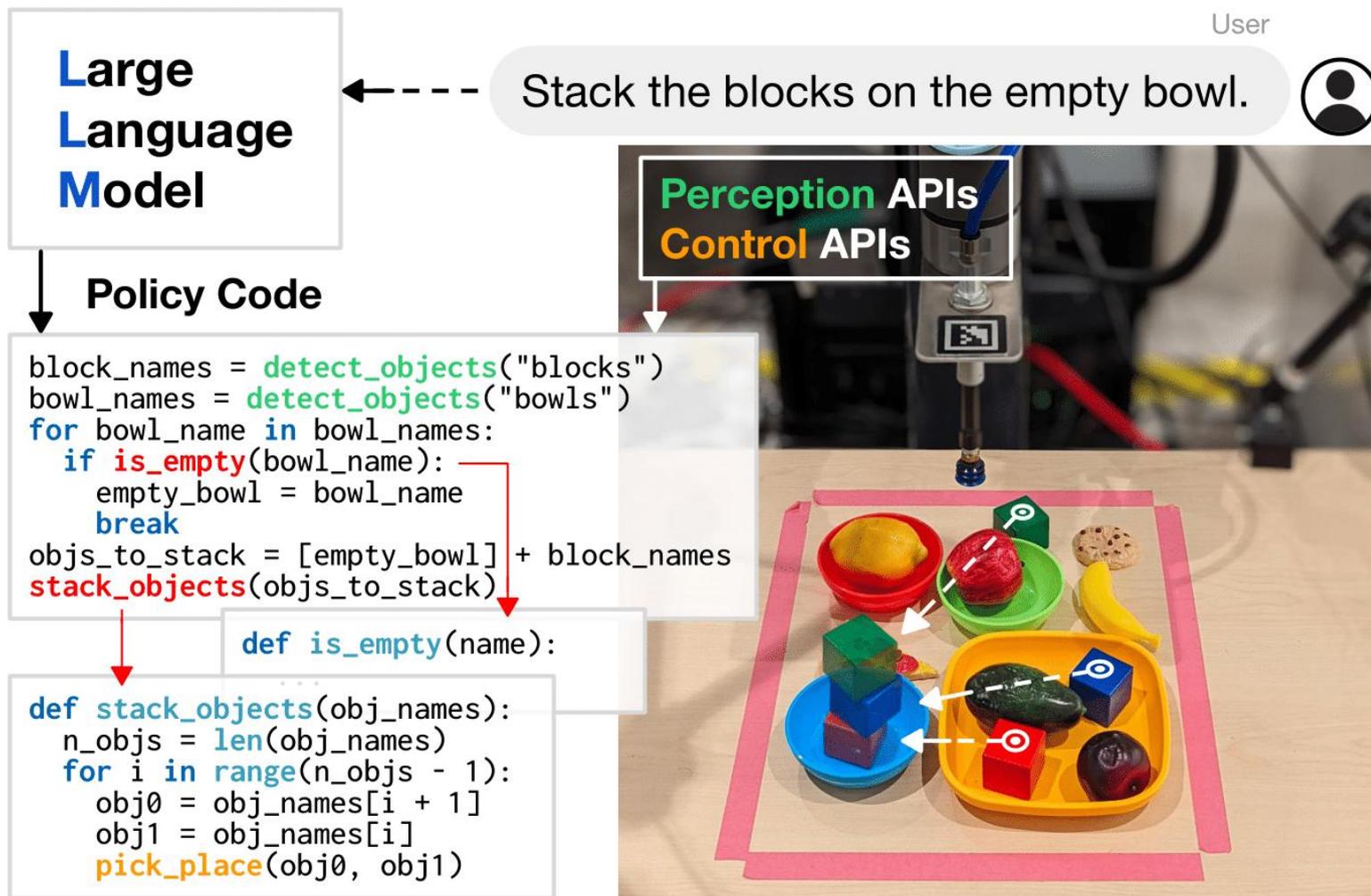
## 以AutoGPT为代表的大模型解决方案

让AutoGPT查询资料学习AutoGPT



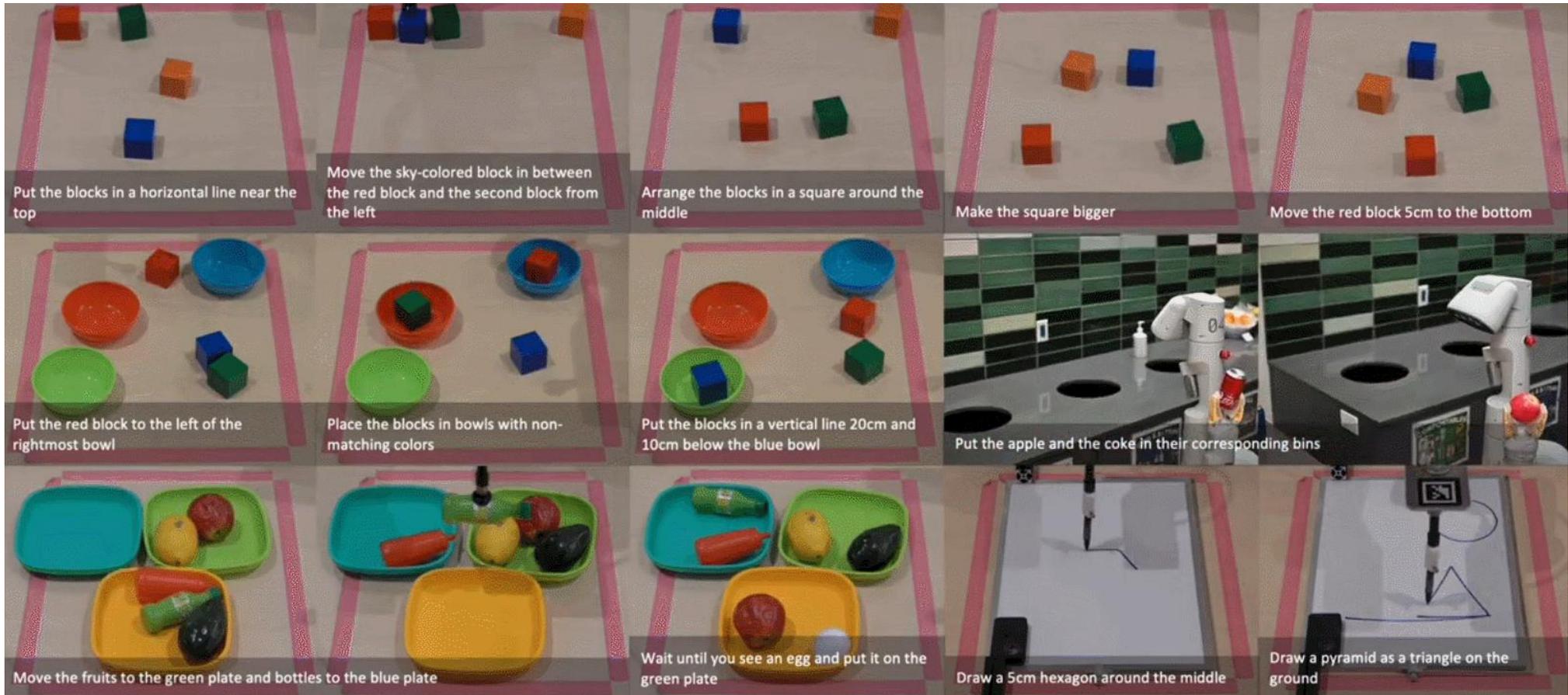
# 智能自主设计的范式——自顶向下

## 基于大模型和代码生成的自然语言机器控制



# ▶ 智能自主设计的范式——自顶向下

## 基于大模型和代码生成的自然语言机器控制



# ▶ 智能自主设计的范式——自顶向下

## 大模型分解的问题：语义世界的分解不可执行

在Minecraft中，大模型会产生无法执行的action



Stevan Harnad. 1990. The symbol grounding problem

<https://github.com/Significant-Gravitas/Auto-GPT-Plugins>

AutoGPT针对各种应用适配插件诞生

Plugin	Description	Repository
Alpaca-Trading	Trade stocks and crypto, paper or live with Auto-GPT	<a href="#">danikhan632/Auto-GPT-AlpacaTrader-Plugin</a>
AutoGPTReddit	Reddit Access	<a href="#">NeonN3mesis/AutoGPTReddit</a>
AutoGPT User Input Request	Allow Auto-GPT to specifically request user input in continous mode	<a href="#">HFrovinJensen/Auto-GPT-User-Input-Plugin</a>
BingAI	Enable Auto-GPT to fetch information via BingAI, saving time, API requests while maintaining accuracy. This does not remove the need for OpenAI API keys	<a href="#">gravelBridge/AutoGPT-BingAI</a>
Crypto	Trade crypto with Auto-GPT	<a href="#">isaiahbjork/Auto-GPT-Crypto-Plugin</a>
Discord	Interact with your Auto-GPT instance through Discord	<a href="#">gravelBridge/AutoGPT-Discord</a>
Dolly AutoGPT Cloner	A way to compose & run multiple Auto-GPT processes that cooperate, till core has multi-agent support	<a href="#">pr-0f3t/Auto-GPT-Dolly-Plugin</a>
Google Analytics	Connect your Google Analytics Account to Auto-GPT.	<a href="#">isaiahbjork/Auto-GPT-Google-Analytics-Plugin</a>
IFTTT webhooks	This plugin allows you to easily integrate IFTTT connectivity using Maker	<a href="#">AntonioCiolino/AutoGPT-IFTTT</a>
iMessage	Send and Get iMessages using Auto-GPT	<a href="#">danikhan632/Auto-GPT-Messages-Plugin</a>
Instagram	Instagram access	<a href="#">jpetzke/AutoGPT-Instagram</a>
Mastodon	Simple Mastodon plugin to send toots through a Mastodon account	<a href="#">ppetermann/AutoGPTMastodonPlugin</a>
		<a href="#">isaiahbjork/Auto-GPT-MetaTrader</a>

# 智能自主设计的范式——自顶向下

## 仍需要大量的人为抽象和手工设计

### AutoGPT针对各种应用适配插件

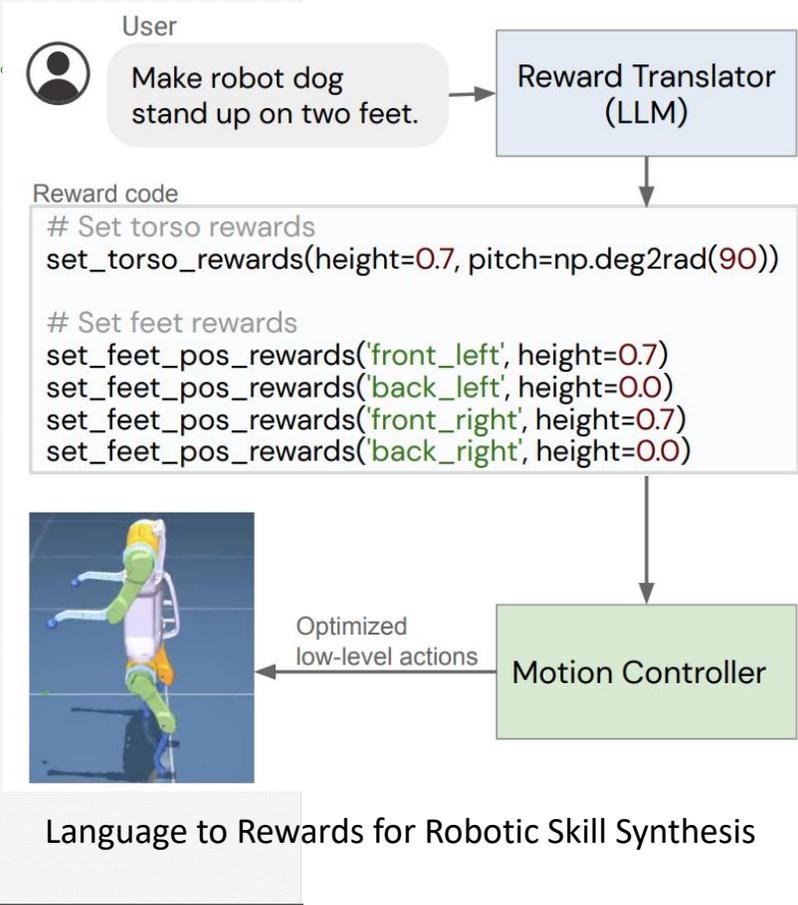
Plugin	Description	Repository
Alpaca-Trading	Trade stocks and crypto, paper or live with Auto-GPT	<a href="#">danikhan632/Auto-GPT-AlpacaTrader-Plugin</a>
AutoGPTReddit	Reddit Access	<a href="#">NeonN3mesis/AutoGPTReddit</a>
AutoGPT User Input Request	Allow Auto-GPT to specifically request user input in continuous mode	<a href="#">HFrovinJensen/Auto-GPT-User-Input-Plugin</a>
BingAI	Enable Auto-GPT to fetch information via BingAI, saving time, API requests while maintaining accuracy. This does not remove the need for OpenAI API keys	<a href="#">gravelBridge/AutoGPT-BingAI</a>
Crypto	Trade crypto with Auto-GPT	<a href="#">isaiahbjork/Auto-GPT-Crypto-Plugin</a>
Discord	Interact with your Auto-GPT instance through Discord	<a href="#">gravelBridge/AutoGPT-Discord</a>
Dolly AutoGPT Cloner	A way to compose & run multiple Auto-GPT processes that cooperate, till core has multi-agent support	<a href="#">pr-0f3t/Auto-GPT-Dolly-Plugin</a>
Google Analytics	Connect your Google Analytics Account to Auto-GPT.	<a href="#">isaiahbjork/Auto-GPT-Google-Analytics-Plugin</a>
IFTTT webhooks	This plugin allows you to easily integrate IFTTT connectivity using Maker	<a href="#">AntonioCiolino/AutoGPT-IFTTT</a>
iMessage	Send and Get iMessages using Auto-GPT	<a href="#">danikhan632/Auto-GPT-Messages-Plugin</a>
Instagram	Instagram access	<a href="#">jpetzke/AutoGPT-Instagram</a>
Mastodon	Simple Mastodon plugin to send toots through a Mastodon account	<a href="#">ppetermann/AutoGPTMastodonPlugin</a>

