AiDD AI+ 软件研发数字峰会
AI+ software Development Digital summit

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 + 技术团队的共同选择

KEYLINK ing

K+峰会

K+峰会 深圳站

K+ 全球软件研发行业创新峰会

会议时间：2024.05.24-25

K+峰会 上海站

K+ 全球软件研发行业创新峰会

会议时间：2024.09.20-21

AiDD 峰会

AiDD峰会 深圳站

AI+ 软件研发数字峰会

会议时间：2023.11.24-25

AiDD峰会 北京站

AI+ 软件研发数字峰会

会议时间：2024.07.19-20

AiDD峰会 深圳站

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

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：Lamda演算

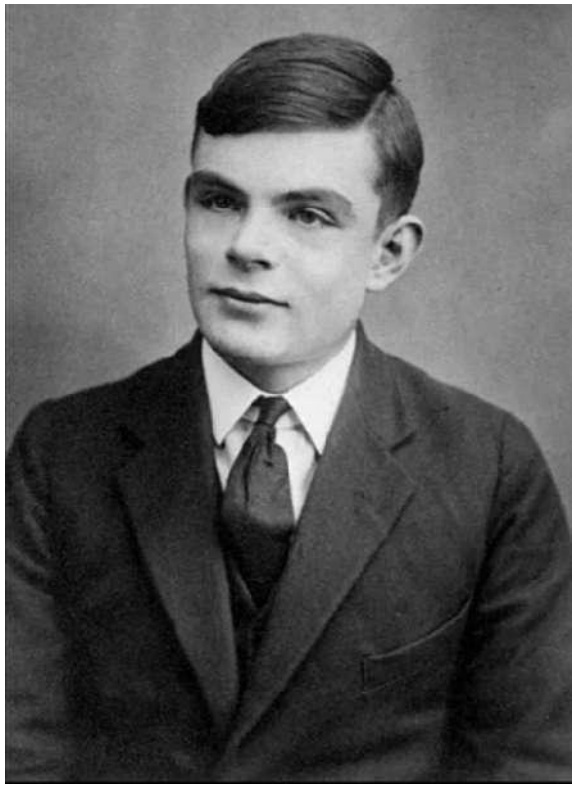
可有效计算的问题

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 = 图灵机计算过程

侧重语言和逻辑　　侧重物理和机器

Allen Turing：图灵机



解决可计算问题的过程

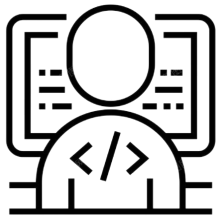
可有效计算的问题

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$F: X \to Y$,
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