Code as policies中需要手工API实现 人为问题抽象和Reward API

#### LMP Prompts

```
prompt_tabletop_ui = '''
# Python 2D robot control script
import numpy as np
from env_utils import put_first_on_second, get_obj_pos, get_obj_names, say, get_corner_name, get_side
from plan_utils import parse_obj_name, parse_position, parse_question, transform_shape_pts

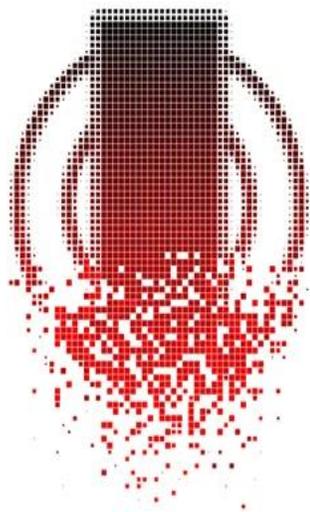
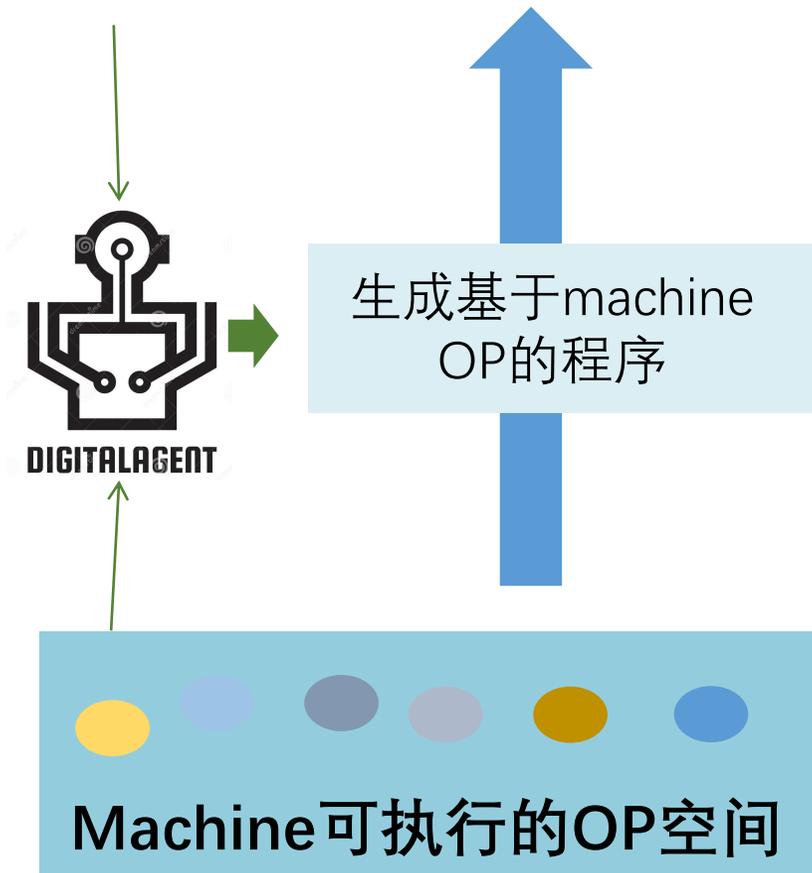
objects = ['yellow block', 'green block', 'yellow bowl', 'blue block', 'blue bowl', 'green bowl']
# the yellow block on the yellow bowl.
say('Ok - putting the yellow block on the yellow bowl')
put_first_on_second('yellow block', 'yellow bowl')
objects = ['yellow block', 'green block', 'yellow bowl', 'blue block', 'blue bowl', 'green bowl']
# which block did you move.
say('I moved the yellow block')
objects = ['yellow block', 'green block', 'yellow bowl', 'blue block', 'blue bowl', 'green bowl']
# move the green block to the top right corner.
say('Got it - putting the green block on the top right corner')
corner_pos = parse_position('top right corner')
put_first_on_second('green block', corner_pos)
objects = ['yellow block', 'green block', 'yellow bowl', 'blue block', 'blue bowl', 'green bowl']
# stack the blue bowl on the yellow bowl on the green block.
order_bottom_to_top = ['green block', 'yellow block', 'blue bowl']
say('Sure - stacking from top to bottom: {}'.format(' '.join(order_bottom_to_top)))
stack_objects_in_order(object_names=order_bottom_to_top)
objects = ['cyan block', 'white block', 'cyan bowl', 'blue block', 'blue bowl', 'white bowl']
# move the cyan block into its corresponding bowl.
matches = {'cyan block': 'cyan bowl'}
say('Got it - placing the cyan block on the cyan bowl')
for first, second in matches.items():
    put_first_on_second(first, get_obj_pos(second))
objects = ['cyan block', 'white block', 'cyan bowl', 'blue block', 'blue bowl', 'white bowl']
# make a line of blocks on the right side.
say('No problem! Making a line of blocks on the right side')
block_names = parse_obj_name('the blocks', f'objects = {get_obj_names()}')
line_pts = parse_position(f'a 30cm vertical line on the right with {len(block_names)} points')
for block_name, pt in zip(block_names, line_pts):
    put_first_on_second(block_name, pt)
objects = ['yellow block', 'red block', 'yellow bowl', 'gray block', 'gray bowl', 'red bowl']
# put the small banana colored thing in between the blue bowl and green block.
say('Sure thing - putting the yellow block between the blue bowl and the green block')
target_pos = parse_position('a point in the middle between the blue bowl and the green block')
put_first_on_second('yellow block', target_pos)
objects = ['yellow block', 'red block', 'yellow bowl', 'gray block', 'gray bowl', 'red bowl']
# can you cut the bowls in half.
say('no, I can only move objects around')
objects = ['yellow block', 'green block', 'yellow bowl', 'gray block', 'gray bowl', 'green bowl']
# stack the blocks on the right side with the gray one on the bottom.
say('Ok. stacking the blocks on the right side with the gray block on the bottom')
right_side = parse_position('the right side')
put_first_on_second('gray block', right_side)
order_bottom_to_top = ['gray block', 'green block', 'yellow block']
stack_objects_in_order(object_names=order_bottom_to_top)
objects = ['yellow block', 'green block', 'yellow bowl', 'blue block', 'blue bowl', 'green bowl']
```



# ▶ 智能自主设计的范式

可有效计算的问题

问题描述 I/O数据输入 验证条件



自顶向下

问题空间表示转换  
解决描述复杂度问题

问题空间语义分解  
减小搜索空间

问题：  
脱离环境、不可执行

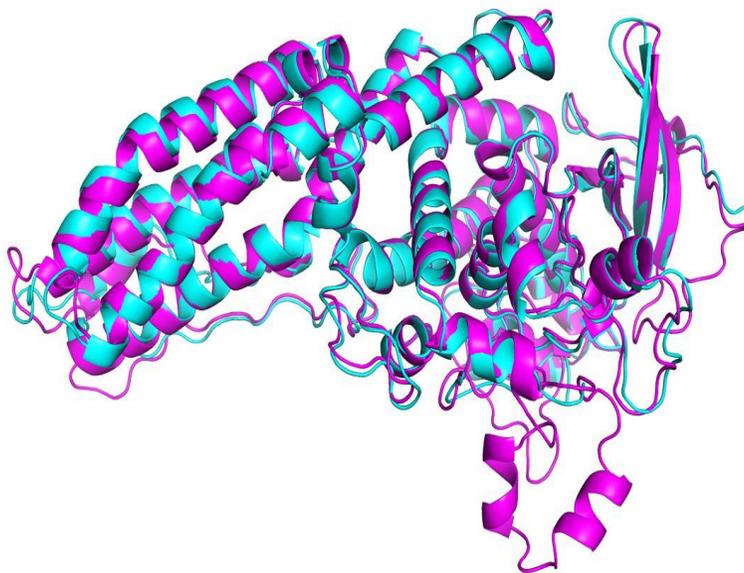
# ▶ 智能自主设计的范式——自底向上

## 深度强化学习的复杂问题探索

AlphaGo



AlphaFold



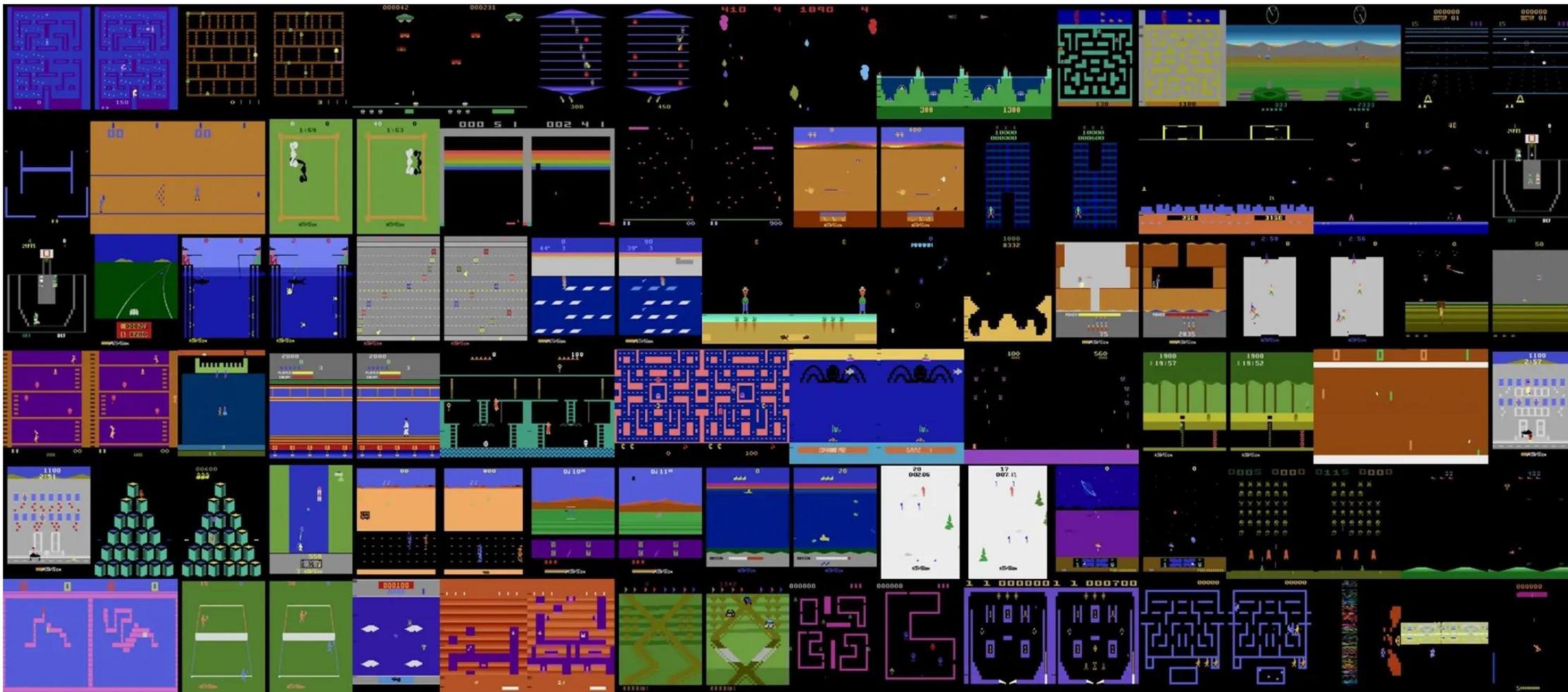
AlphaTensor



<https://www.deepmind.com/blog/agent57-outperforming-the-human-atari-benchmark>

# 智能自主设计的范式——自底向上

## 基于强化学习的游戏智能体

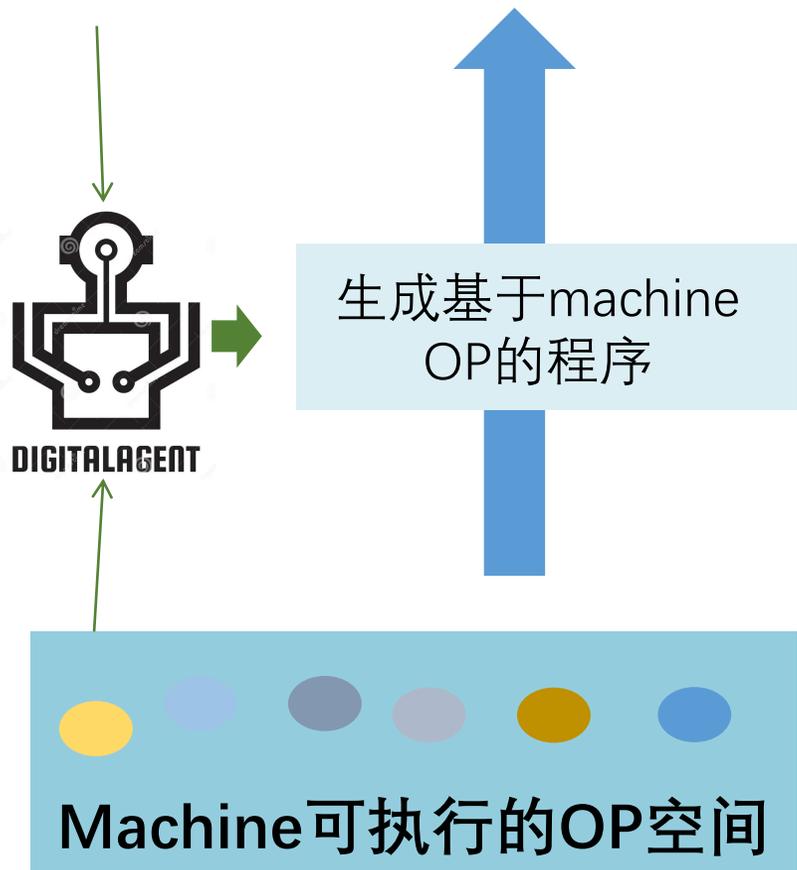


<https://www.deepmind.com/blog/agent57-outperforming-the-human-atari-benchmark>

# 智能自主设计的范式

## 可有效计算的问题

问题描述 I/O数据输入 验证条件



## 自顶向下

问题空间表示转换  
解决描述复杂度问题

问题空间语义分解  
减小搜索空间

无法保证正确、可执行、可验证

搜索效率低 解迁移性差

搓工具

构造功能更复杂、  
更多样的可计算载体

快搜索

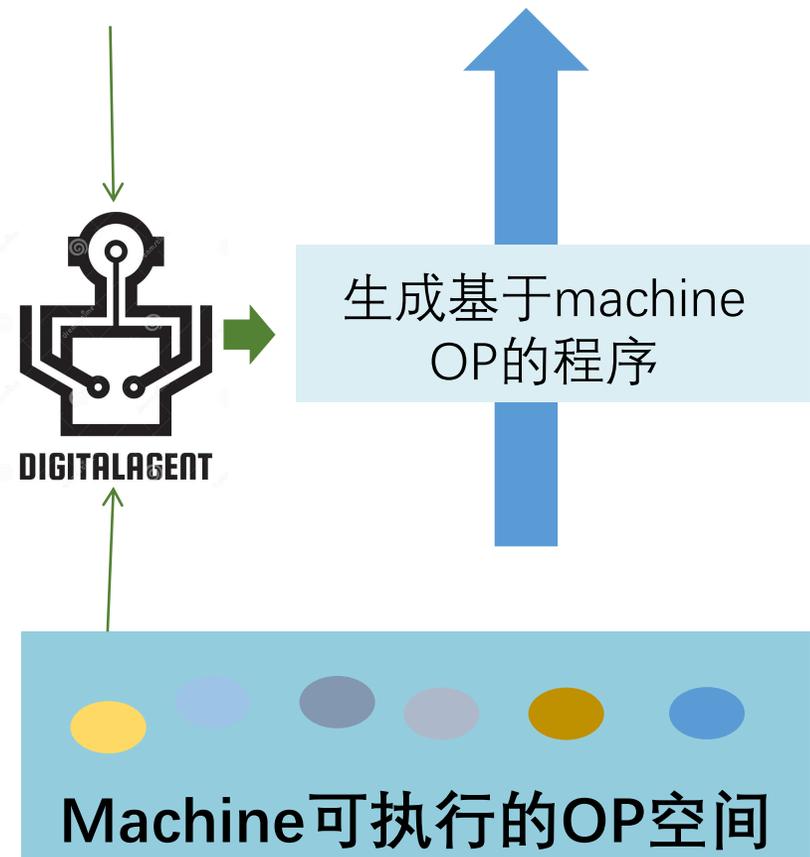
优化搜索方法寻找  
更有价值的搜索空间

## 自底向上

# 智能自主设计的范式

## 可有效计算的问题

问题描述 I/O数据输入 验证条件



## 自顶向下

问题空间表示转换  
解决描述复杂度问题

问题空间语义分解  
减小搜索空间

利用强先验

利用强搜索

搓工具  
构造功能更复杂、  
更多样的可计算载体

快搜索  
优化搜索方法寻找  
更有价值的搜索空间

## PART 03

# 极致开放度的智能自主设计任务： 开放场景编程

# ▶ 极致开放度要求的智能自主设计：开放编程

指令遵循 (Instruction Following) 通过自然语言指令交互  
使智能体在环境中完成指定任务



Build house



Shear  
Sheep

“我的世界”中指令遵循任务



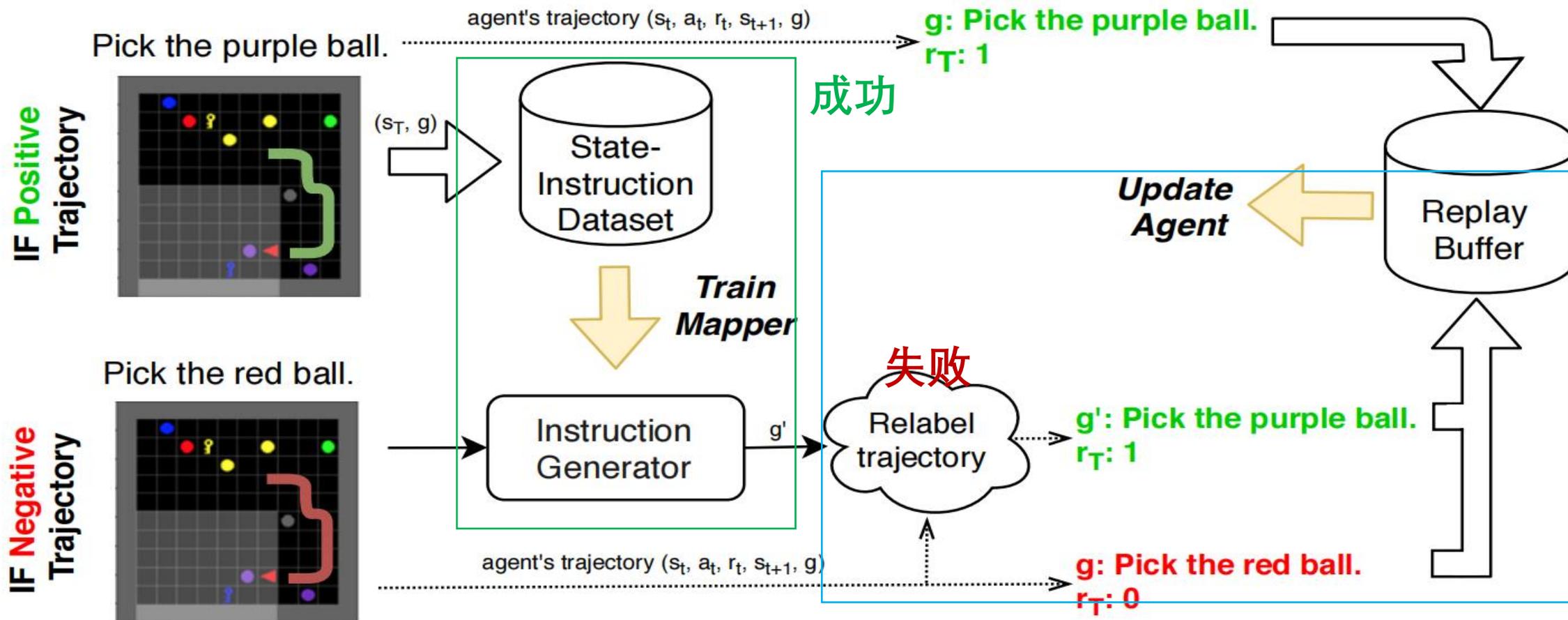
一些模拟环境机器人的指令遵循任务

## 关键挑战：语义世界和环境世界的鸿沟

# 自底向上的指令遵循：强化学习

关键挑战：稀疏奖励、空间大、轨迹长

数据重标



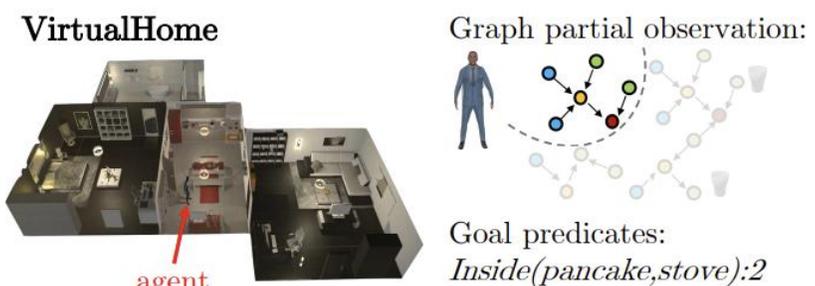
Hindsight Generation for Experience Replay (HIGhER)

# 自底向上的指令遵循：强化学习+模仿学习

关键挑战：稀疏奖励、空间大、轨迹长

← 模仿学习

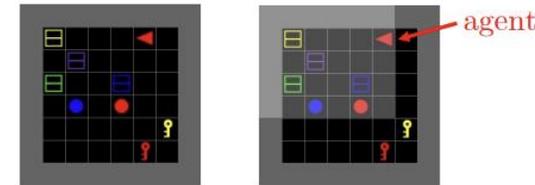
**VirtualHome**



Graph partial observation:

Goal predicates:  
 $Inside(pancake, stove): 2$

**BabyAI**



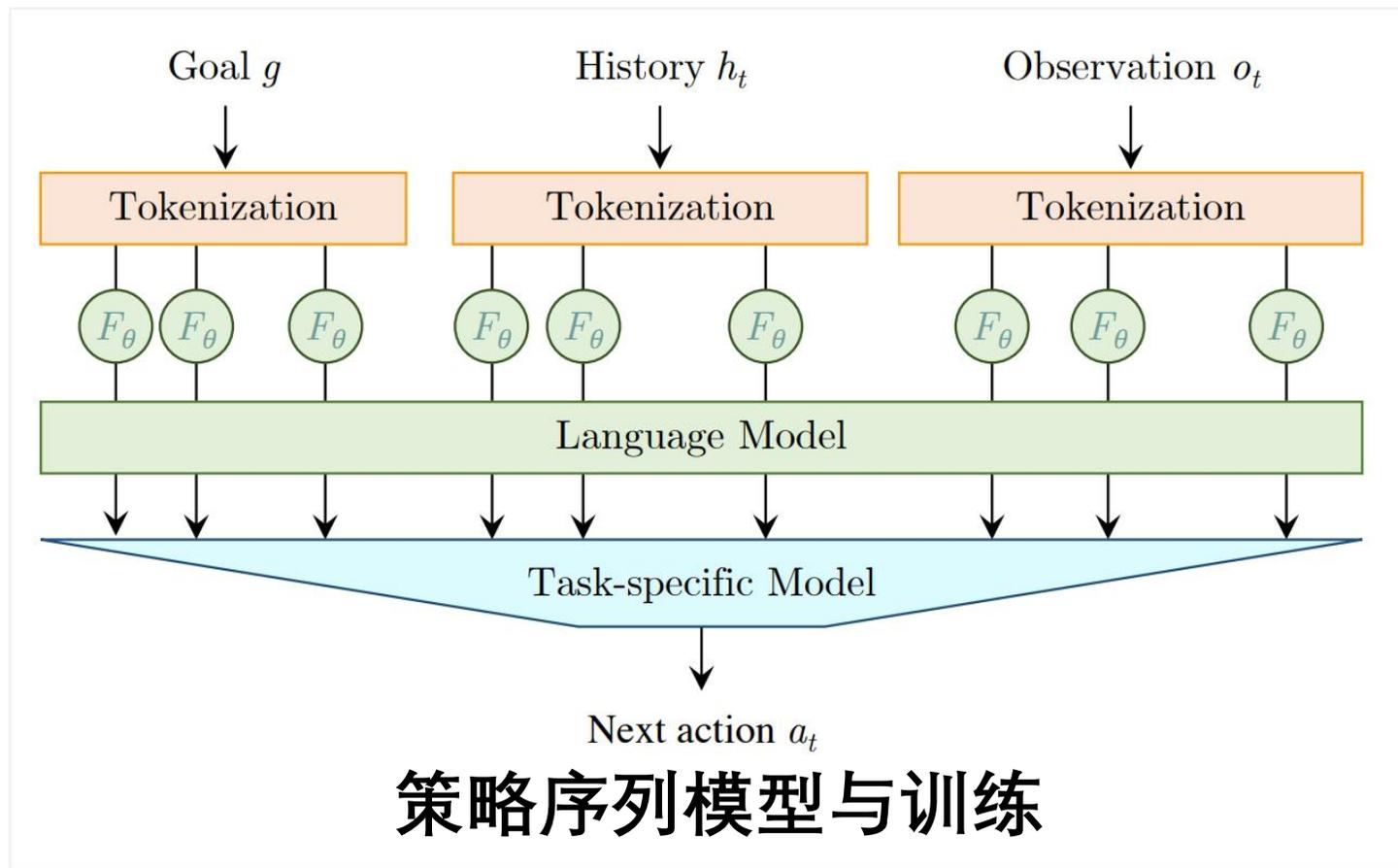
Grid partial observation:

Language goal:  
*Put the green box next to the purple box*

agent

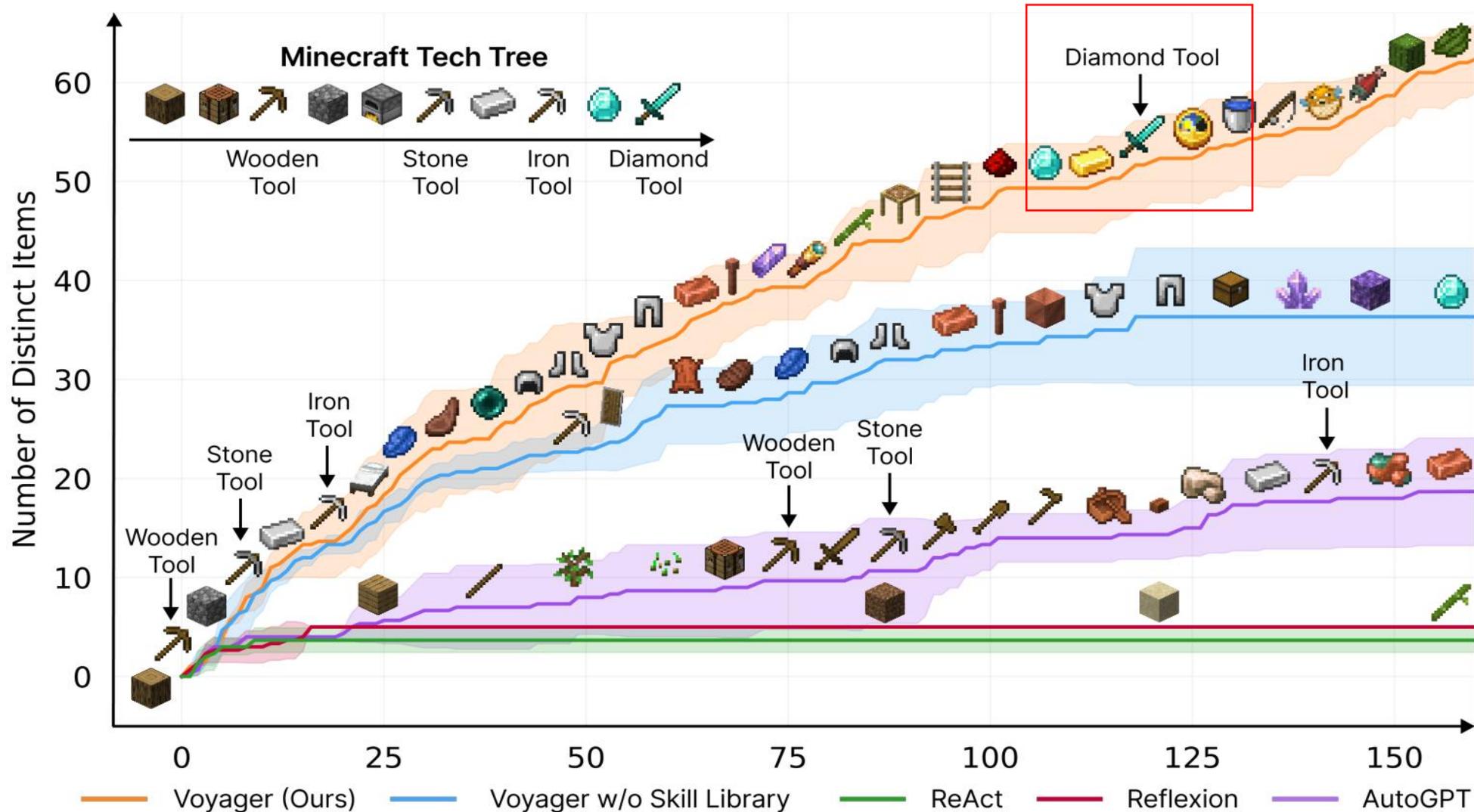
agent

## 人类演示数据



Pre-Trained Language Models for Interactive Decision-Making (LID), NeurIPS2022

# LLM 具身智能：以“我的世界”为例

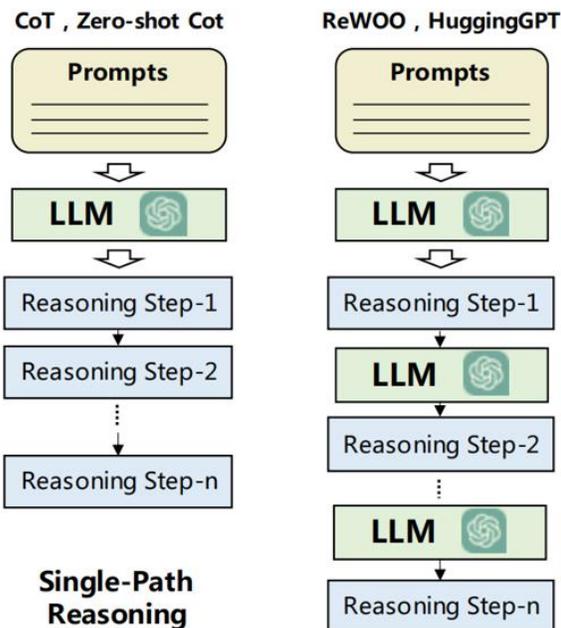


# LLM 具身智能

## 以程序生成的形式完成原本是强化学习的任务

### 规划探索空间

Automatic Curriculum



### 生成子空间程序

Iterative Prompting Mechanism

```
async function combatZombie(bot) {  
  // Equip a weapon  
  const sword = bot.inventory.findInventoryItem(  
    mcData.itemsByName["stone_sword"].id);  
  if (sword) {  
    await bot.equip(sword, "hand");  
  }  
  else {  
    await craftStoneSword(bot);  
  }  
  // Craft and equip a shield  
  await craftShield(bot);  
  ...  
}
```

New Task

Skill Retrieval

Env Feedback Execution Errors

Code as Actions

Refine Program

Add New Skill



Environment

Self-Verification

### 更新代码库

Skill Library

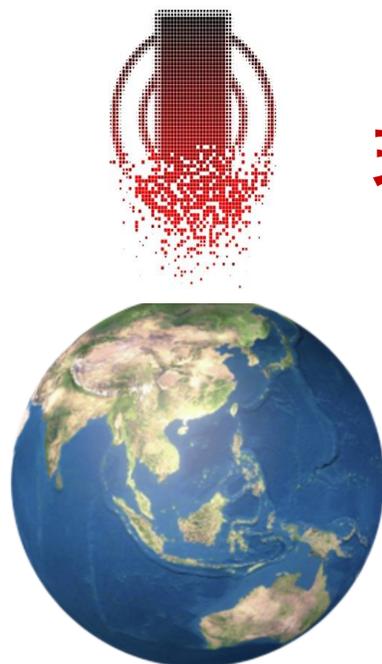
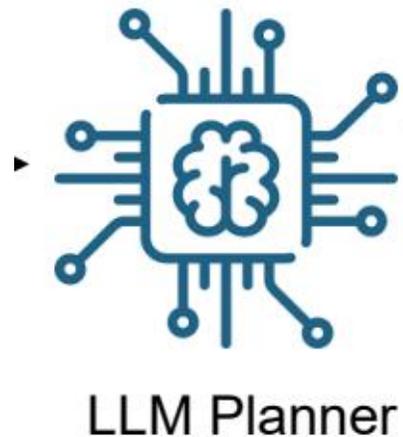
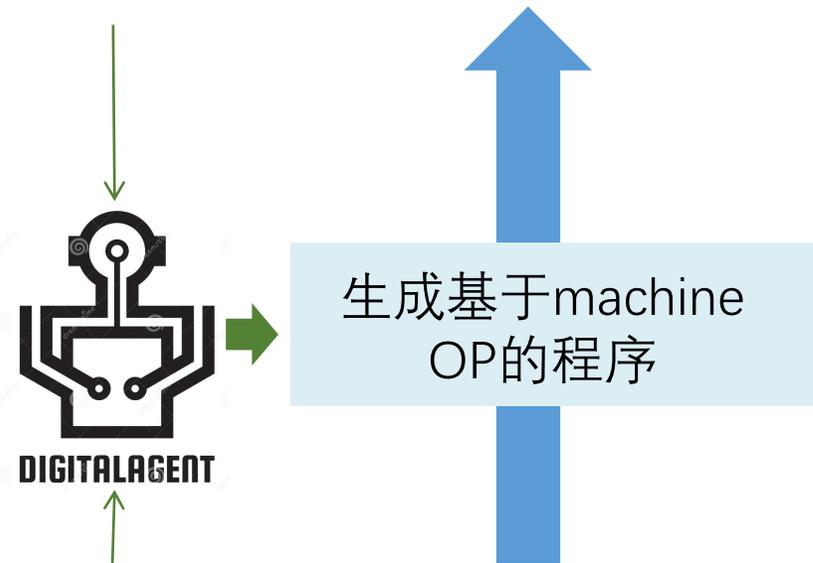
- Mine Wood Log
- Make Crafting Table
- Craft Stone Sword
- Make Furnace
- Craft Shield
- Cook Steak
- Combat Zombie

<https://github.com/MineDojo/Voyager>

# LLM Embodied Agents挑战

可有效计算的问题

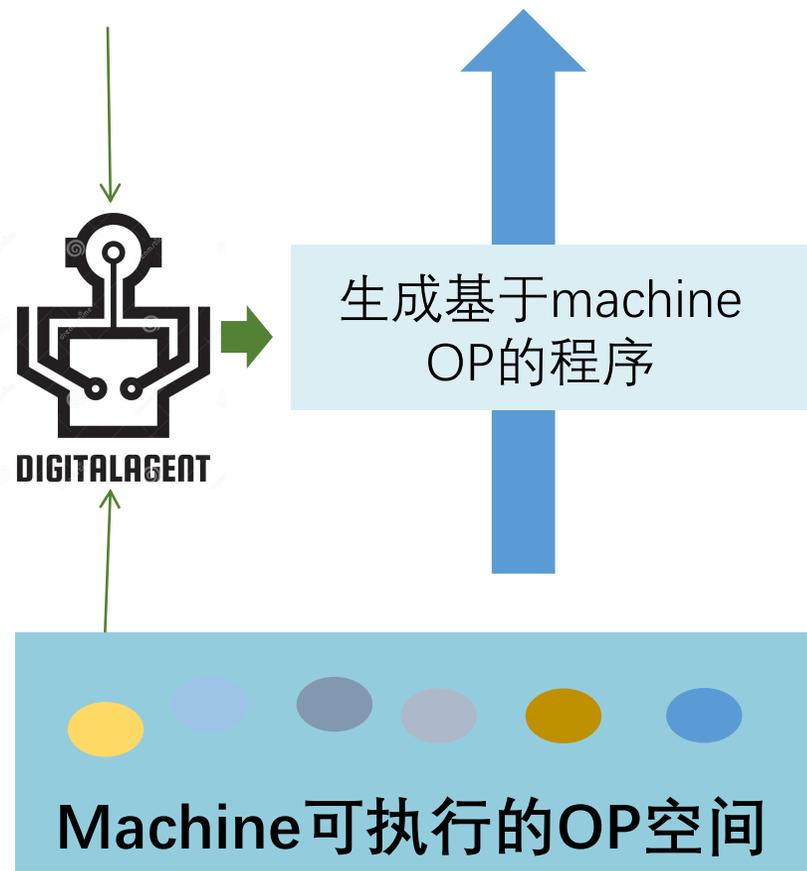
问题描述 I/O数据输入 验证条件



# 自动语言对齐的强化学习

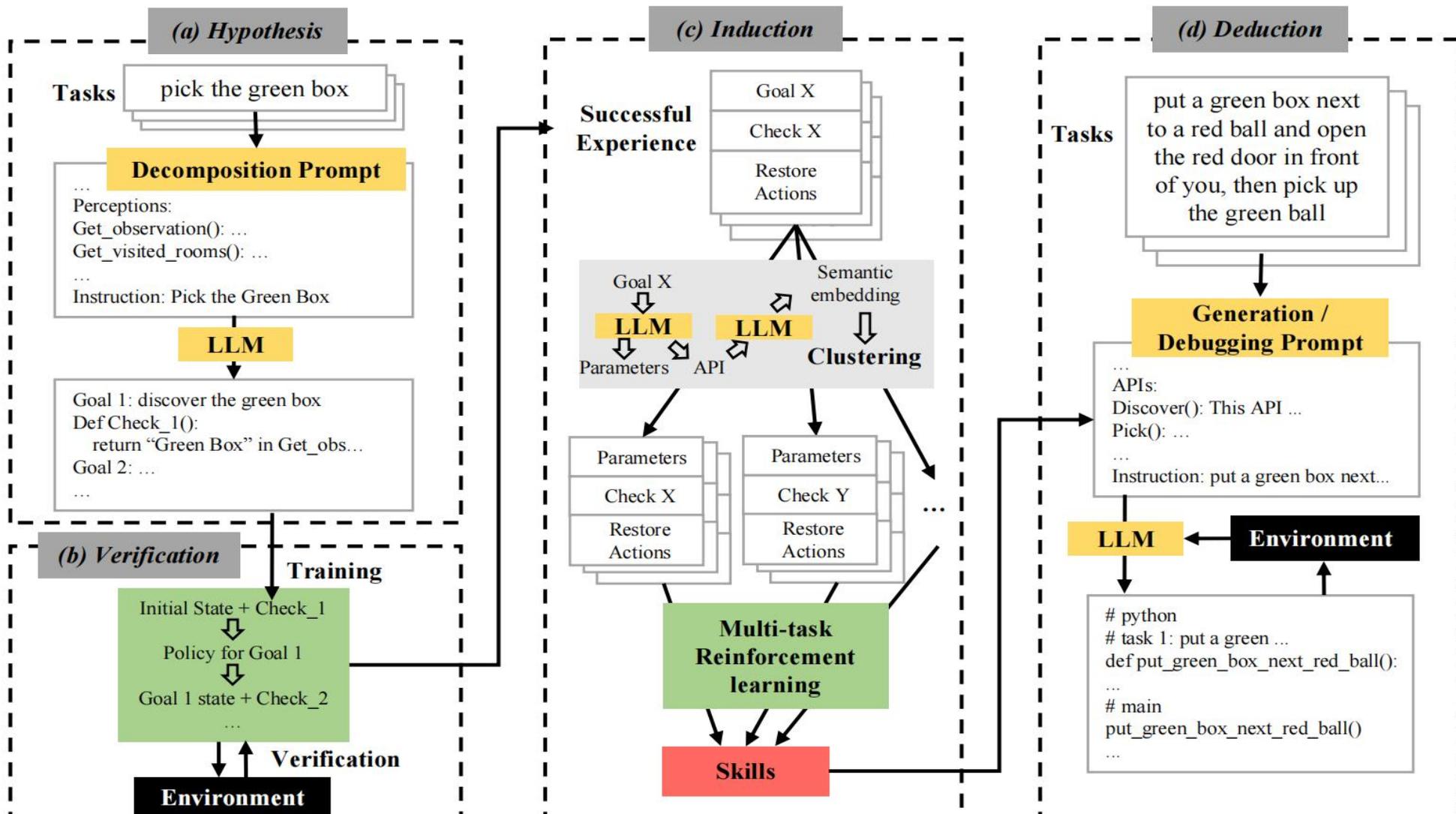
可有效计算的问题

问题描述 I/O数据输入 验证条件



Self-driven Grounding: Large Language Model Agents with Automatical Language-aligned Skill Learning, arxiv23

# 自动语言对齐的强化学习



Self-driven Grounding: Large Language Model Agents with Automatic Language-aligned Skill Learning,

arxiv23

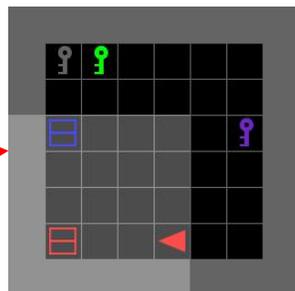
# 自动化对齐：猜想验证

## ● 验证结果示例

Discover the green key

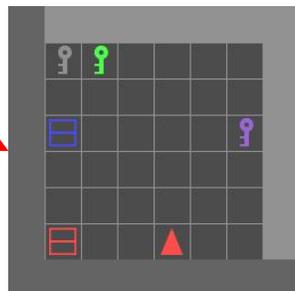
```
Task Instruction: "Go to the green key."
Goals:
(1) Check if the robot has discovered the green key.

python
def check_discovered_green_key(robot):
    obs = robot.get_observation()
    return "green key" in obs.keys()
```



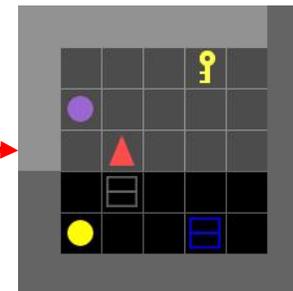
Next to the green key

```
python
def check_next_to_green_key(robot):
    obs = robot.get_observation()
    if "green key" in obs.keys():
        return sum(abs(obs["green key"])) == 1
    else:
        return False
```