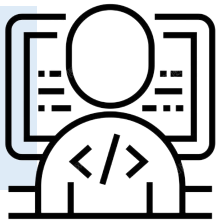
可执行的操作集合Y

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

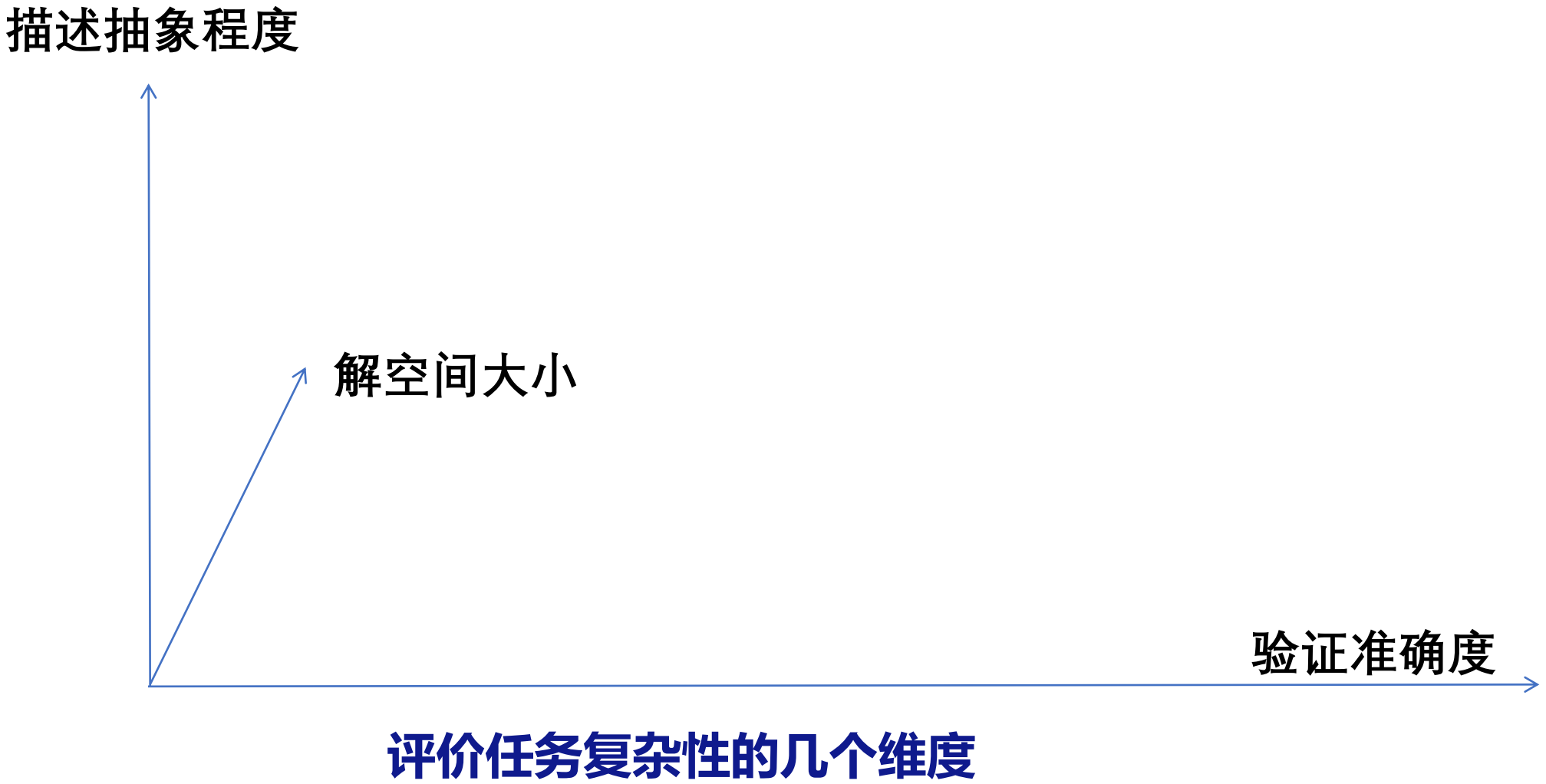
问题描述　I/O数据输入　验证条件

DIGITALAGENT

生成基于machine OP的程序

Machine可执行的OP空间

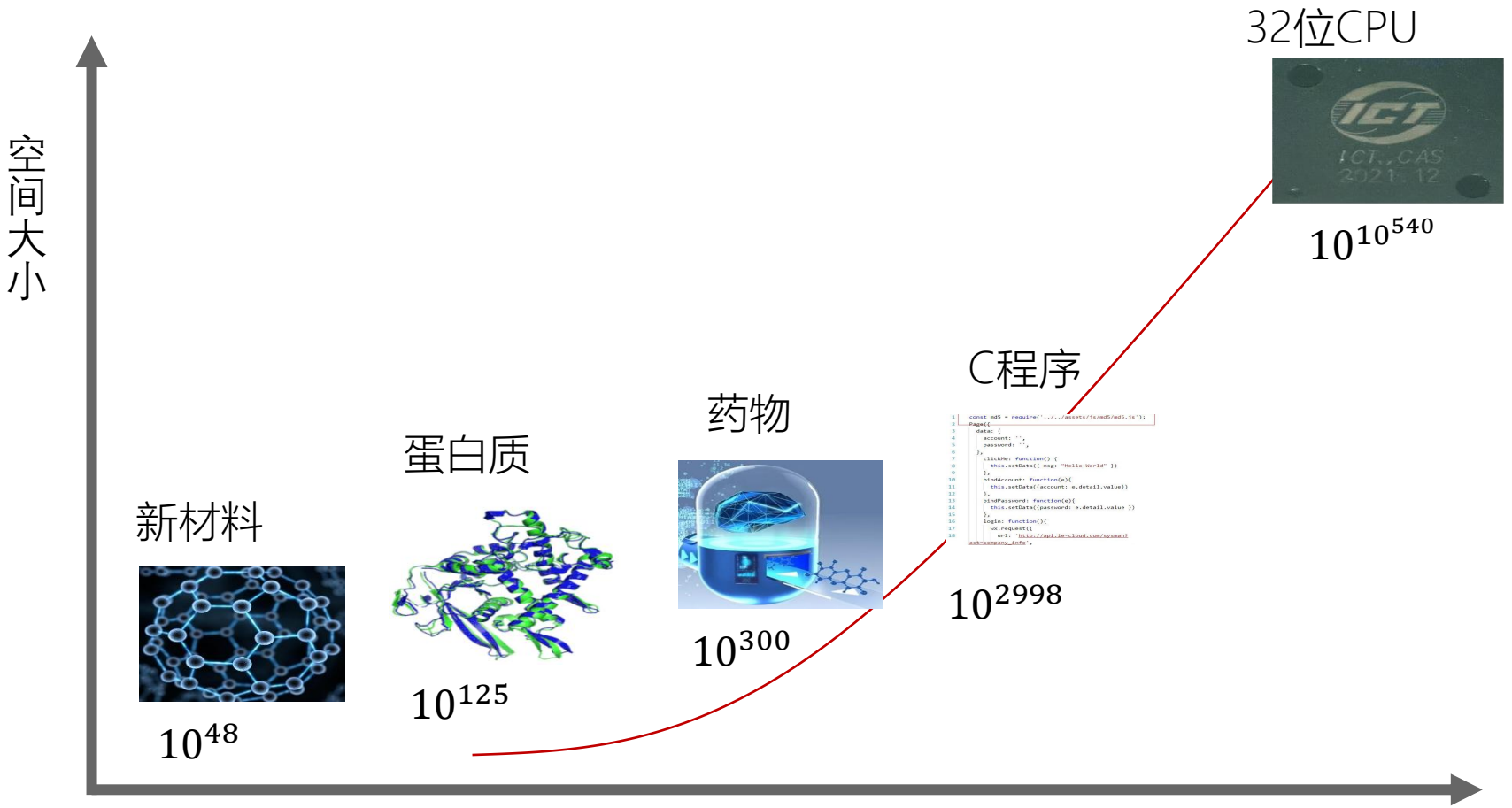
来自现实场景任务的挑战



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



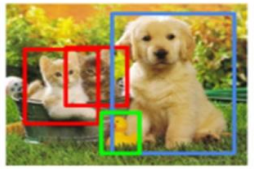
智能自主设计的挑战：来自现实场景中的任务

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**99.99999999999%**的准确率

VS

图像分类
~90%



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

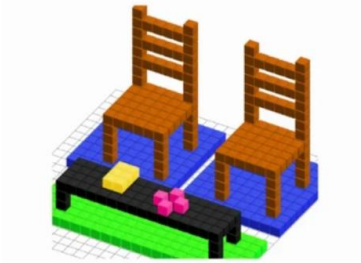
开放多样的抽象输入

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

从用户的输入输出样例中生成程序

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

add two chairs 5 spaces apart

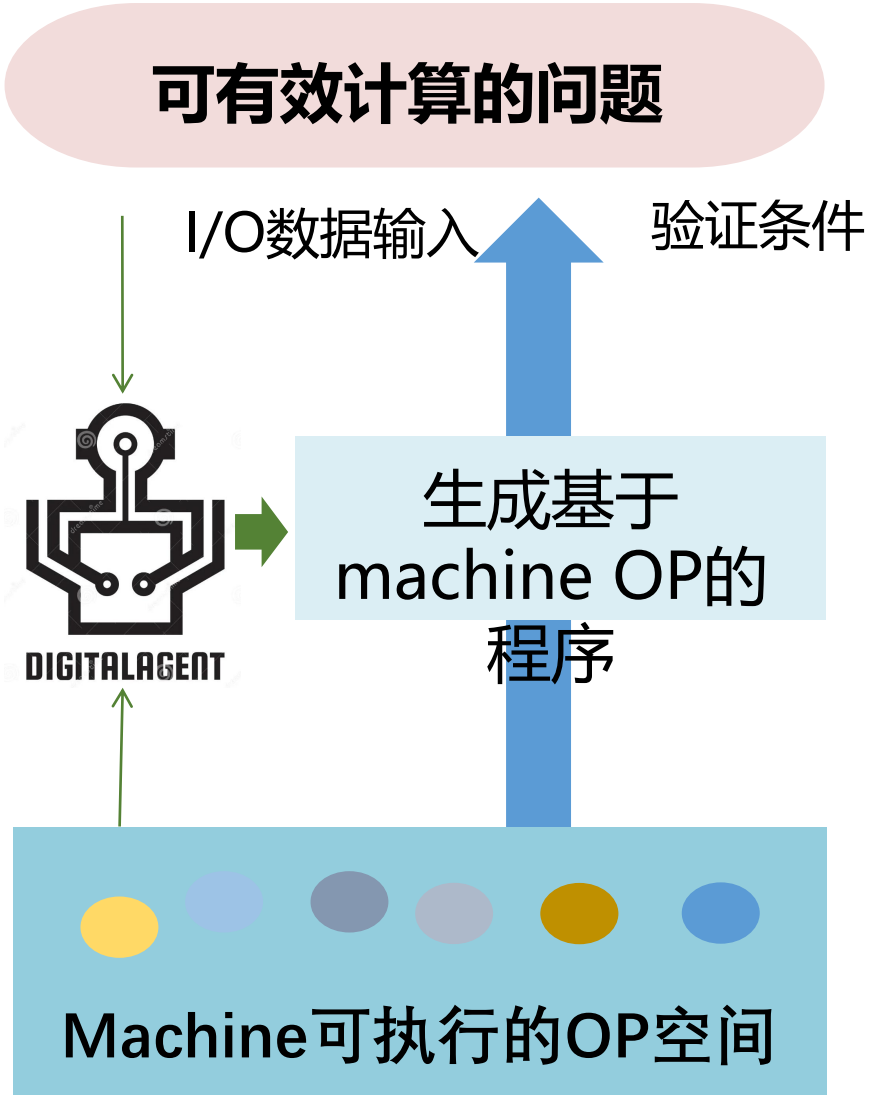


从用户的自然语言描述中生成程序

炒一盘番茄鸡蛋

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

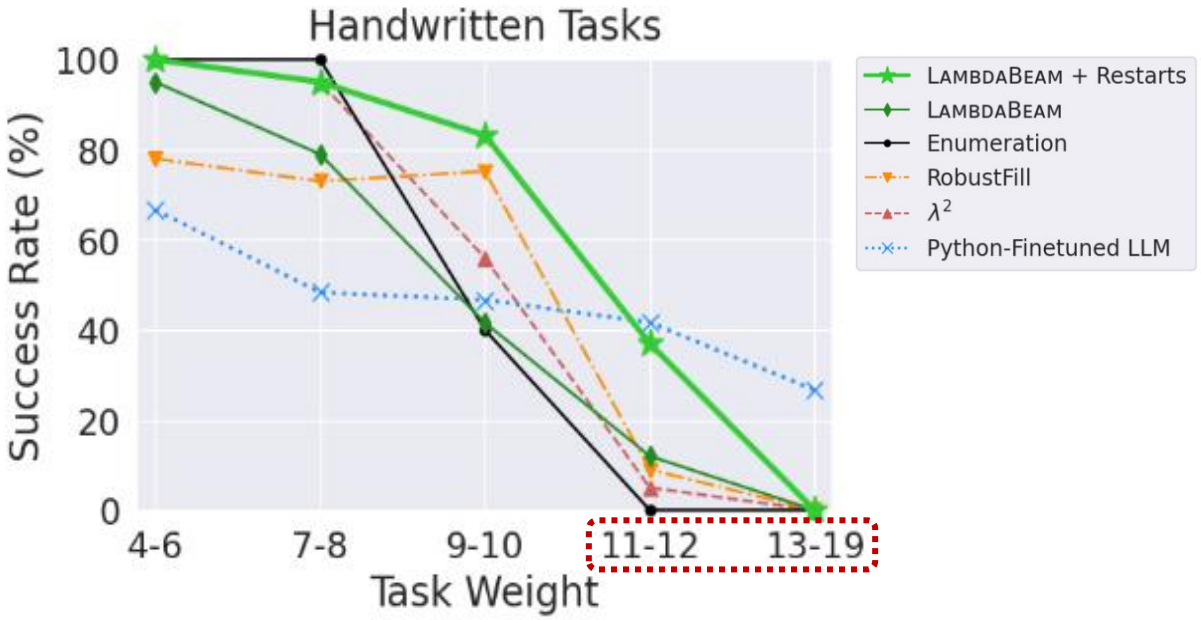
DSL设计与程序的表达能力和
搜索空间大小紧密相关

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
s ← [int]	*Input*:
b ← REVERSE s	[1 2 4 5 7]
c ← ZIPWITH (−) b s	*Output*:
d ← FILTER (>0) c	9
e ← SUM d	

List processing样例



生成正确率与语法树大小的关系[Shi'23]

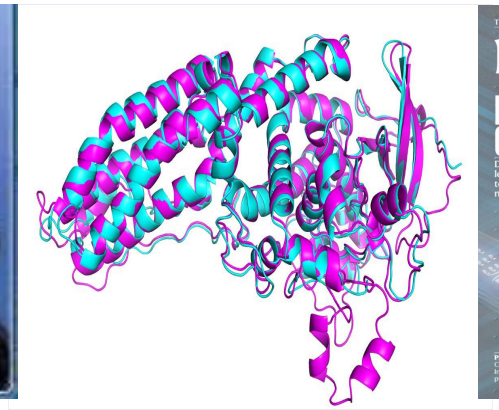
PART 02
智能自主设计的方法和挑战

智能新时代的两大里程碑

深度强化学习

DeepMind 2014

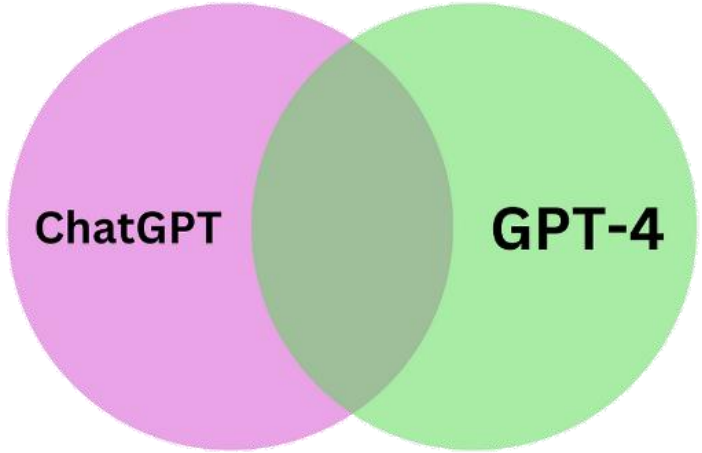
AlphaGo AlphaFold AlphaTensor



专精

大模型

OpenAI 2022



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

生成基于machine OP的程序

Machine可执行的OP空间

自顶向下

问题空间表示转换
解决描述复杂度问题

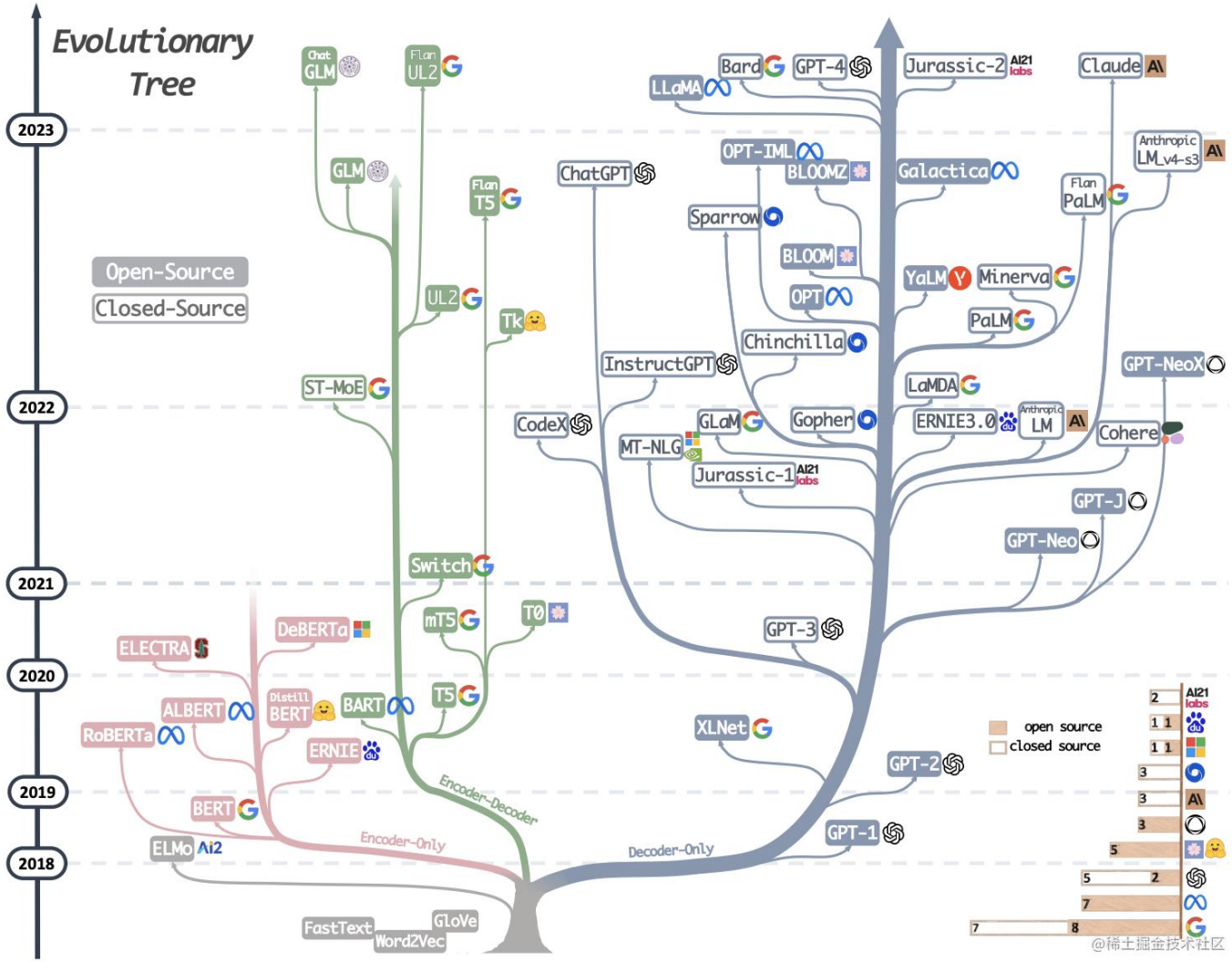
问题空间语义分解
减小搜索空间

自底向上

搓工具
构造功能更复杂、更多样的可计算载体

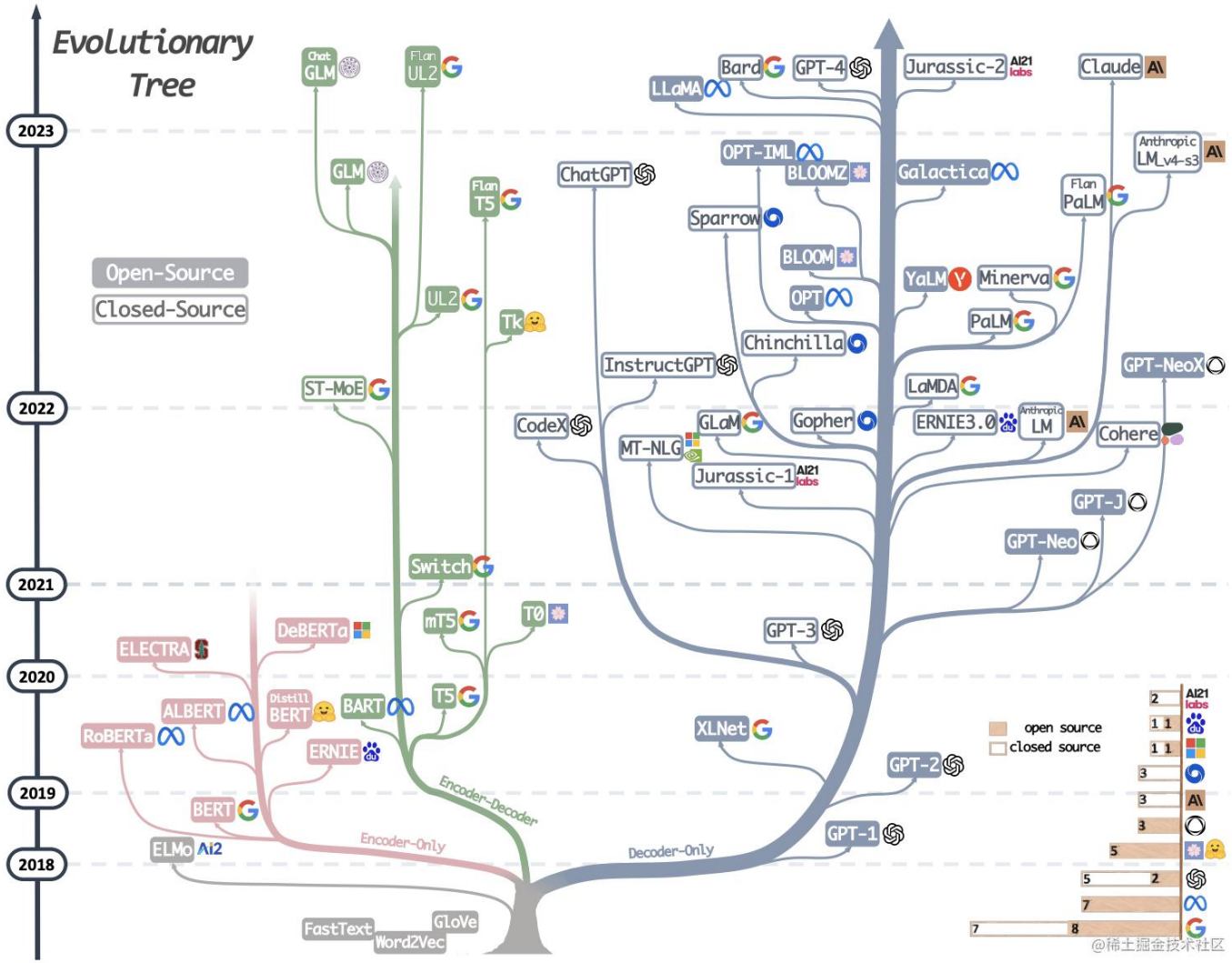
快搜索
优化搜索方法寻找更有价值的搜索空间

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！

智能自主设计的范式——自顶向下

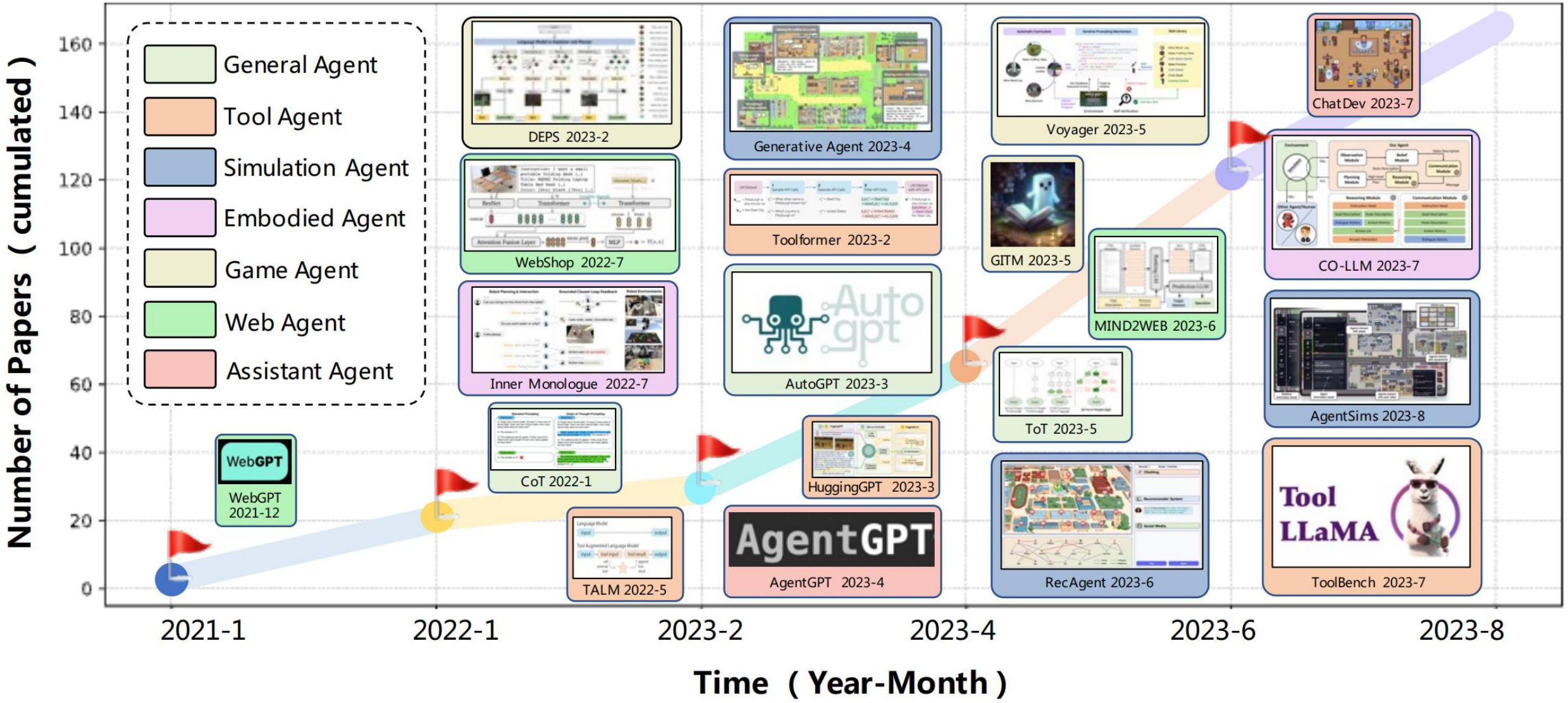


能自顶向下来解决问题的核心驱动：LLM！

语言表征学习

人类通识经验

基于LLMs的智能体设计

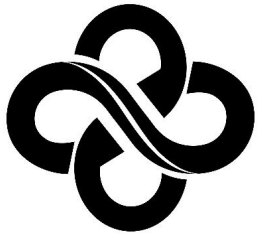


智能自主设计的范式——自顶向下

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().