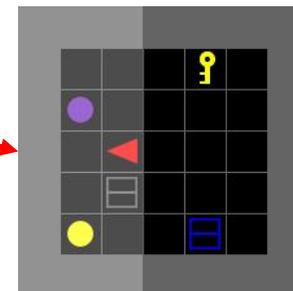


```
Task instruction: "Pick up the yellow ball"
Goals:
(1) Discover the yellow ball.

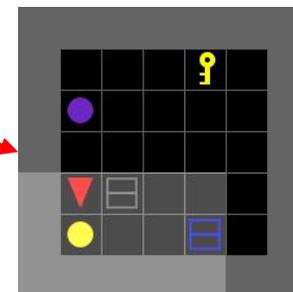
python
def check_discover_yellow_ball(robot):
    obs = robot.get_observation()
    return "yellow ball" in obs.keys()
```



```
python
def check_go_next_to_yellow_ball(robot):
    obs = robot.get_observation()
    if "yellow ball" in obs.keys():
        return sum(abs(obs["yellow ball"])) == 1
    else:
        return False
```



```
python
def check_pick_up_yellow_ball(robot):
    return robot.get_carried() == "yellow ball"
```



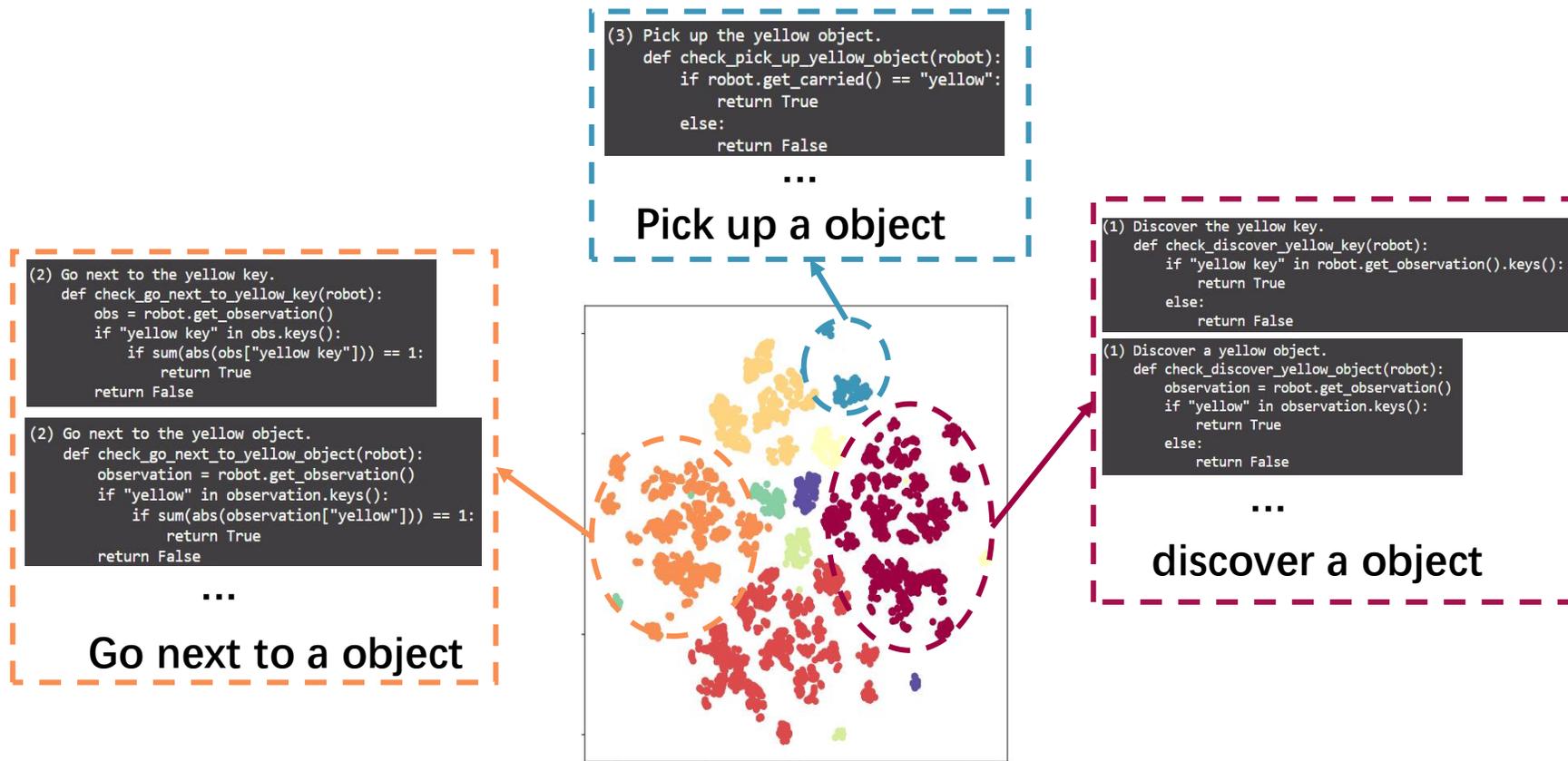
足所有子目标和任务完成，每个子目标及其检查函数作为成功“对齐”的经验被收集

Self-driven Grounding: Large Language Model Agents with Automatical Language-aligned Skill Learniv23

# 自动化对齐：归纳

- 归纳聚类结果

- 成功验证过的子目标根据语义聚类，每类训练对应的泛化技能



Self-driven Grounding: Large Language Model Agents with Automatical Language-aligned Skill Learning, arxiv23

# 自动化对齐：演绎

利用对齐技能程序生成完成更复杂的人类指令

User:  
Instruction: put the red key next to the yellow key



程序生成

```
1 # Candidate program:
2 # task 1: put the red key next to the
  yellow key
3 def put_red_key_next_to_yellow_key(robot
  ):
4     # 1. discover the red key and yellow
      key
5     robot.discover_object("red key")
6     robot.discover_object("yellow key")
7
8     # 2. go to the red key
9     robot.go_next_to("red key")
10
11    # 3. pick the red key
12    robot.pick("red key")
13
14    # 4. go to the yellow key,
15    robot.go_next_to("yellow key")
16
17    # 5. put the red key next to the
18    yellow key
19    robot.put_next_to("red key", "yellow
20    key")
21    return
22 # main
23 put_red_key_next_to_yellow_key(robot)
```

Discover red key后没有直接寻路过去，导致pick失败

反馈

# Error messages: Pick error!



debugging

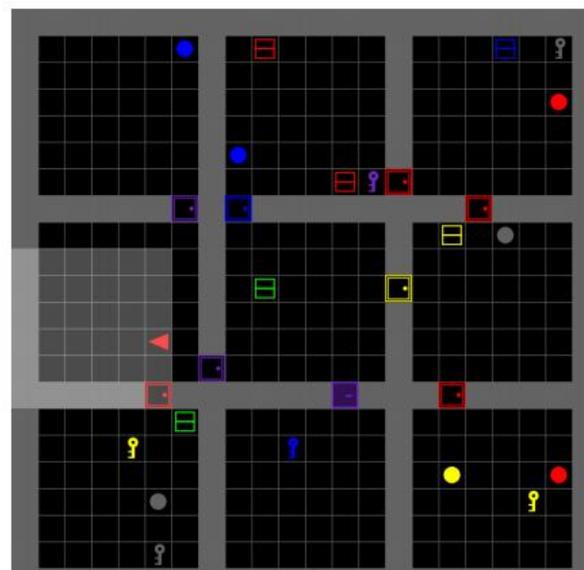
ChatGPT:

```
1 # Modified program begin:
2 # task 1: put the red key next to the
  yellow key
3 def put_red_key_next_to_yellow_key(robot
  ):
4     # 1. discover the red key
5     robot.discover_object("red key")
6
7     # 2. go to the red key
8     if "red key" in robot.
9         get_observation().keys():
10            robot.go_next_to("red key")
11    else:
12        raise ValueError("The robot didn
13        't discover the red key
14        before go to it")
15
16    # 3. pick the red key
17    if sum(abs(robot.get_observation()["
18        red key"])) == 1:
19        robot.pick("red key")
20    else:
21        raise ValueError("The robot is
22        not next to the red key")
23
```

Self-driven Grounding: Large Language Model Agents with Automatical Language-aligned Skill Learning, arxiv23

# ► 自动化对齐在虚拟环境中的实验

- BabyAI环境
  - 格子世界
  - 基于模板生成的自然语言指令集合
  - 包含一系列从简单到复杂的任务
    - 多种组合性物品，颜色/形状/种类
    - 寻路，捡/放物品，开门等
    - 组合任务



Instruction:  
open the red door on your left,  
pick up a green box and then  
put the purple key next to a red  
ball

Action Space:  
Turn left  
Turn right  
Move forward  
Pickup  
Drop  
Toggle  
Done

Level	Map	Scenario	Mission Space
GoToLocal	single room	single-task	go to the {color} {type}
PickupLoc	single room	single-task	pick up the {color} {type}
PutNextLocal	single room	single-task	put the {color} {type} next to the {color} {type}
Open	3 × 3 rooms	single-task	open a {color} door
SynthSeq	3 × 3 rooms	combined-task	Combined instructions of "GoTo", "PickUp", "PutNext", "Open" using "and", ", then" and "after you"
BossLevel	3 × 3 rooms	combined-task	Command can be any sentence drawn from the Baby Language grammar

Self-driven Grounding: Large Language Model Agents with Automatical Language-aligned Skill Learning, arxiv23

# 自动化对齐有效提升搜索效率和泛化能力

Task	Method	Type	Demos	rate
GoToLocal <small>演示模仿 演示模仿 LLM+基础action空间</small>	Original	IL	10K	99.8
	LID-Text	IL	10K	99.5
	HYVIN-action	LLM	0	55.1
	HYVIN	LLM	0	99.9
PickupLoc	Original	IL	10K	99.8
	LID-Text	IL	10K	99.8
	HYVIN-action	LLM	0	47.6
	HYVIN	LLM	0	92.9
PutNextLocal	Original	IL	10K	97.7
	LID-Text	IL	10K	99.9
	HYVIN-action	LLM	0	0
	HYVIN	LLM	0	91.9
Open	Original	IL	1M	100
	HYVIN-action	LLM	0	0
	HYVIN	LLM	0	92.4
SynthSeq	Original	IL	1M	87.7
	LISA	IL	100K	61.2
	HYVIN-action	LLM	0	0
	HYVIN	LLM	0	78.9
BossLevel	Original	IL	1M	84.3
	LISA	IL	100K	69.8
	HYVIN-action	LLM	0	0
	HYVIN	LLM	0	75.9

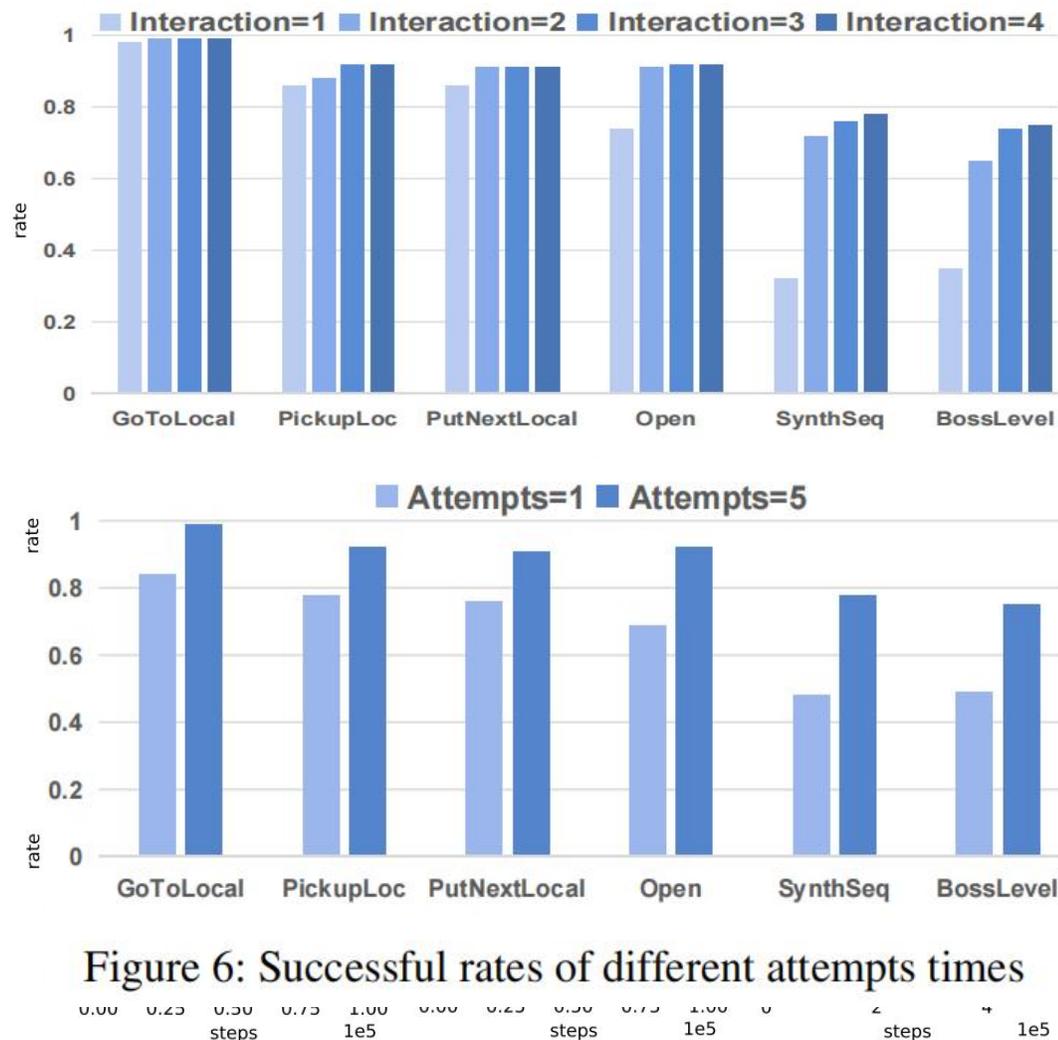


Figure 6: Successful rates of different attempts times

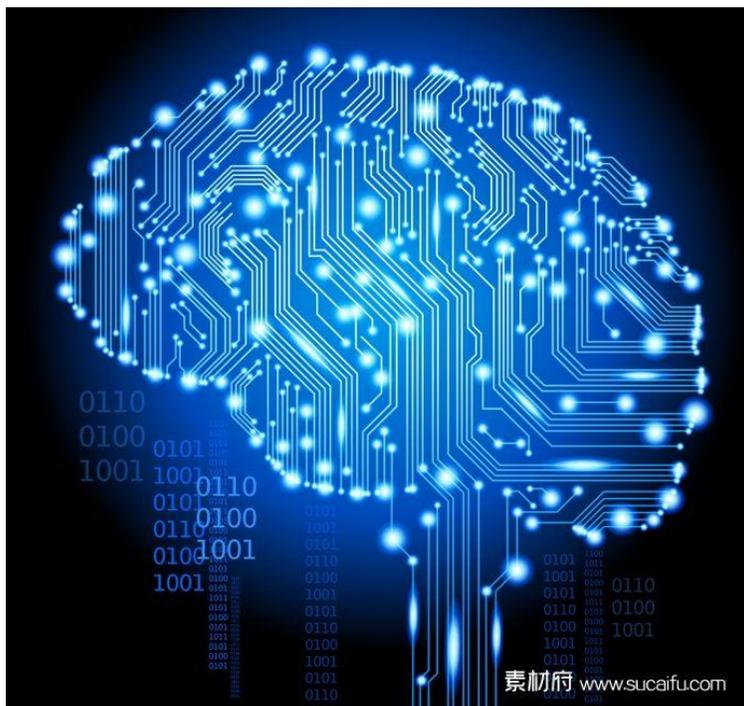
Self-driven Grounding: Large Language Model Agents with Automatical Language-aligned Skill Learning, arxiv23