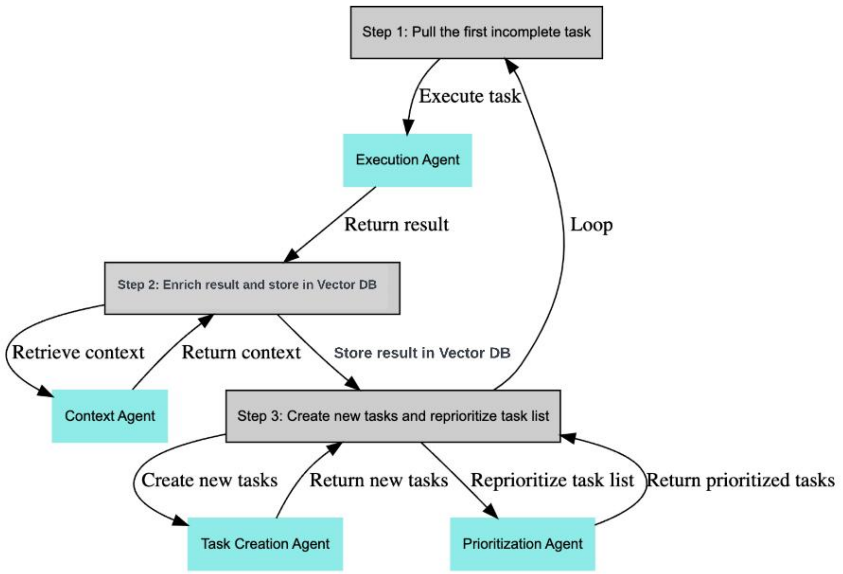


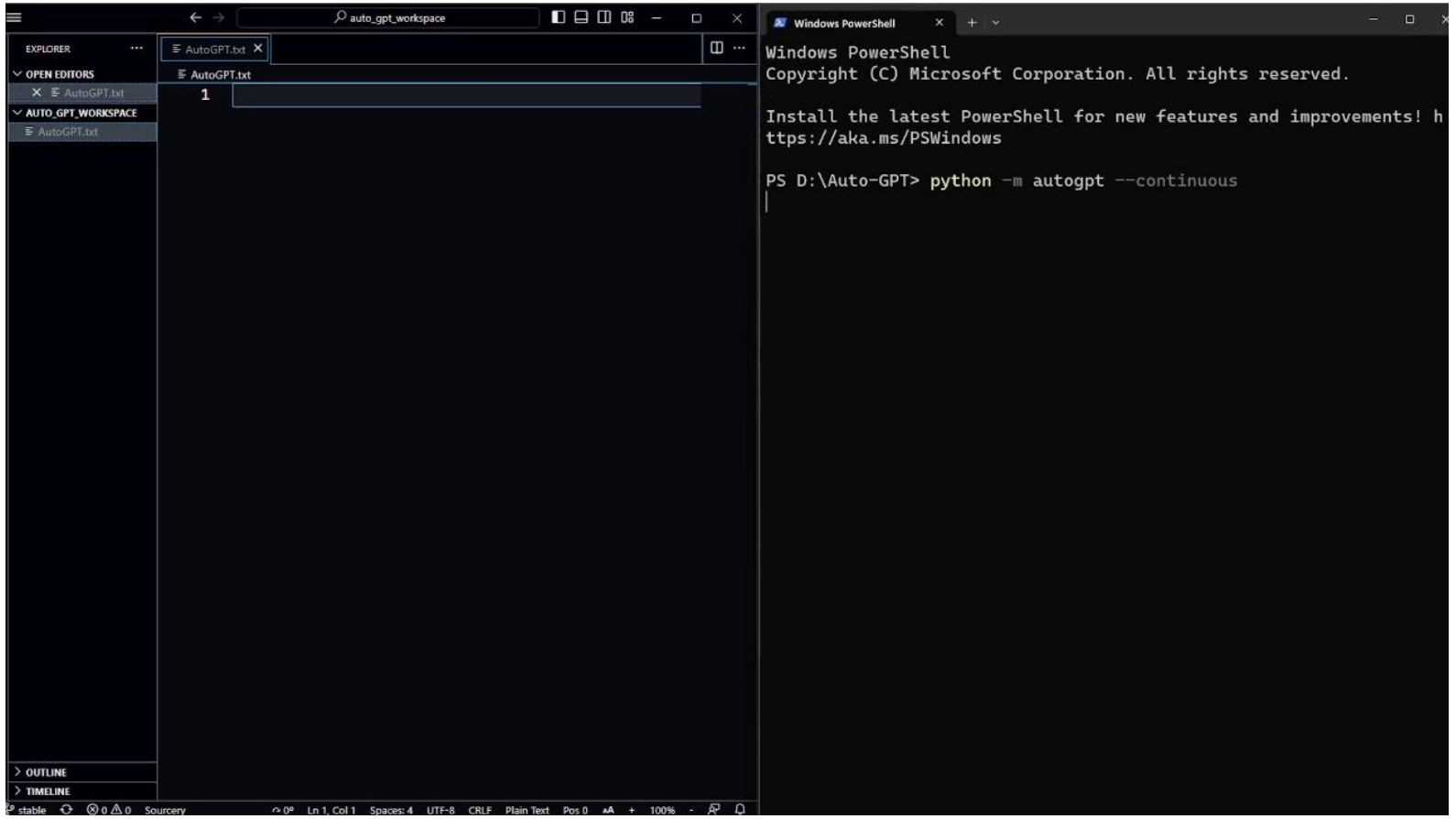
以AutoGPT为代表的大模型解决方案

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

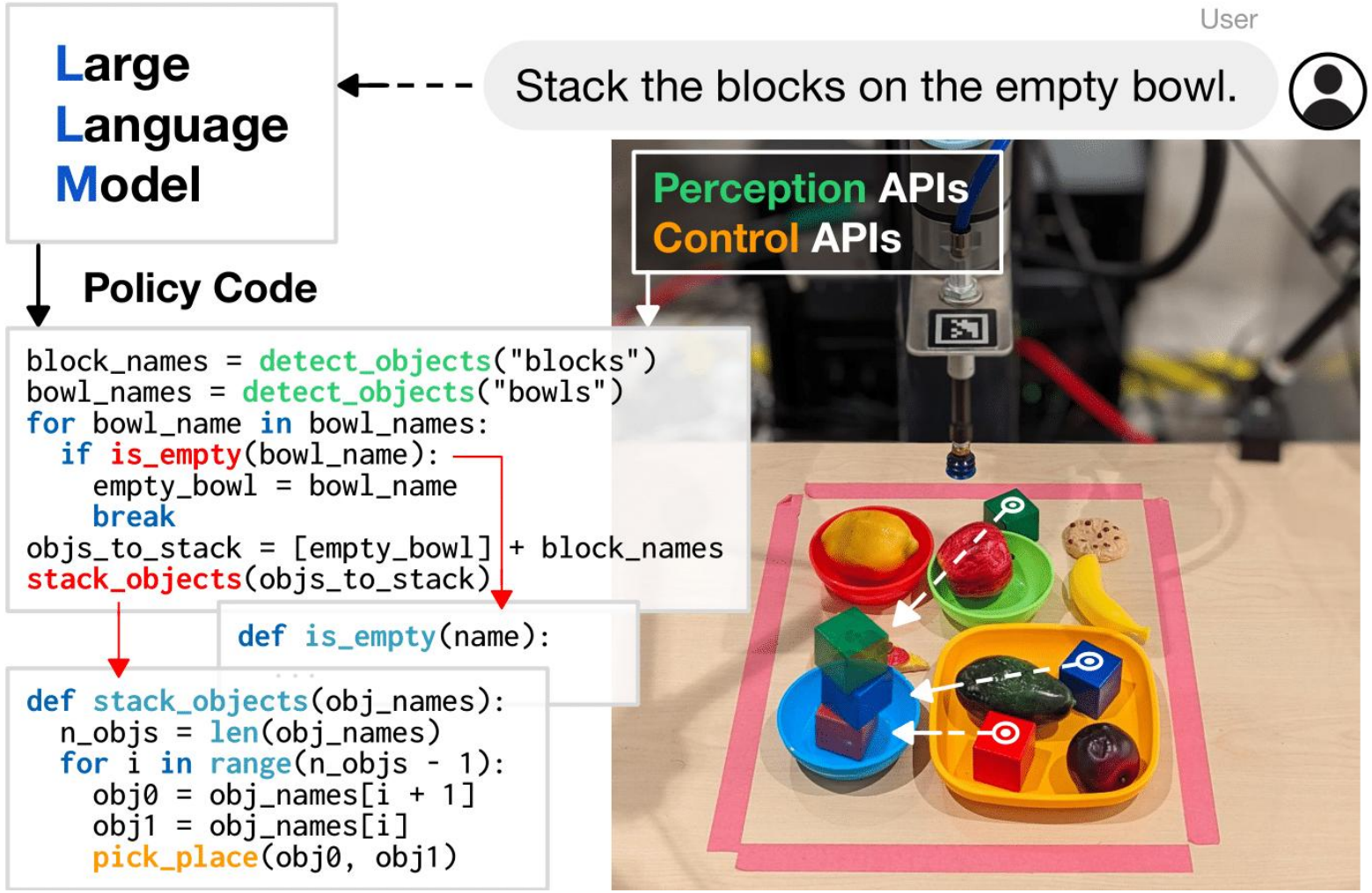
让AutoGPT查询资料学习AutoGPT





智能自主设计的范式——自顶向下

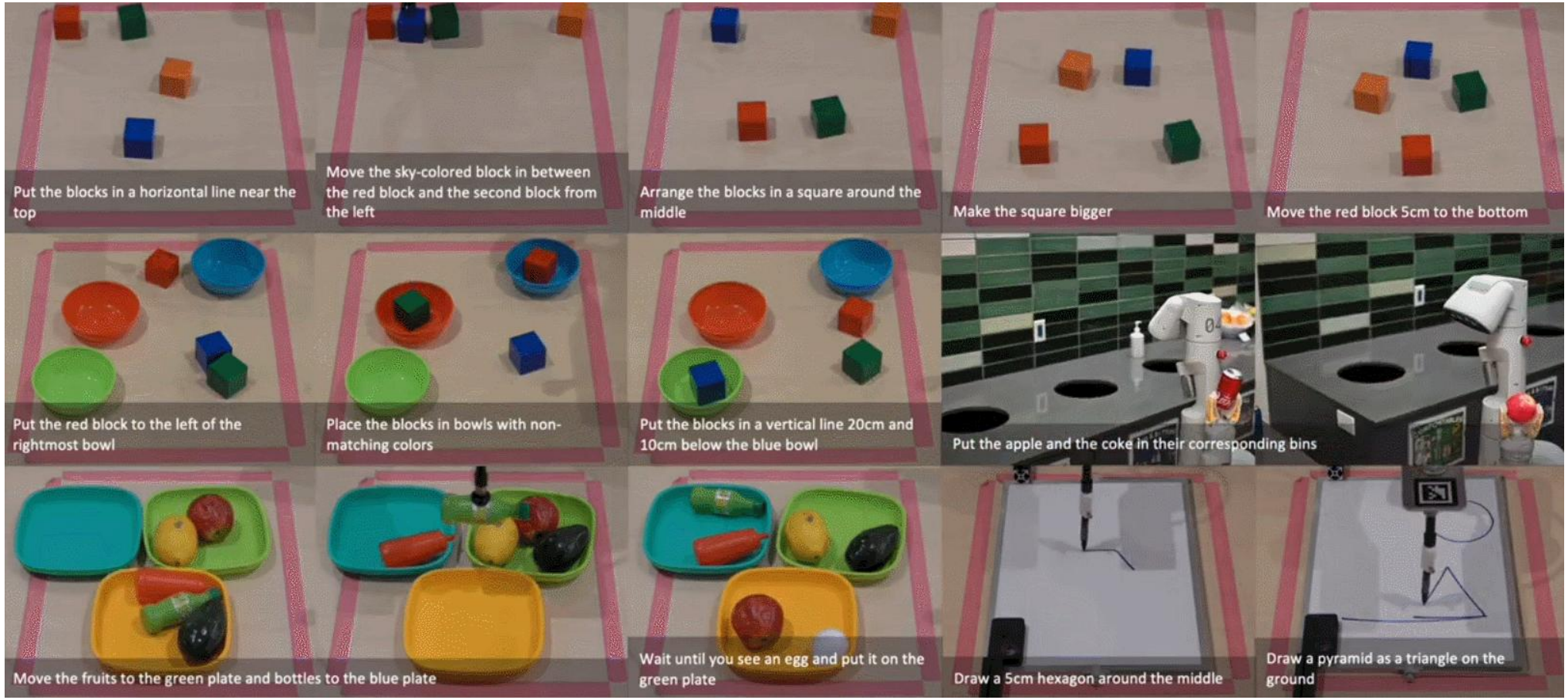
基于大模型和代码生成的自然语言机器控制



Code as Policies: Language Model Programs for Embodied Control

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

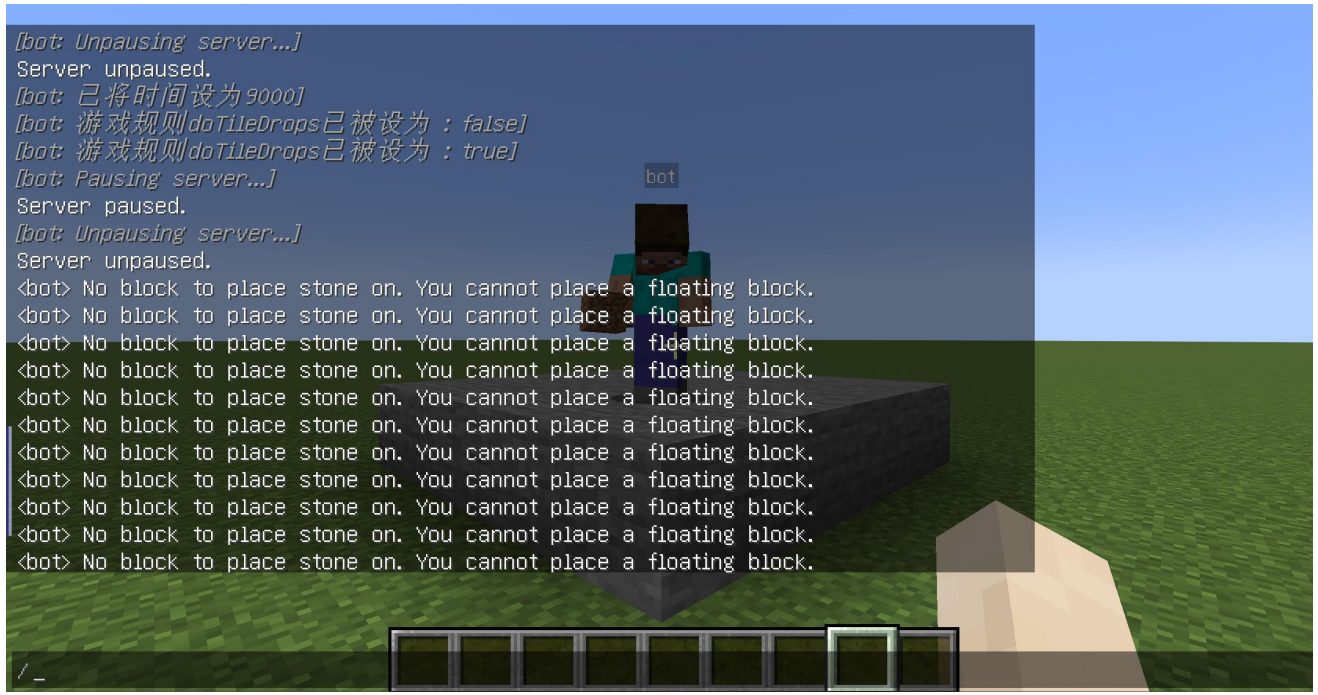


https://code-as-policies.github.io/

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	danikhan632/Auto-GPT-AlpacaTrader-Plugin
AutoGPTReddit	Reddit Access	NeonN3mesis/AutoGPTReddit
AutoGPT User Input Request	Allow Auto-GPT to specifically request user input in continous mode	HFrovinJensen/Auto-GPT-User-Input-Plugin
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	gravelBridge/AutoGPT-BingAI
Crypto	Trade crypto with Auto-GPT	isaiahbjork/Auto-GPT-Crypto-Plugin
Discord	Interact with your Auto-GPT instance through Discord	gravelBridge/AutoGPT-Discord
Dolly AutoGPT Cloner	A way to compose & run multiple Auto-GPT processes that cooperate, till core has multi-agent support	pr-0f3t/Auto-GPT-Dolly-Plugin
Google Analytics	Connect your Google Analytics Account to Auto-GPT.	isaiahbjork/Auto-GPT-Google-Analytics-Plugin
IFTTT webhooks	This plugin allows you to easily integrate IFTTT connectivity using Maker	AntonioCiolino/AutoGPT-IFTTT
iMessage	Send and Get iMessages using Auto-GPT	danikhan632/Auto-GPT-Messages-Plugin
Instagram	Instagram access	jpetzke/AutoGPT-Instagram
Mastodon	Simple Mastodon plugin to send toots through a Mastodon account	ppetermann/AutoGPTMastodonPlugin

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

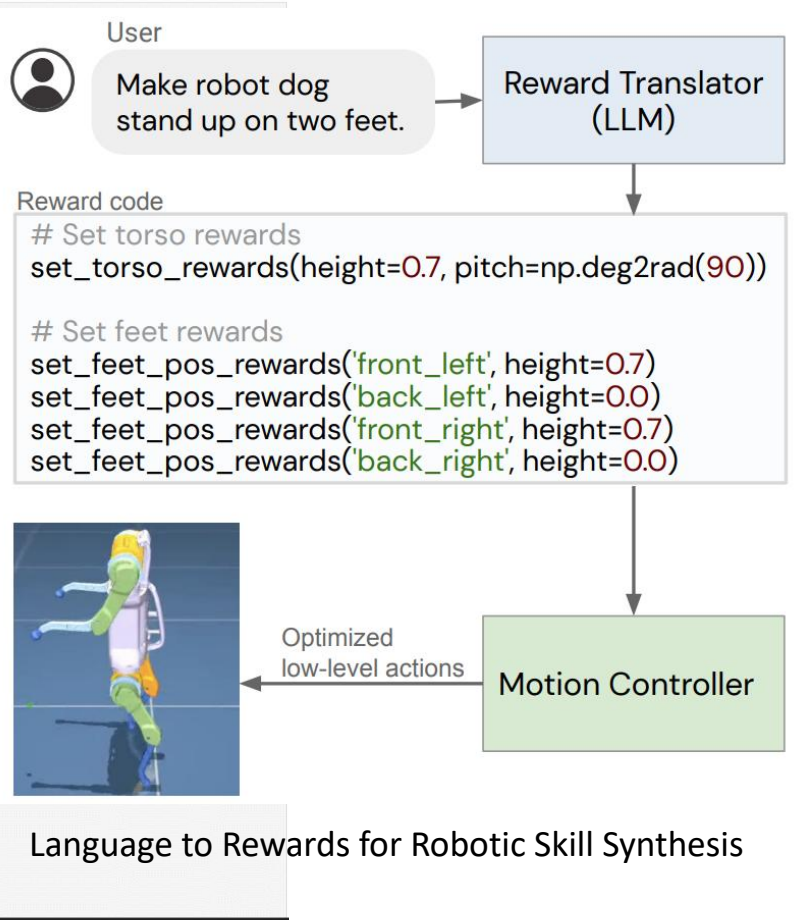
AutoGPT针对各种应用适配插件

Plugin	Description	Repository
Alpaca-Trading	Trade stocks and crypto, paper or live with Auto-GPT	danikhan632/Auto-GPT-AlpacaTrader-Plugin
AutoGPTReddit	Reddit Access	NeonN3mesis/AutoGPTReddit
AutoGPT User Input Request	Allow Auto-GPT to specifically request user input in continous mode	HFrovinJensen/Auto-GPT-User-Input-Plugin
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	gravelBridge/AutoGPT-BingAI
Crypto	Trade crypto with Auto-GPT	isaiahbjork/Auto-GPT-Crypto-Plugin
Discord	Interact with your Auto-GPT instance through Discord	gravelBridge/AutoGPT-Discord
Dolly AutoGPT Cloner	A way to compose & run multiple Auto-GPT processes that cooperate, till core has multi-agent support	pr-0f3t/Auto-GPT-Dolly-Plugin
Google Analytics	Connect your Google Analytics Account to Auto-GPT.	isaiahbjork/Auto-GPT-Google-Analytics-Plugin
IFTTT webhooks	This plugin allows you to easily integrate IFTTT connectivity using Maker	AntonioCiolino/AutoGPT-IFTTT
iMessage	Send and Get iMessages using Auto-GPT	danikhan632/Auto-GPT-Messages-Plugin
Instagram	Instagram access	jpetzke/AutoGPT-Instagram
Mastodon	Simple Mastodon plugin to send toots through a Mastodon account	ppetermann/AutoGPTMastodonPlugin

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_si
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(f'Sure - stacking from top to bottom: {", ".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 betweeen 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)
```



Language to Rewards for Robotic Skill Synthesis

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

生成基于machine OP的程序

Machine可执行的OP空间

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

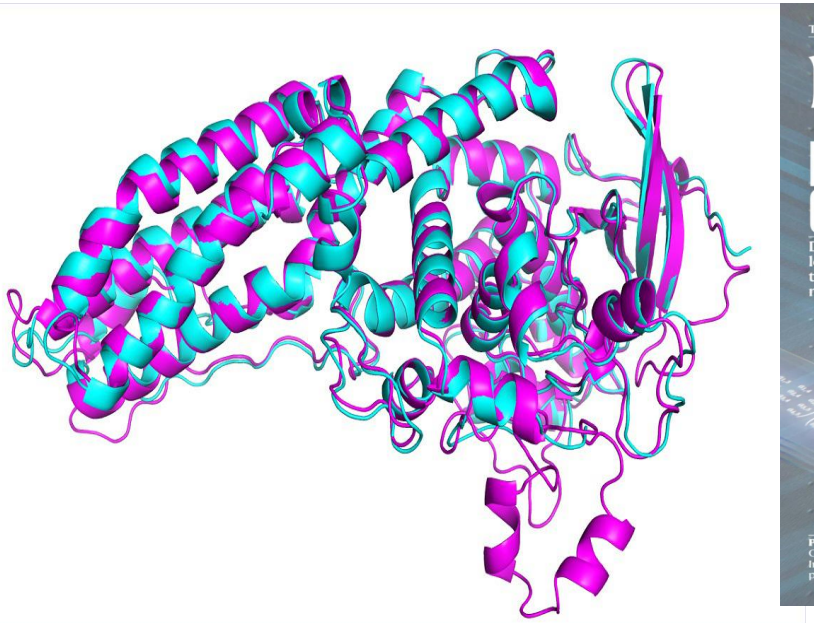
问题：
脱离环境、不可执行

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

DIGITALAGENT

生成基于machine OP的程序

Machine可执行的OP空间

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