## **PART 04**

# **展望：从程序生成的角度看具身智能**

# ▶ 从语义世界到物理世界

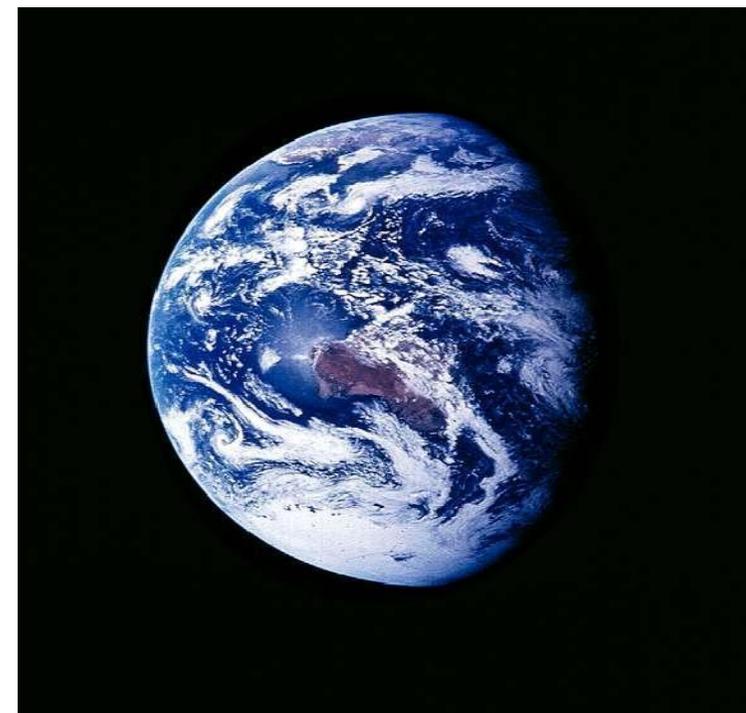
语义世界



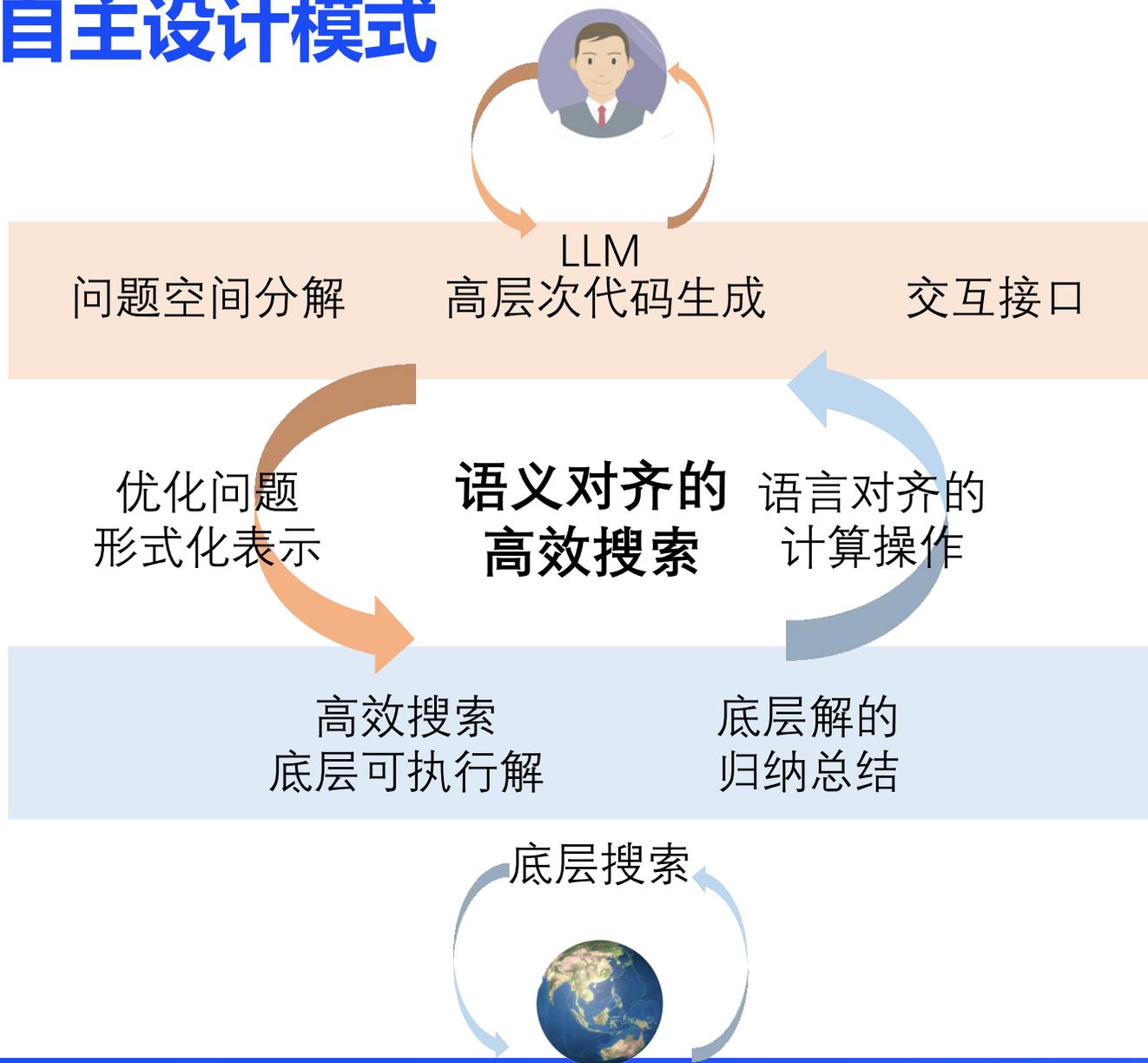
模拟环境世界



真实环境世界



# ▶ 智能自主设计模式



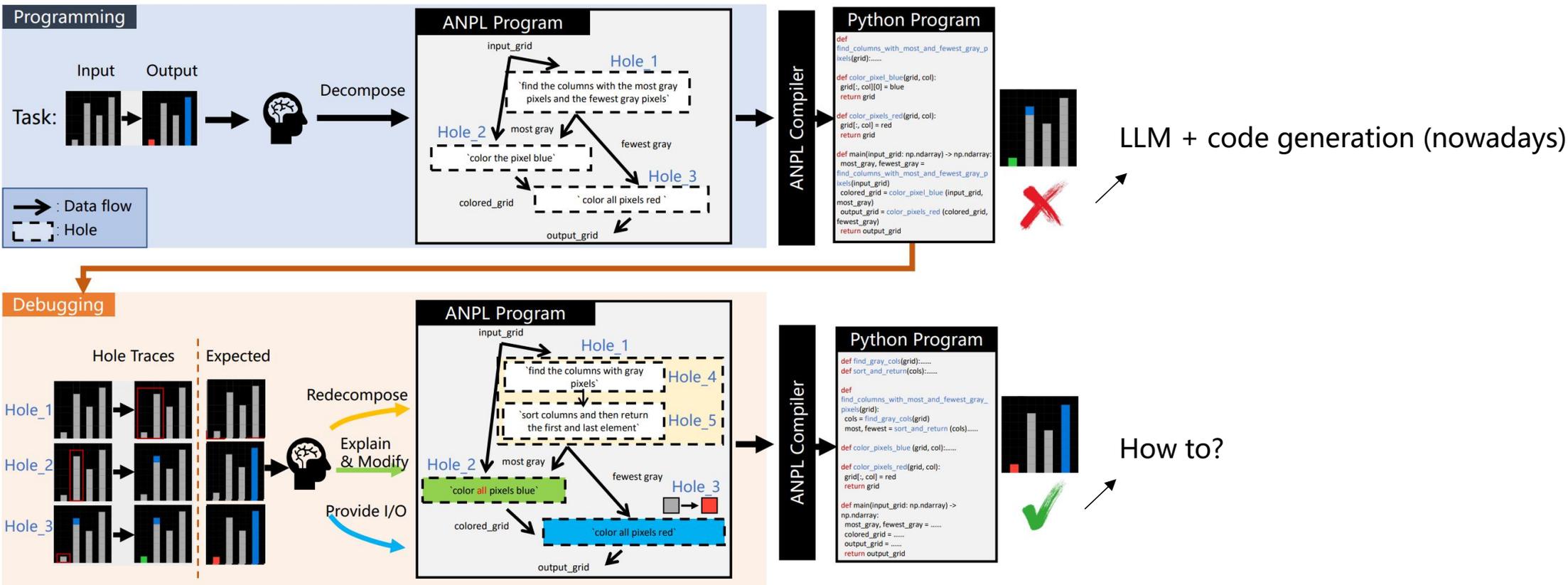
交互基础设施

代码基础设施

数据基础设施

# 新抽象交互编程基础设施

## 大模型自然语言代码生成方法不能通过稳定交互完成复杂任务



[1] Are automated debugging techniques actually helping programmers?  
 [2] Expertise in debugging computer programs: A process analysis

# ▶ 新抽象交互编程基础设施

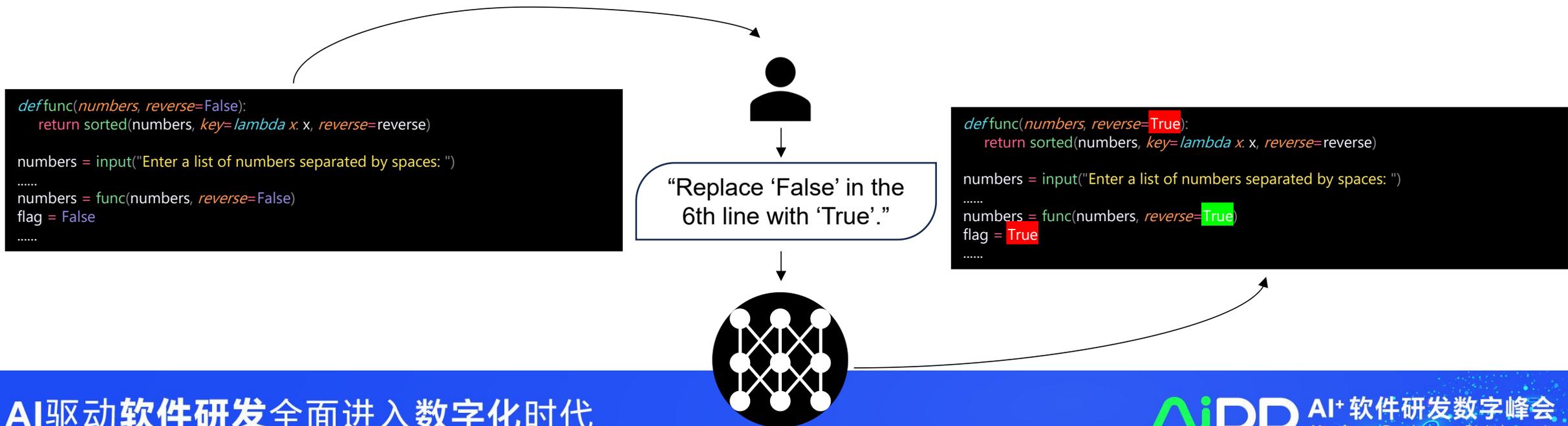
大模型的难预测性和不稳定性导致无法通过交互保证正确性收敛

普通程序：随着交互进行程序向**正确性收敛**

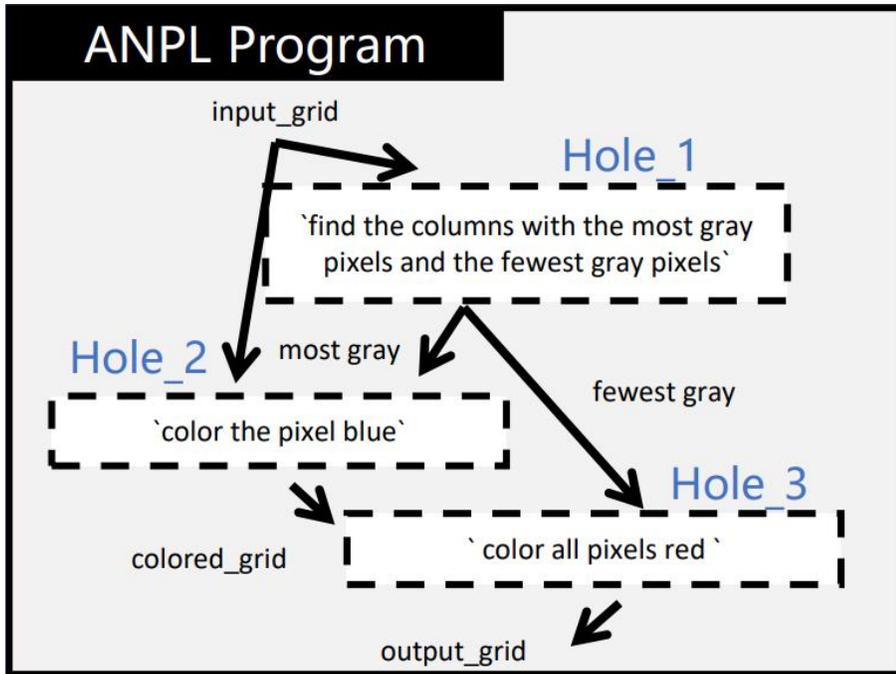
自然语言程序：大模型具有**难预测性和不稳定性**

针对编程错误进行的修正和针对复杂任务进行的分解无法像传统编程的debugging过程一样起到修正结果的作用，导致programming + interaction < programming

意味着大模型难以随着交互进行逐渐明确用户意图，是一种**低效交互**



# ▶ 基于抽象语义和确定性流程的自然语言编程框架



**ANPL: The Abstracted Natural Programming Language**  
 通过分离**控制数据流和功能**来定义交互框架，明晰用户意图

```

def main(input_grid: np.ndarray) -> np.ndarray:
    half_grid = input_grid[input_grid.shape[0] // 2:, :]
    mirror_grid = `flip the half_grid up and down`(half_grid)
    output_grid = `stack two grids with the first grid on top`(mirror_grid, half_grid)
    return output_grid
  
```

兼顾**语义明确性**和**编程复杂度**

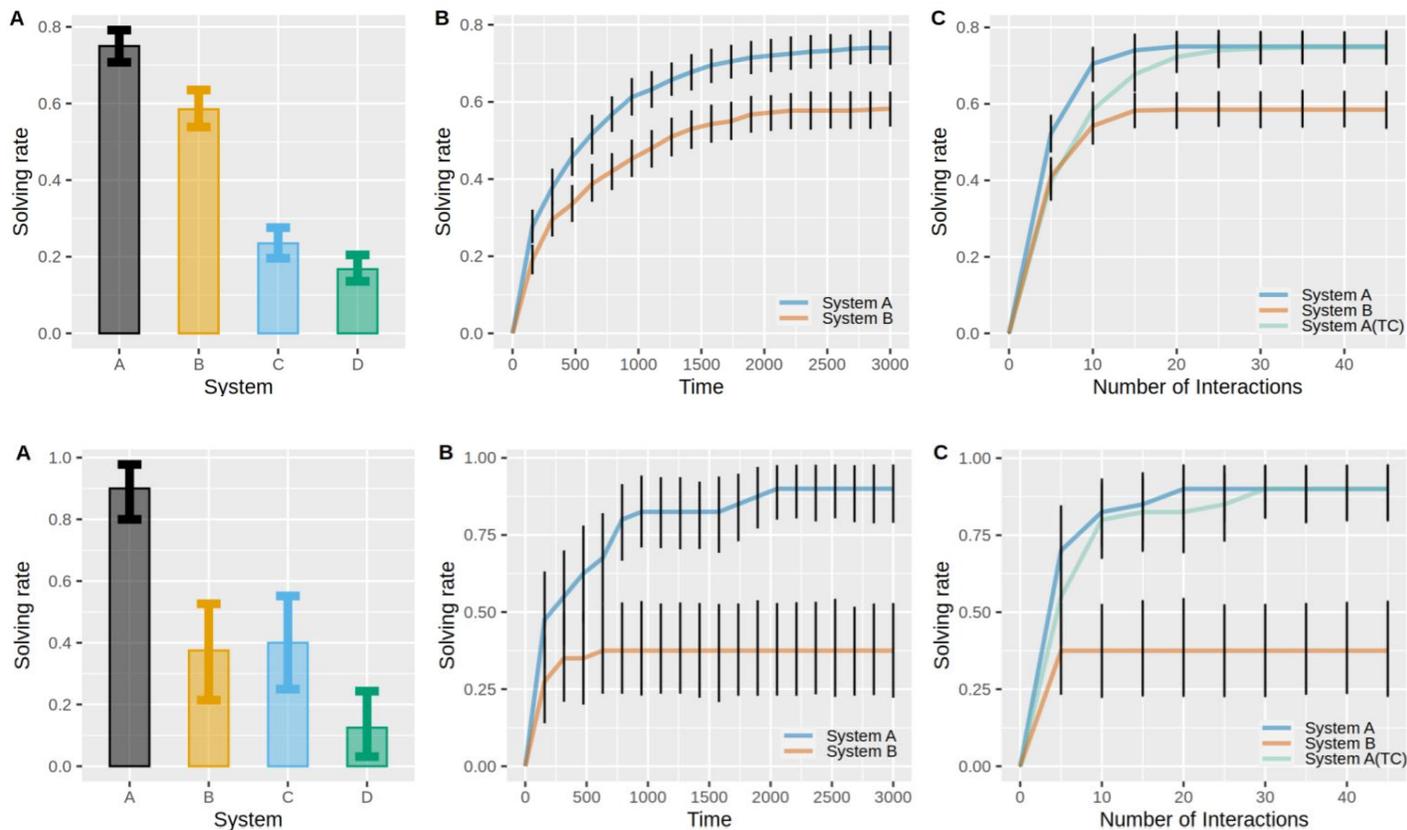
[NeurIPS-23] Di Huang, Ziyuan Nan, Xing Hu, Pengwei Jin, Shaohui Peng, Yuanbo Wen, Rui Zhang, Zidong Du, Qi Guo, Yewen Pu, Yunji Chen. "ANPL: Compiling Natural Programs with Interactive Decomposition". Neural Information Processing Systems (NeurIPS), 2023

# ▶ 440 man-hours Human studies

- 19 primary Pythoners
  - System A: ANPL
  - System B: ChatGPT
  - Random orders of A/B
- W/ Python, 400 tasks (↑ **28.25%**)
  - A: 75.0%
  - B: 58.4%
  - C: 23.5% (A without interaction)
  - D: 16.8% (B without interaction)
- W/O Python, 40 tasks (↑ **141.67%**)
  - A: 89.9%
  - B: 37.2%
  - C: 40.2% (A without interaction)
  - D: 12.5% (B without interaction)

对于熟练使用Python的用户, **ANPL能提升16.6%解题率**

对于不熟练使用Python的用户, **ANPL能提升52.8%解题率**



[NeurIPS-23] Di Huang, Ziyuan Nan, Xing Hu, Pengwei Jin, Shaohui Peng, Yuanbo Wen, Rui Zhang, Zidong Du, Qi Guo, Yewen Pu, Yunji Chen. "ANPL: Compiling Natural Programs with Interactive Decomposition". Neural Information Processing Systems (NeurIPS), 2023

# ▶ 基于Human Studies的任务数据集

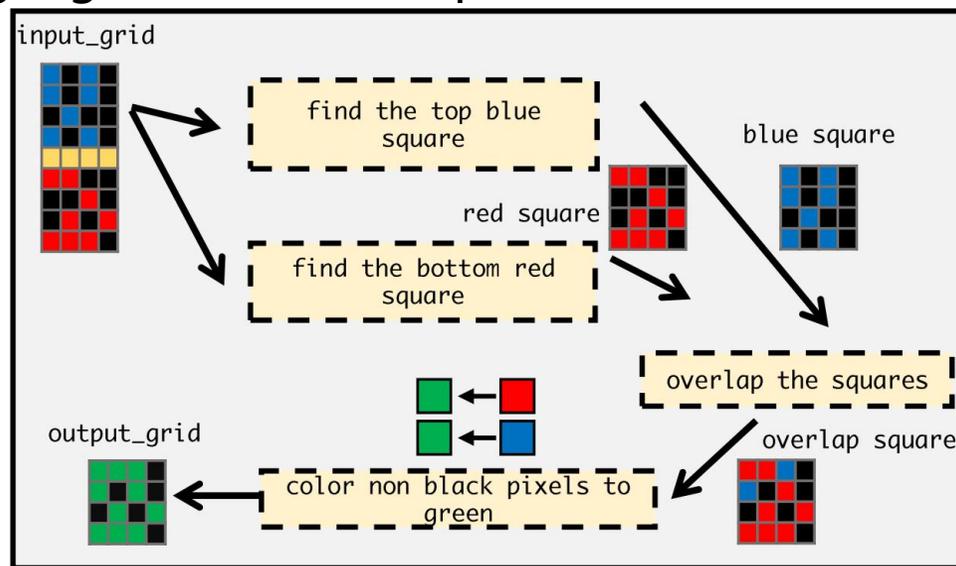
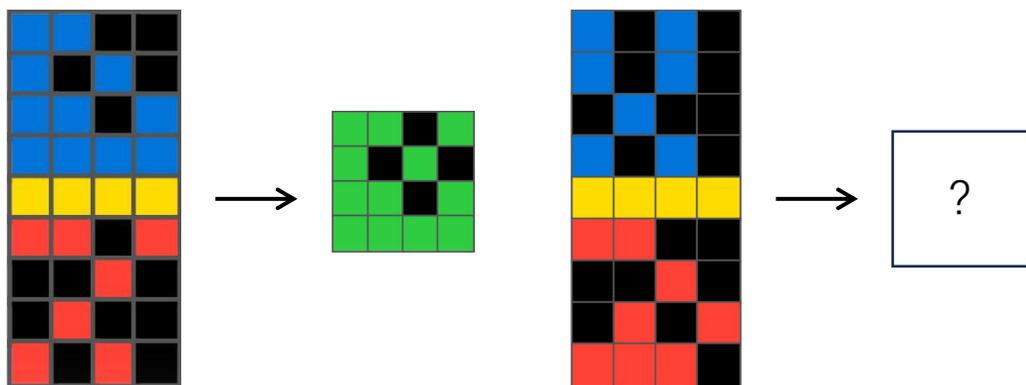
## DARC: A Recursive Decomposition Dataset of ARC Tasks

human分解策略

300可执行程序

用户的交互历史

旨在促进相关领域发展: LLMs, programming languages, human-computer interaction, and cognitive science



# ANPL可以用于编写复杂的任务

## LS-8 CPU

- 一个简易的8-bit CPU模拟器 (written in Python)

## Robot controller

- 一个可以完成line-following任务的机器小车

## Text editor

- 一个具有insert, delete, undo, copy/paste...等功能的文本编辑器

## Naïve MAGIC card game

- 一个简单的万智牌 (2个玩家的卡牌游戏, 实现对战机制以及具有不同功能的卡牌)