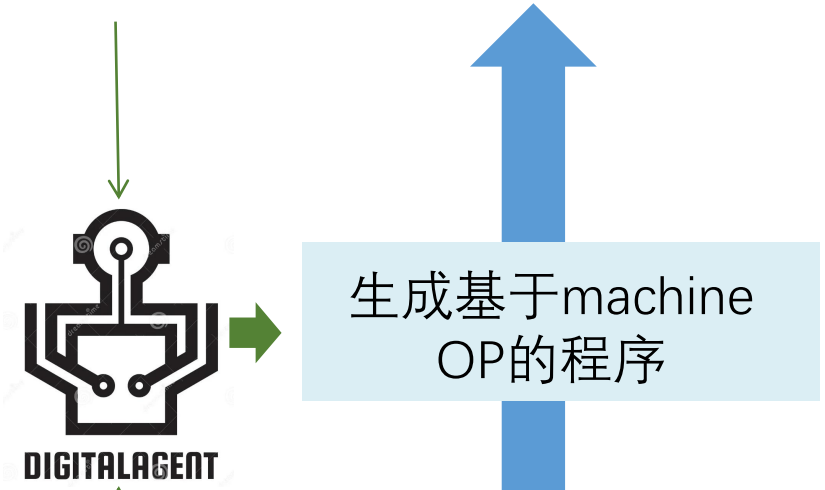
快搜索
优化搜索方法寻找更有价值的搜索空间

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

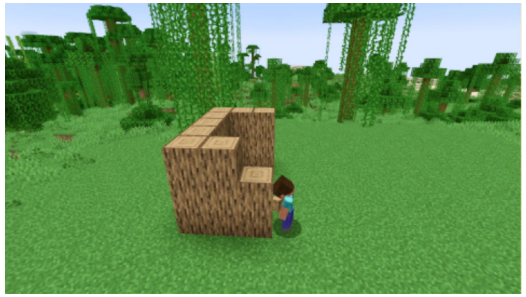
快搜索
优化搜索方法寻找更有价值的搜索空间

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

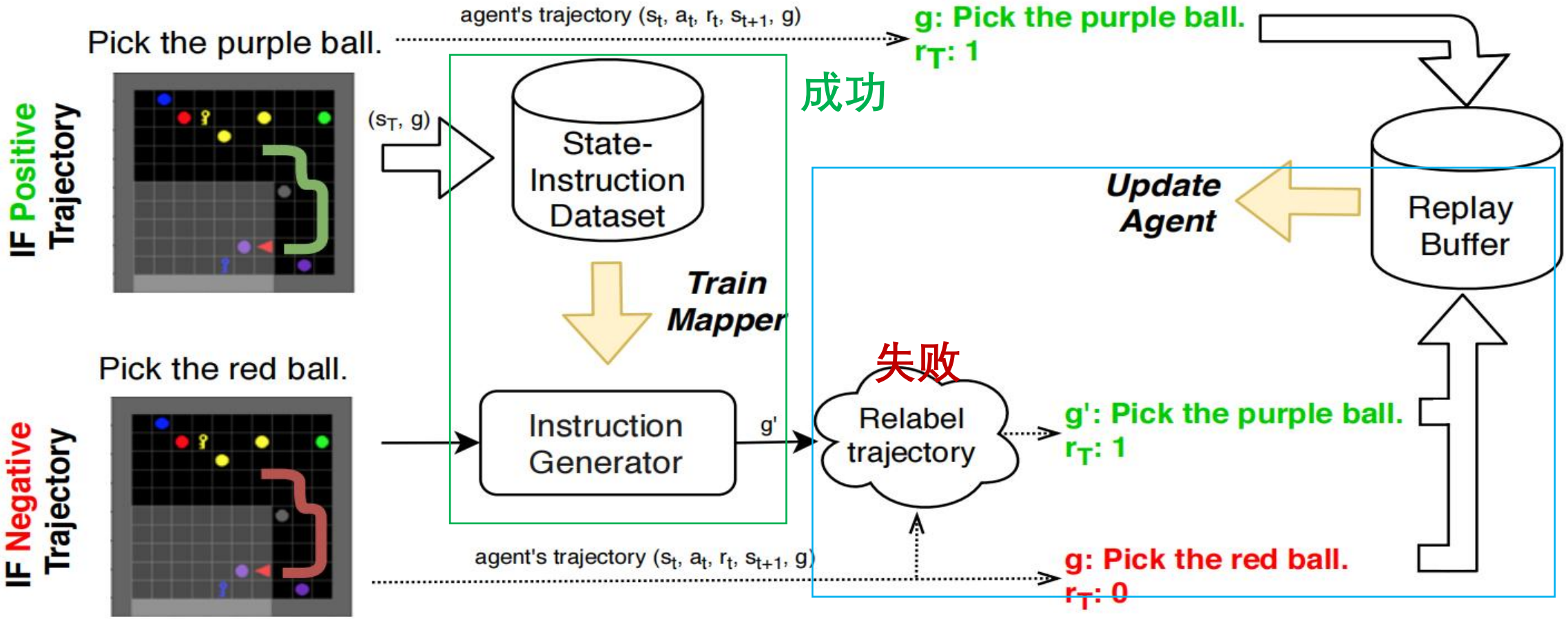
“我的世界”中指令遵循任务

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

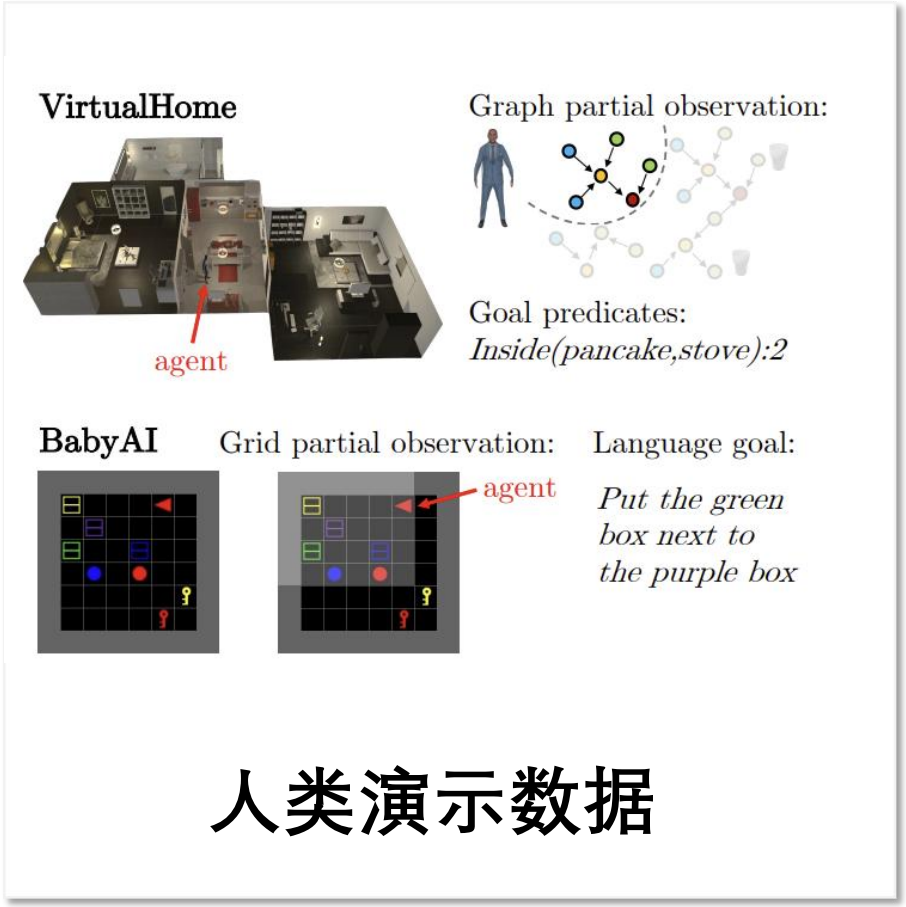
关键挑战：稀疏奖励、空间大、轨迹长 ⟵ 数据重标



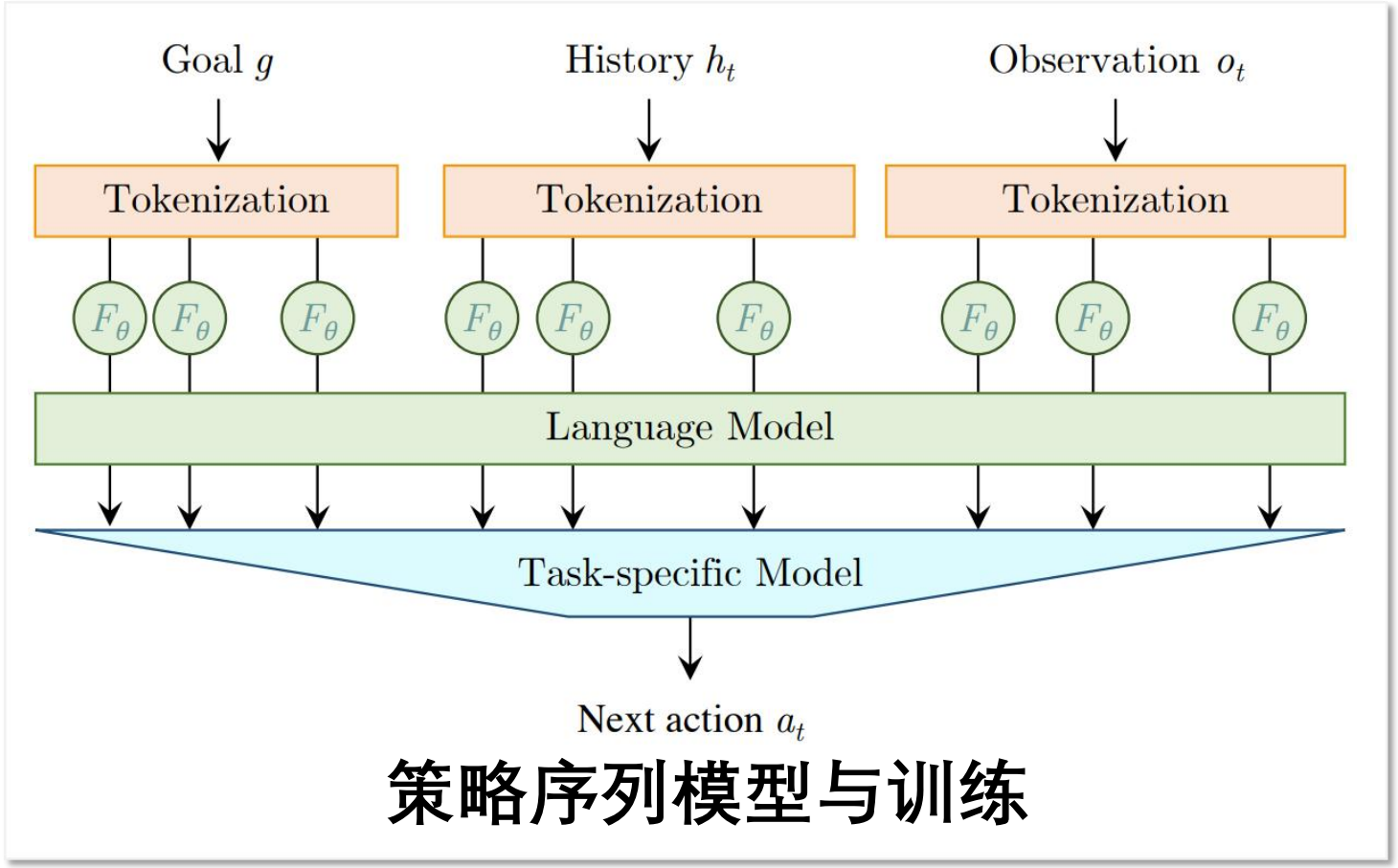
Hindsight Generation for Experience Replay (HIGhER

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



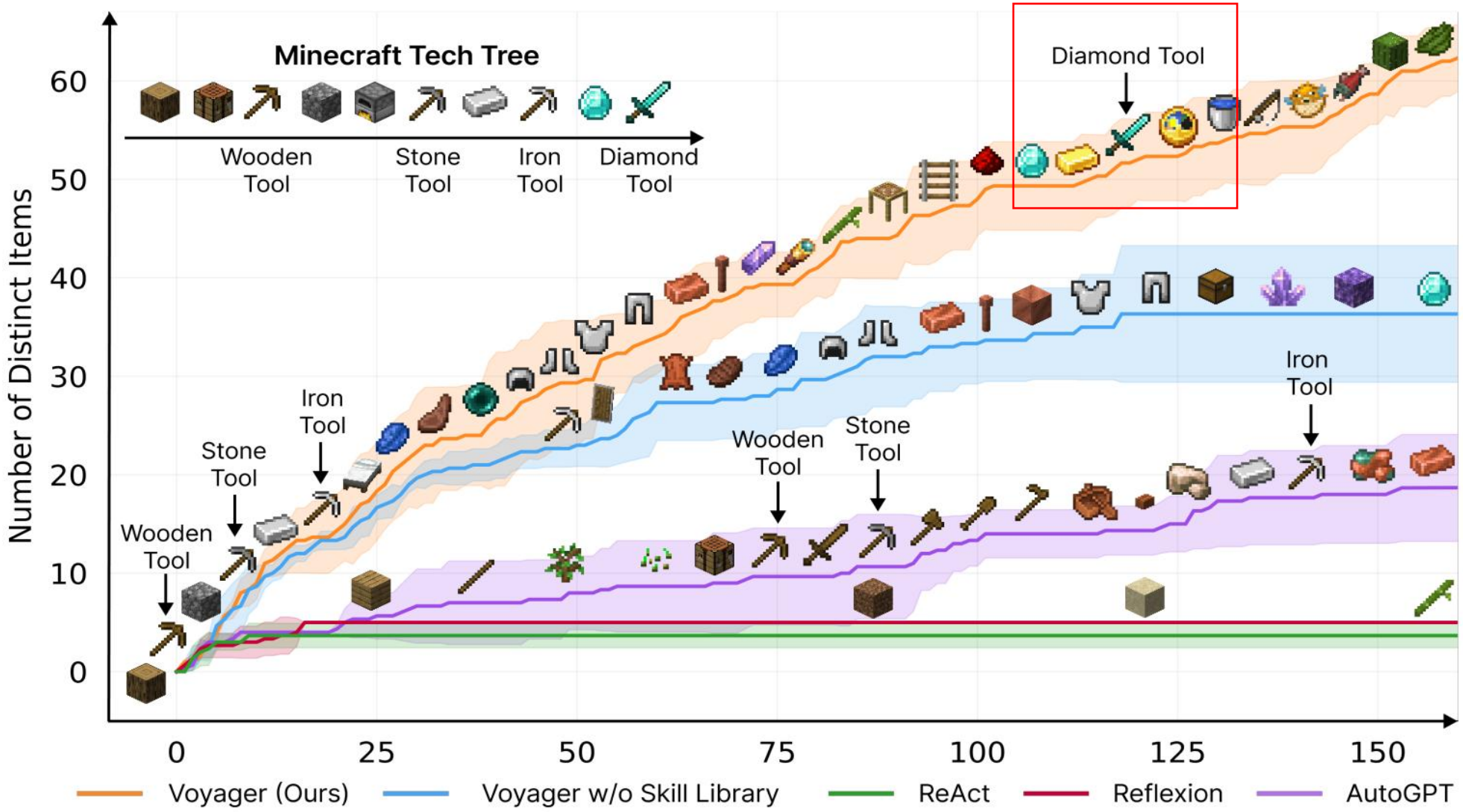
人类演示数据



策略序列模型与训练

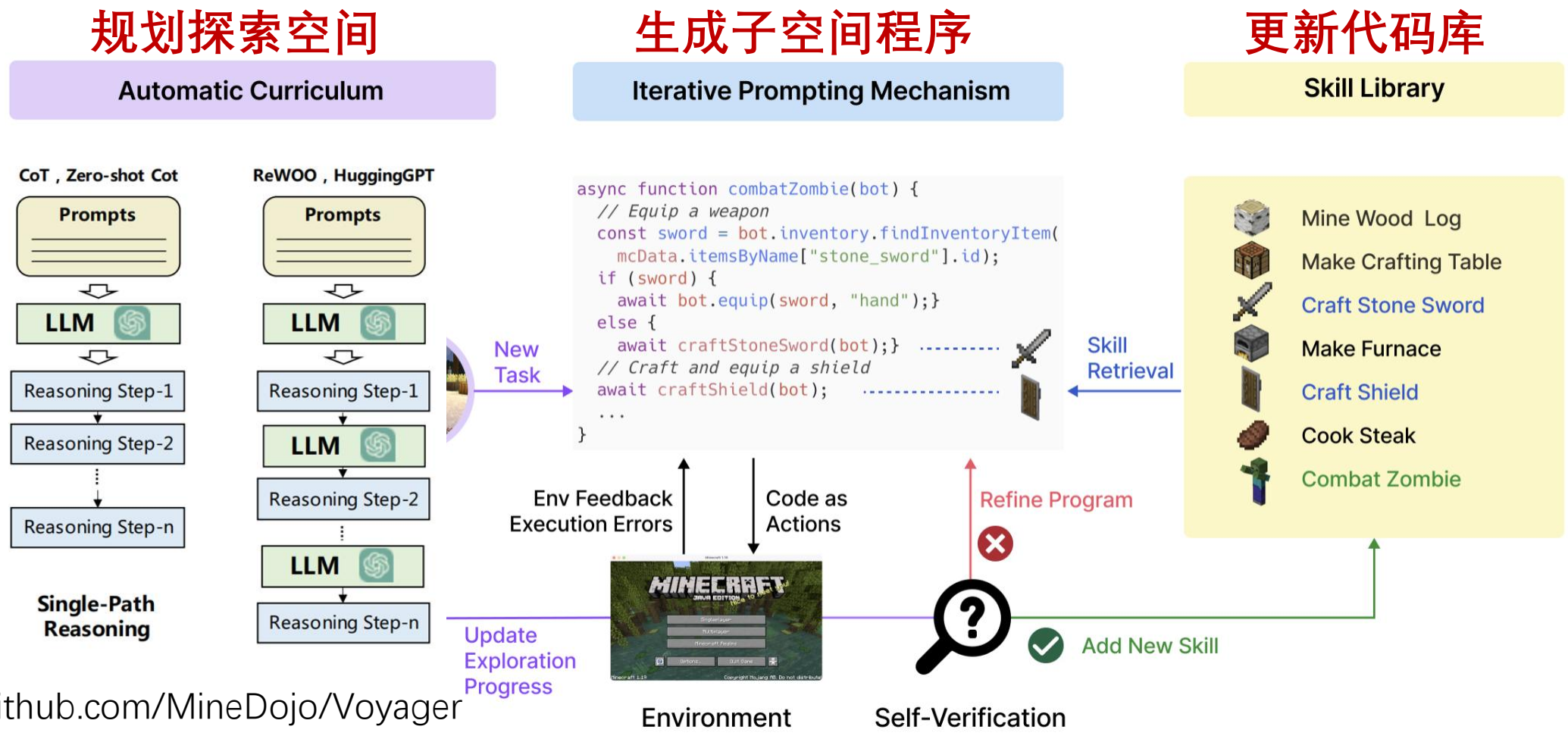