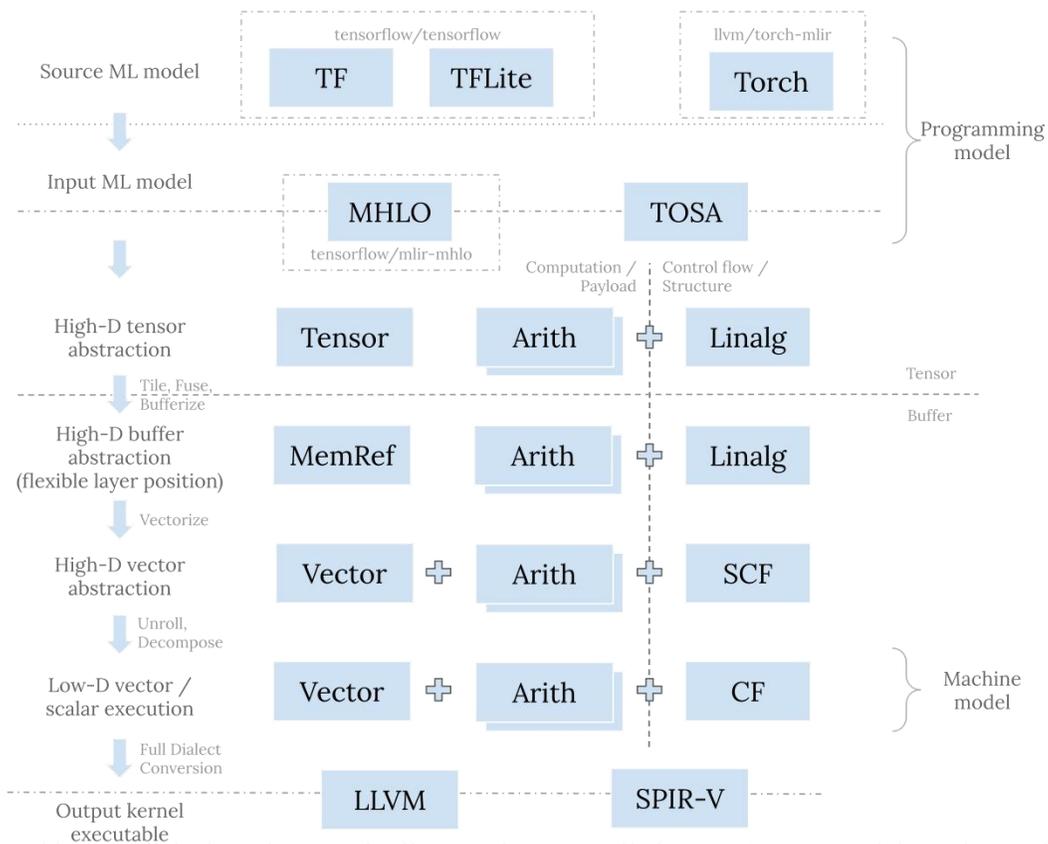
<p>ANPL</p> <pre> 1 def get_cpu(): 2     return {'memory': [0] * 256, 3             'registers': [0] * 8, 'pc': 0, 'flag': 0} 4 5 def aluop(op: int, reg_a, reg_b, cpu): 6     """This is an ALU of an 8-bit CPU. ....""" 7 8 def jmpop(op, reg_a, reg_b, cpu): 9     """This is operations which can directly change pc. Stack pointer is the 8th register. If call, push pc into the stack, and then set pc to the value of register reg_a. ....""" 10 11 def lsop(op, reg_a, reg_b, cpu): 12     """This function should support load and store operations. ....""" 13 14 def initialize_memory(cpu, filepath): 15     """This function initializes memory from file. ....""" 16 17 def main(filepath): 18     cpu = get_cpu() 19     initialize_memory(cpu, filepath) 20     running = True 21     while running: 22         exit() </pre>	<pre> 1 from pybricks.parameters import * 2 from pybricks.hubs import EV3Brick 3 from pybricks.ev3devices import * 4 from pybricks.tools import wait 5 from pybricks.robotics import 6 DriveBase 7 8 def main(): 9     BLACK = 9 10    WHITE = 85 11 12    base_speed = 200 13    correction_factor = 0.1 14 15    left_motor, right_motor, 16    color_sensor = Initialize the EV3 brick 17    and devices with A as left B as right and 18    S1 as color sensor (Ports). 19 20    while True: 21        intensity = 'Get the 22        intensity through color_sensor' 23        (color_sensor=color_sensor) 24        correction = 'Calculate the 25        correction using the correction_factor 26        and the intensity deviation from the two 27        colors threshold (intensity=intensity, 28        color_list=[BLACK, WHITE], 29        correction_factor=correction_factor) 30 31        exit() </pre>	<pre> 1 def text_editor_main(text_editor, 2 signal): 3     if signal == 'backspace': 4         """Remove the element in 'text' 5         before the 'cursor_position' (text_editor, 6         signal) 7         """ 8         ..... 9         """ 10        def main(): 11            text_editor = dict() 12 13            """Init text_editor with 'text', 14            'cursor_position', 'clipboard', 15            'operation_stack', 'selection', 16            'text_stack' (text_editor) 17 18            signal = input() 19 20            while signal != 'quit': 21                record the 'text' into 22                'text_stack' of text_editor (text_editor, 23                signal) 24                ..... 25                signal = input() 26            return </pre>	<pre> 1 def tutor_effect(card: Card, player: 2 Player) -&gt; None: 3     """A Tutor card will add a copy of 4     the first card in your hand to your hand. 5     ....""" 6     ..... 7     """ 8     def ta_effect(card, opponent_card, 9     player): 10        """A TA card discards the card with 11        the highest power in your hand, and add 12        the discarded card's attack and defense 13        to its own respective stats. ....""" 14        ..... 15        """ 16        def fight(card1, card2, player1, 17        player2): 18            """The cards' power stats are then 19            calculated and compared. ....""" 20            ..... 21            """ 22        def main(deck1, deck2): 23            player1 = create_player(deck1) 24            player2 = create_player(deck2) 25            while not check(player1, player2): 26                draw(player1) 27                ..... 28                draw(player2) 29            card_index1 = 'display the 30            properties of each card in hand and ask 31            the player for the index of the card to 32            choose.' (player1) 33            ..... </pre>
<p>Python</p> <pre> 1 def get_cpu(): 2     return {'memory': [0] * 256, 3             'registers': [0] * 8, 'pc': 0, 'flag': 0} 4 5 def aluop(op: int, reg_a: int, reg_b: 6 int, cpu: dict): 7     """ 8     if op == INC: 9         cpu['registers'][reg_a] += 1 10        cpu['pc'] += 2 11 12        """ 13 14 def jmpop(instr: str, reg_a: int, 15 reg_b: int, cpu: dict): 16     """ 17     if instr == CALL: 18         cpu['registers'][7] -= 1 19         cpu['memory'][cpu['registers'] 20         [7]] = cpu['pc'] 21         cpu['pc'] = 22         cpu['registers'][reg_a] 23         ..... 24 25 def lsop(instr, reg_a, reg_b, cpu): 26     """ 27     if instr == LD: 28         address = 29         cpu['registers'][reg_b] 30         cpu['registers'][reg_a] = 31         cpu['memory'][address] 32         ..... 33 34 def initialize_memory(cpu: Dict[str, 35 Any], filepath: str) -&gt; None: 36     """ 37     with open(filepath, 'r') as f: 38         ..... 39 40 def main(filepath): 41     cpu = get_cpu() 42     initialize_memory(cpu, filepath) 43     running = True 44     while running: 45         ..... 46     exit() </pre>	<pre> 1 from typing import * 2 from pybricks.parameters import * 3 from pybricks.hubs import EV3Brick 4 from pybricks.ev3devices import * 5 from pybricks.tools import wait 6 from pybricks.robotics import 7 DriveBase 8 9 def calculate_correction(intensity: 10 int, color_list: List[int]) -&gt; float: 11     """ 12     correction_factor: float -&gt; float: 13     threshold = sum(color_list) / 14     len(color_list) 15     """ 16 17 def initialize_devices(port: Port) -&gt; 18 Tuple(Motor, Motor, ColorSensor): 19     """ 20     ev3 = EV3Brick() 21     left_motor = Motor(port.A) 22     right_motor = Motor(port.B) 23     color_sensor = 24     ColorSensor(port.S1) 25     return (left_motor, right_motor, 26     color_sensor) 27 28 def main(): 29     BLACK = 9 30     WHITE = 85 31 32     base_speed = 200 33     correction_factor = 0.1 34 35     left_motor, right_motor, 36     color_sensor = initialize_devices(Port) 37 38     while True: 39         intensity = 40         get_intensity(color_sensor=color_sensor) 41         correction = 42         calculate_correction(intensity=intensity, 43         color_list=[BLACK, WHITE], 44         correction_factor=correction_factor) 45 46         ..... </pre>	<pre> 1 from typing import * 2 3 def remove_before_cursor(text: str, 4 cursor_position: int) -&gt; str: 5     """ 6     return text[:cursor_position - 1] 7     + text[cursor_position:] 8     """ 9 10 def text_editor_main(text_editor, 11 signal): 12     """ 13     if signal == 'backspace': 14         remove_before_cursor(text_editor, 15         signal) 16 17         """ 18         def record_text(text_editor: Dict[str, 19         Any], text: str): 20             """ 21             text_editor['text_stack'].append( 22             text) 23             text_editor['redo_stack'] = [] 24             """ 25 26         """ 27         def main(): 28             text_editor = dict() 29             initialize_text_editor(text_editor: 30             Dict[str, Any]) -&gt; None: 31                 """ 32                 text_editor['text'] = '' 33                 text_editor['cursor_position'] = 34                 0 35                 """ 36 37         """ 38         def main(): 39             text_editor = dict() 40             initialize_text_editor(text_editor) 41 42             signal = input() 43 44             while signal != 'quit': 45                 record_text(text_editor, 46                 signal) 47                 ..... 48                 signal = input() 49             return </pre>	<pre> 1 def ta_effect(card: Card, 2 opponent_card: Card, player: Player) -&gt; 3 None: 4     """ 5     max_power = float('-inf') 6     """ 7 8     def tutor_effect(card: Card, player: 9     Player) -&gt; None: 10        """ 11        card.attack = -float('-inf') 12        """ 13 14 def fight(card1: Card, card2: Card, 15 player1: Player, player2: Player) -&gt; None: 16     """ 17     power1 = card1.attack - 18     card2.defense 19     """ 20 21 def prompt_card_choice(player: Player) 22 -&gt; int: 23     """ 24     print(f'{player.name}, choose a 25     card from your hand:') 26     for (i, card) in 27     enumerate(player.hand): 28         print(f'{i}: {card.name} 29         (Attack: {card.attack}, Defense: 30         {card.defense}, Type: {card.type})') 31         while True: 32             choice = input('Enter the 33             index of the card you want to choose: ') 34             if not choice.isdigit(): 35                 print('Invalid input. 36                 Please enter a number.') 37                 continue 38             choice = int(choice) 39             if choice &lt; 0 or choice &gt;= 40             len(player.hand): 41                 print('Invalid input. 42                 Please enter a valid index.') 43                 continue 44             return choice </pre>

Figure 2: The ANPL and Python code of four projects: (a) LS-8 CPU (b) a robot controller (c) a text editor (d) a naive MAGIC. We present part of the code and mark the ANPL code and its corresponding Python code with the same background color.

[NeurIPS-23] Di Huang, Ziyuan Nan, Xing Hu, Pengwei Jin, Shaohui Peng, Yuanbo Wen, Rui Zhang, Zidong Du, Qi Guo, Yewen Pu, Yunji Chen. "ANPL: Compiling Natural Programs with Interactive Decomposition". Neural Information Processing Systems (NeurIPS), 2023

# ▶ 对齐基础设施 系统领域带来的启发

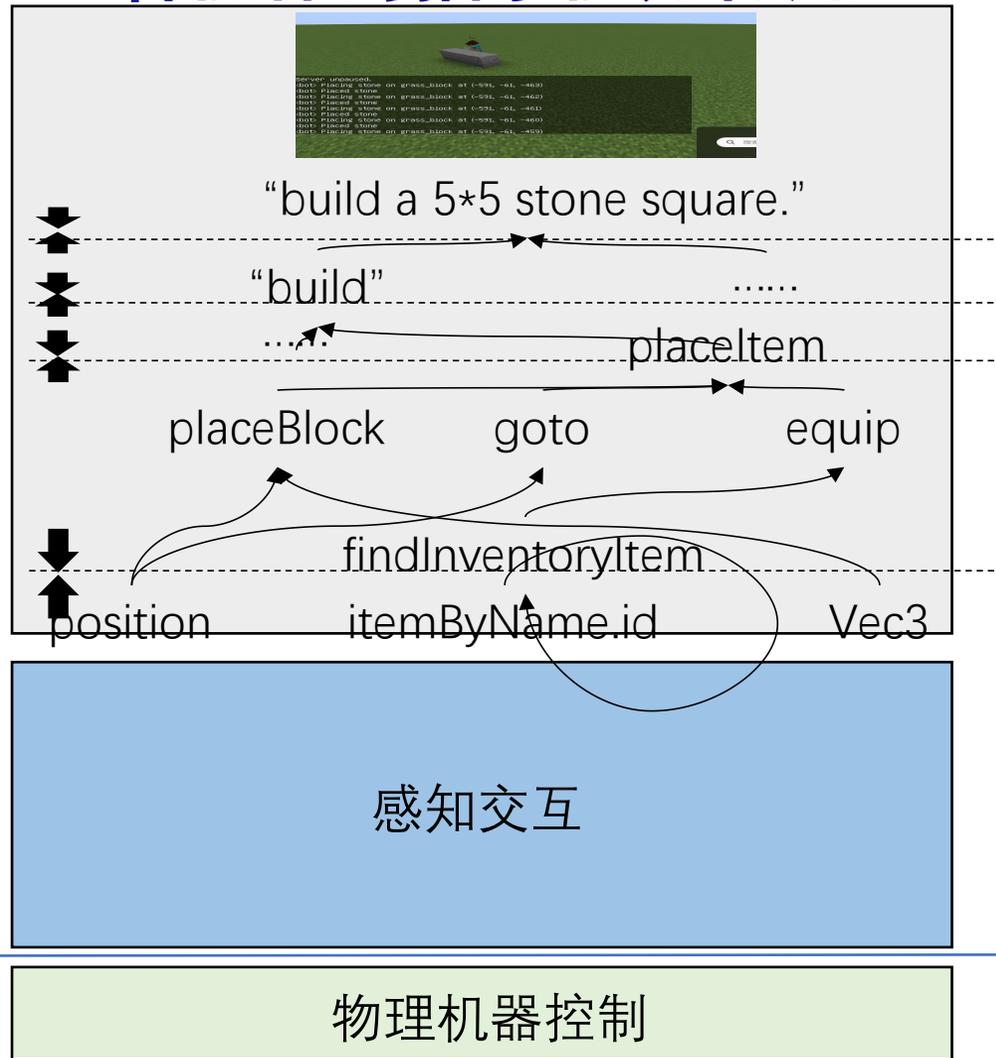
MLIR (Multi-Level Intermediate Representation)  
可复用可扩展的**多层次**编译架构、**减少构建特定领域编译器的开销**



<https://www.lei.chat/posts/mlir-codegen-dialects-for-machine-learning-compilers/>

# 智能体的抽象能力构建

模拟世界



物理世界

# 数据基础设施

## 任务数据

1M Episodes from 311 Scenes  
34 Research Labs across 21 Institutions

22 Embodiments

527 Skills

60 Datasets

1,798 Attributes • 5,228 Objects • 23,486 Spatial Relations

Skills shown: pour, stack, route

## 环境数据

Cable Routing

RT-1

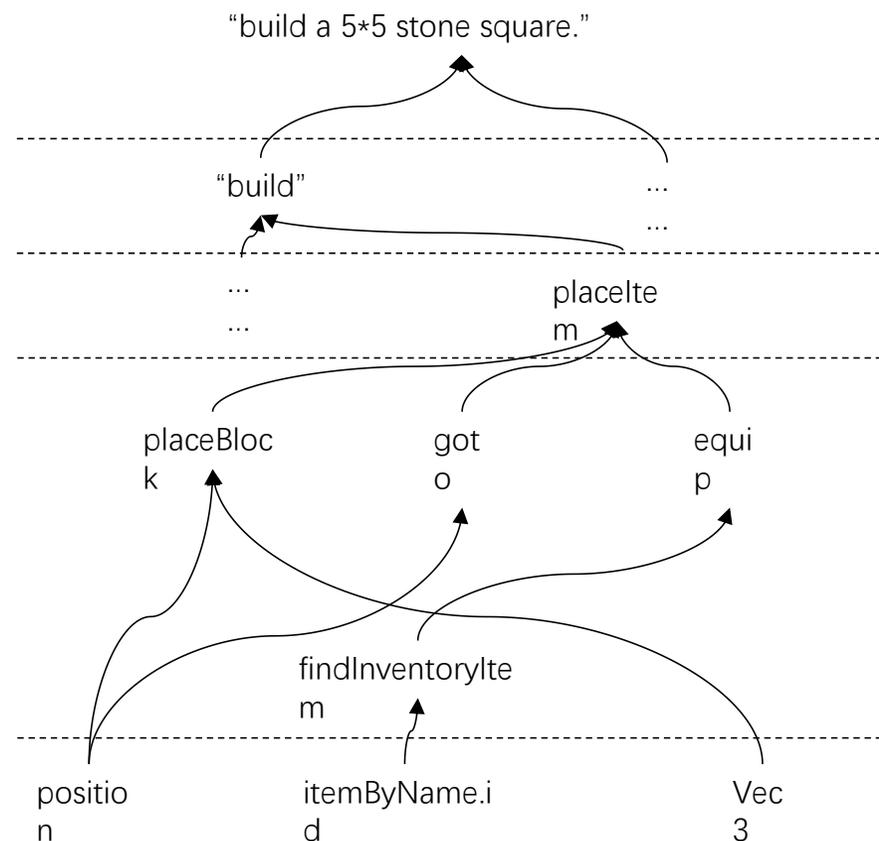
pick green chip bag from counter

set the bowl to the right side of the table

Bridge

Door Opening

## 程序数据



Open X-Embodiment: Robotic Learning Datasets and RT-X Models

# THANKS