Pre-Trained Language Models for Interactive Decision-Making (LID), NeurIPS2022

LLM 具身智能：以“我的世界”为例



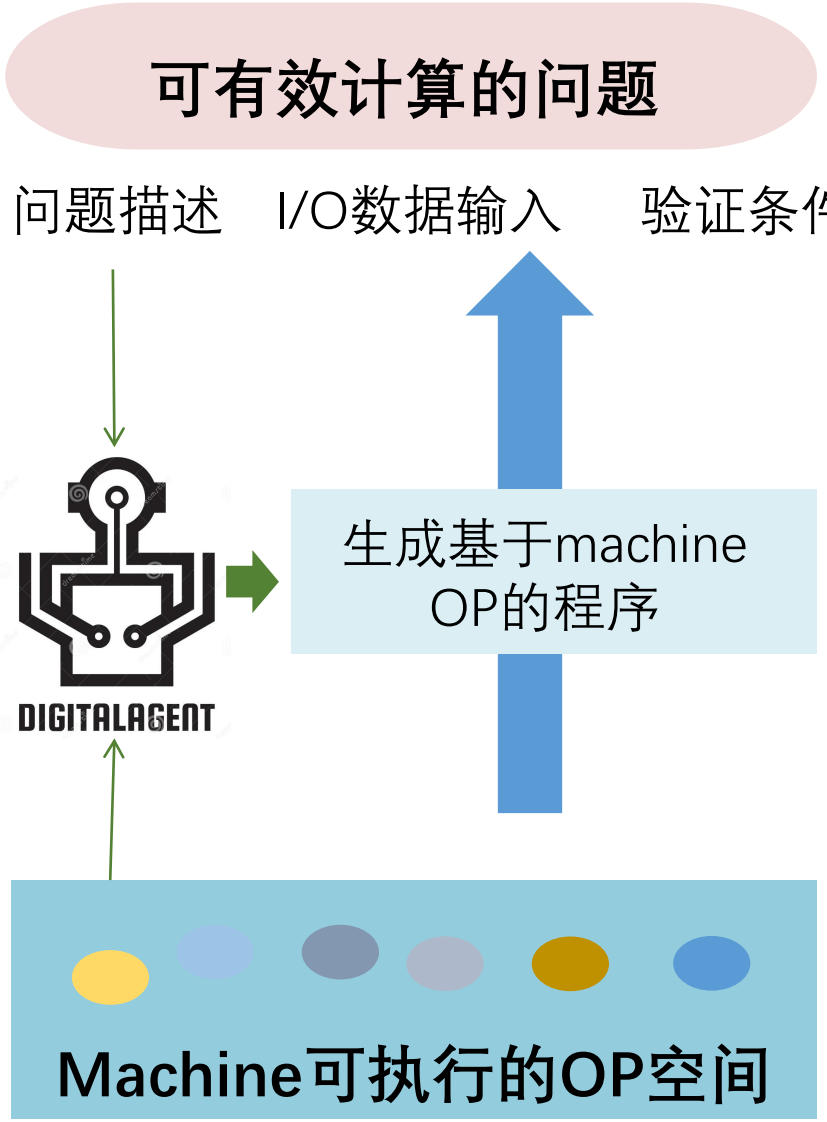
LLM 具身智能

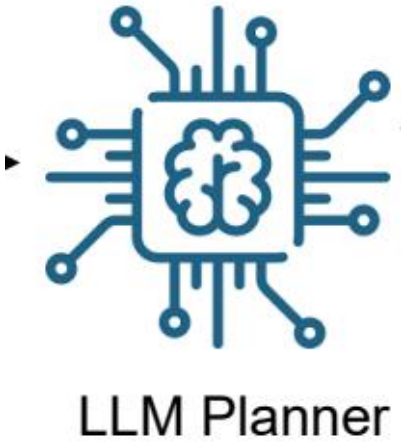
以程序生成的形式完成原本是强化学习的任务



https://github.com/MineDojo/Voyager

LLM Embodied Agents挑战



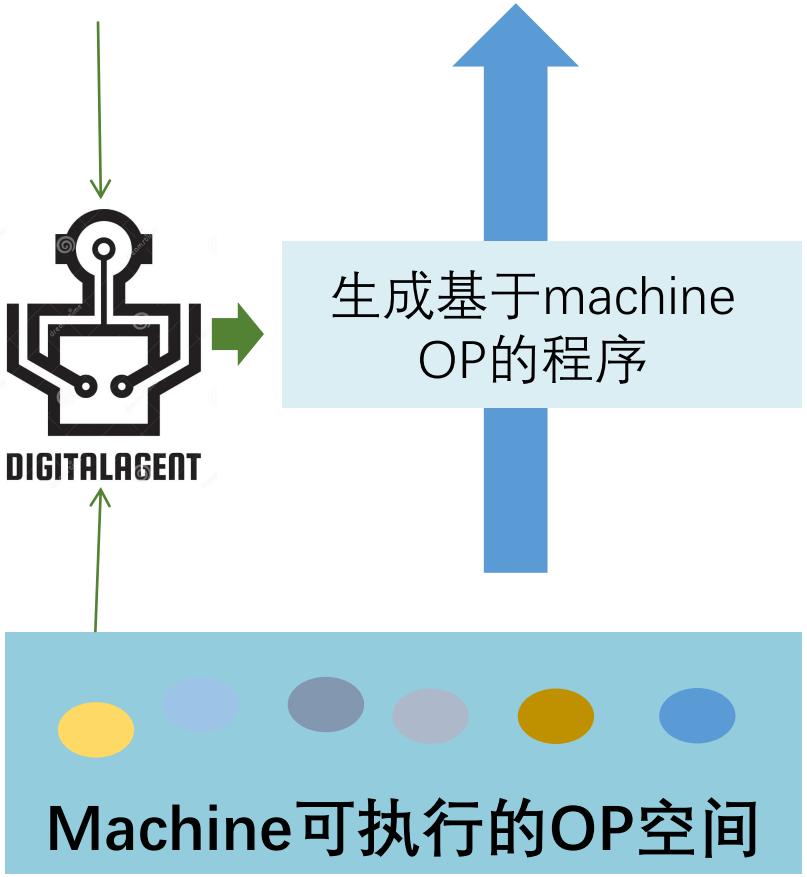


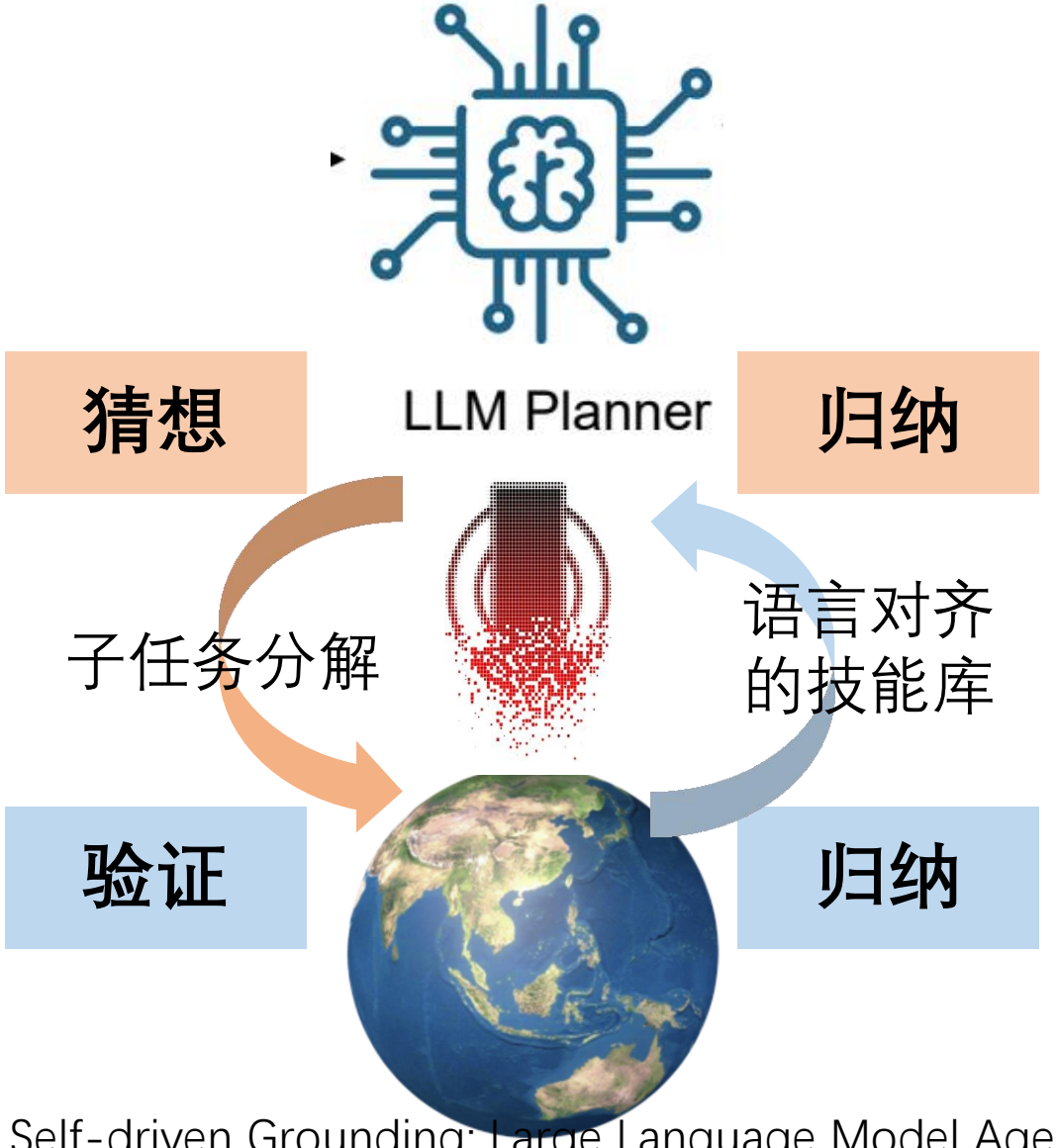
语义世界和
环境世界的鸿沟

自动语言对齐的强化学习

可有效计算的问题

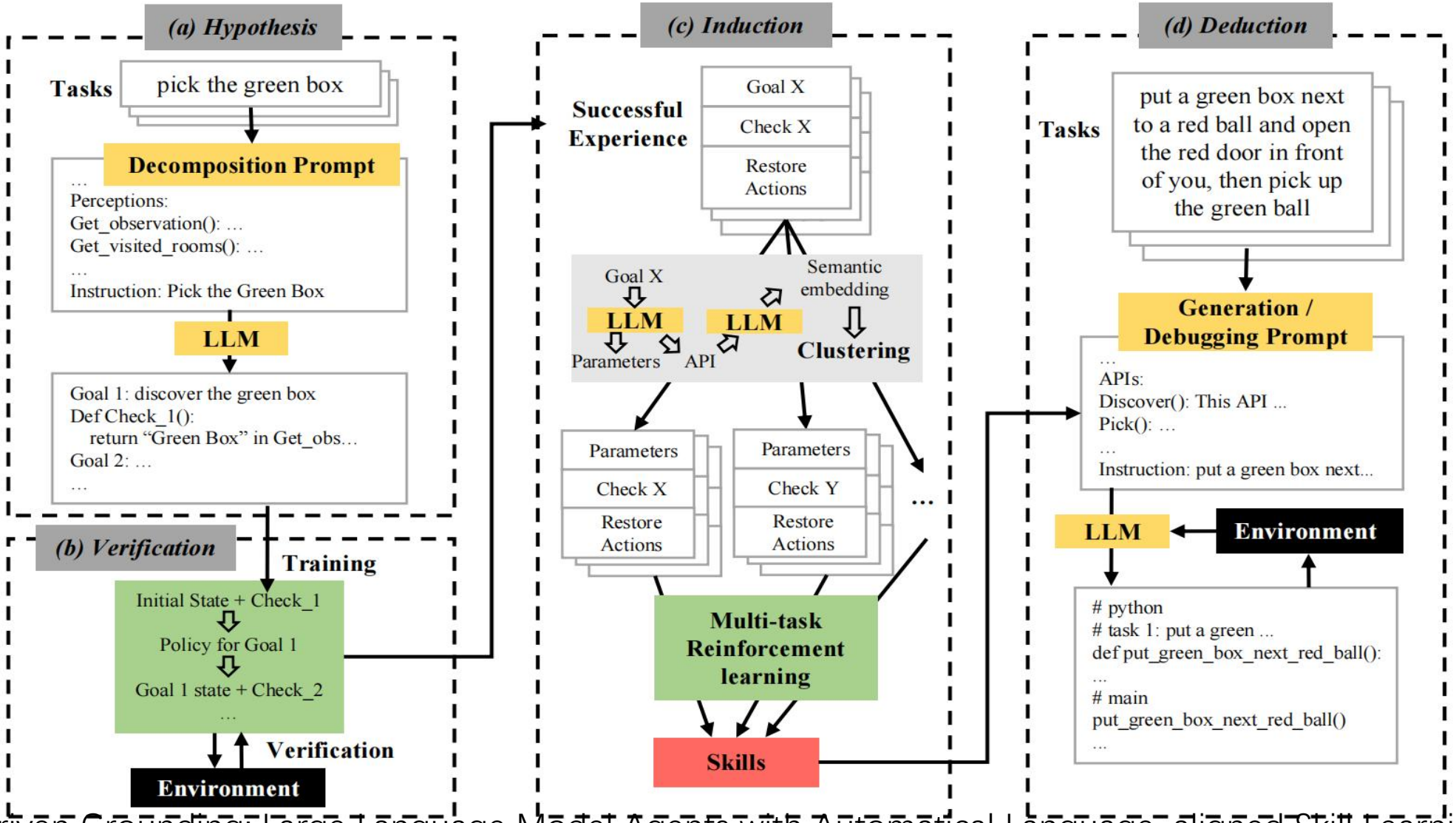
问题描述　I/O数据输入　验证条件





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 Automatical Language-aligned Skill Learning, arxiv23

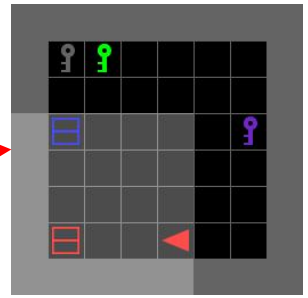
自动化对齐：猜想验证

- 验证结果示例

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()
```

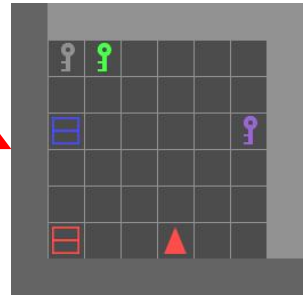
Discover the green key



(2) Check if the robot is next to the green key.

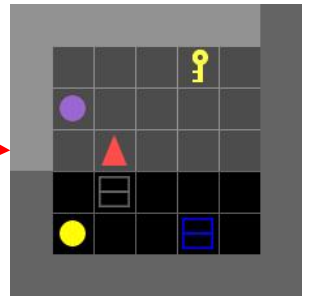
```python
def check_next_to_green_key(robot):
    obs = robot.get_observation()
    if "green key" in obs.keys():
        return sum(abs(obs["green key"])) == 1
    else:
        return False
```

Next to the green key



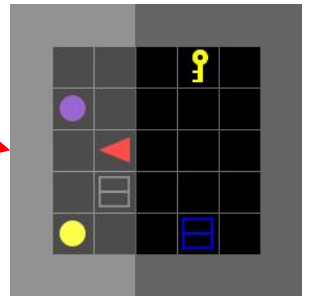
Task instruction: "Pick up the yellow ball"
Goals:
(1) Discover the yellow ball.

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



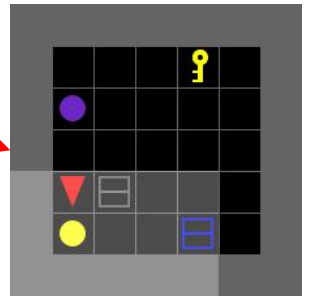
(2) Go next to the yellow ball.

```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
```



(3) Pick up the yellow ball.

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

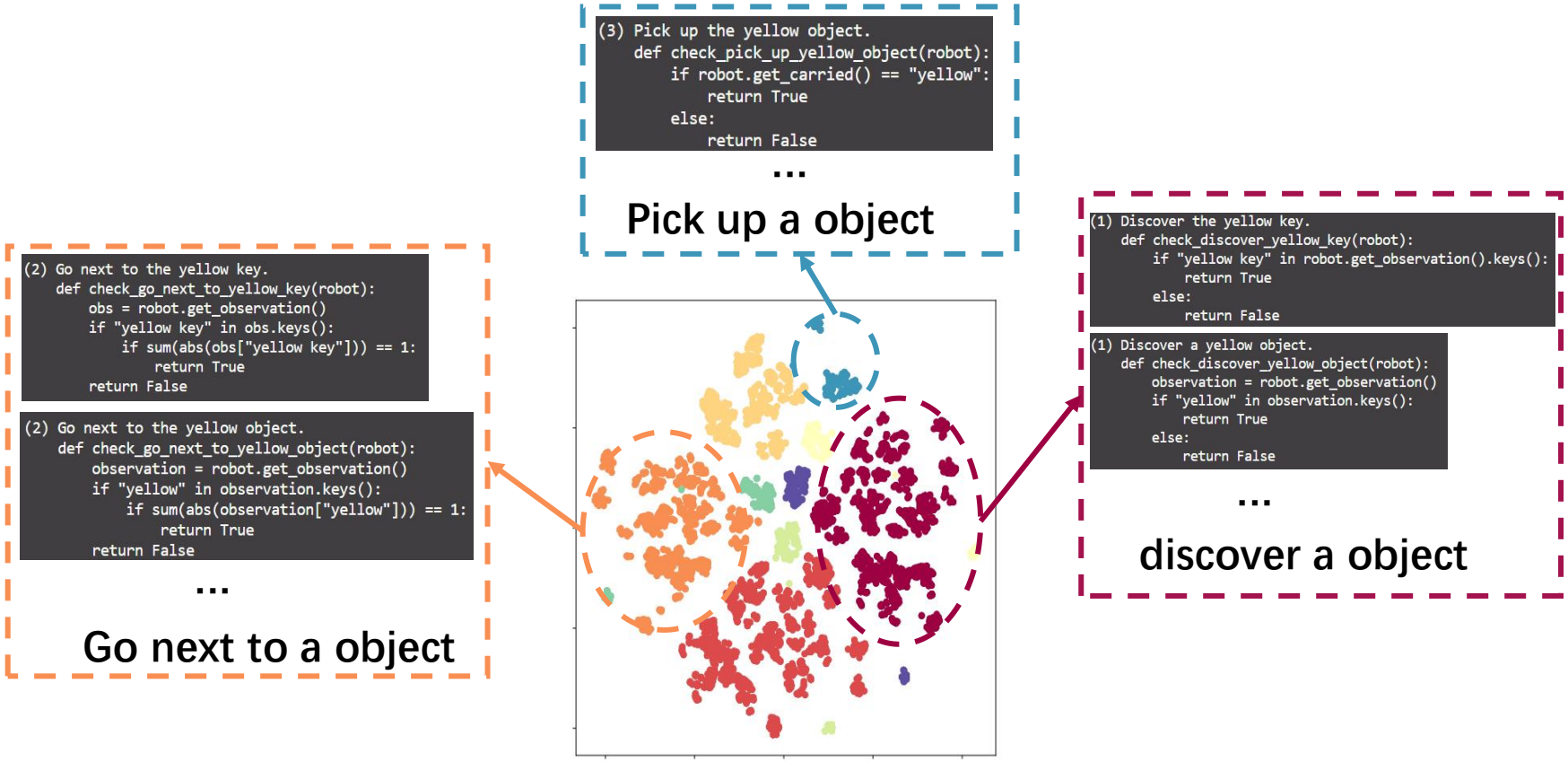


足所有子目标和任务完成，每个子目标及其检查函数作为成功“对齐”的经验被收集

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

```
# Candidate program:
# task 1: put the red key next to the yellow key
def put_red_key_next_to_yellow_key(robot):
    # 1. discover the red key and yellow key
    robot.discover_object("red key")
    robot.discover_object("yellow key")

    # 2. go to the red key
    robot.go_next_to("red key")

    # 3. pick the red key
    robot.pick("red key")

    # 4. go to the yellow key
    robot.go_next_to("yellow key")

    # 5. put the red key next to the yellow key
    robot.put_next_to("red key", "yellow key")
    return

# main
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!
```

debuggin
g

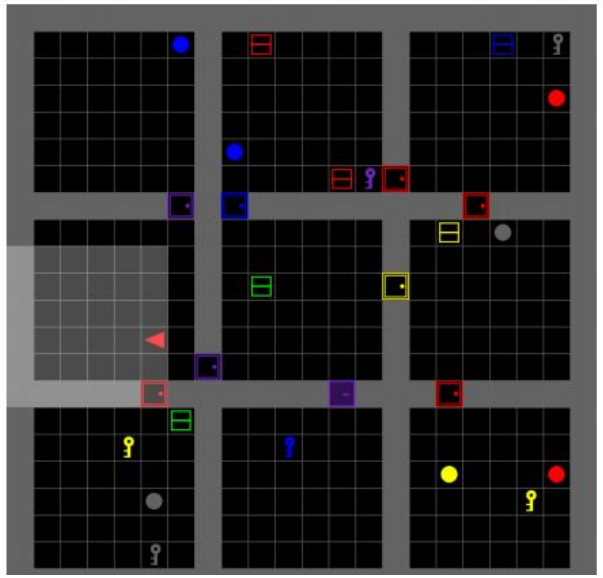
ChatGPT:

```
# Modified program begin:
# task 1: put the red key next to the yellow key
def put_red_key_next_to_yellow_key(robot):
    # 1. discover the red key
    robot.discover_object("red key")

    # 2. go to the red key
    if "red key" in robot.get_observation().keys():
        robot.go_next_to("red key")
    else:
        raise ValueError("The robot didn't discover the red key before go to it")

    # 3. pick the red key
    if sum(abs(robot.get_observation()["red key"])) == 1:
        robot.pick("red key")
    else:
        raise ValueError("The robot is not next to the red key")
```

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	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

演示模仿
演示模仿
LLM+基础action空间

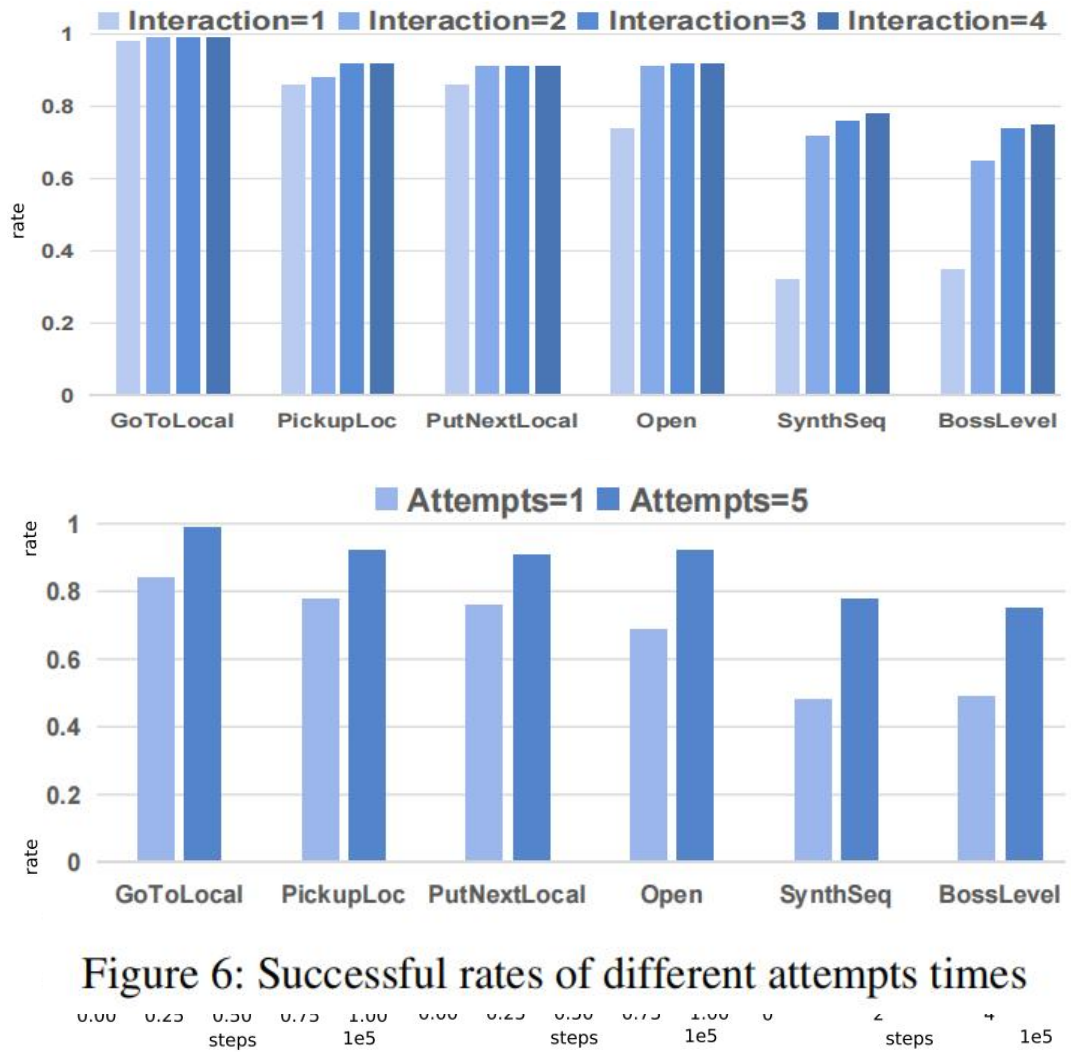


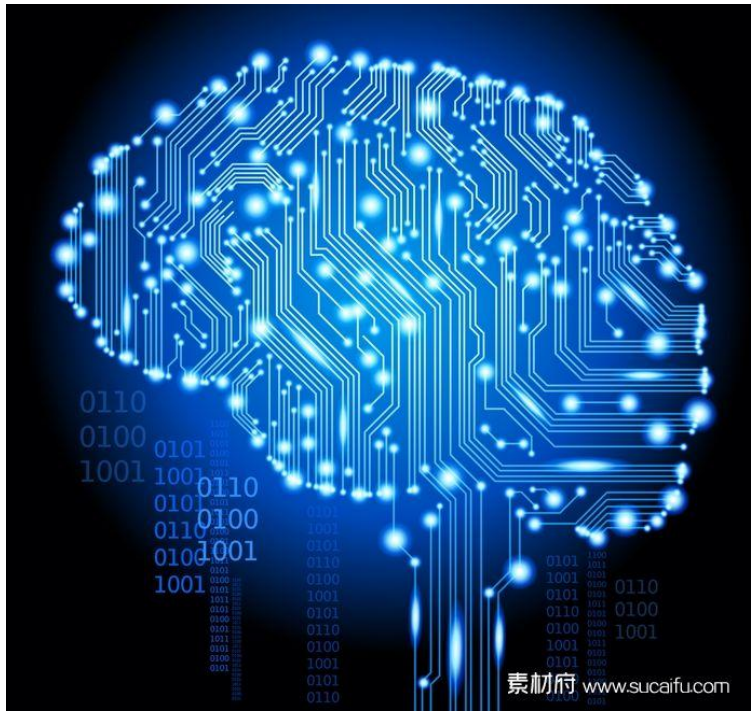
Figure 6: Successful rates of different attempts times

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

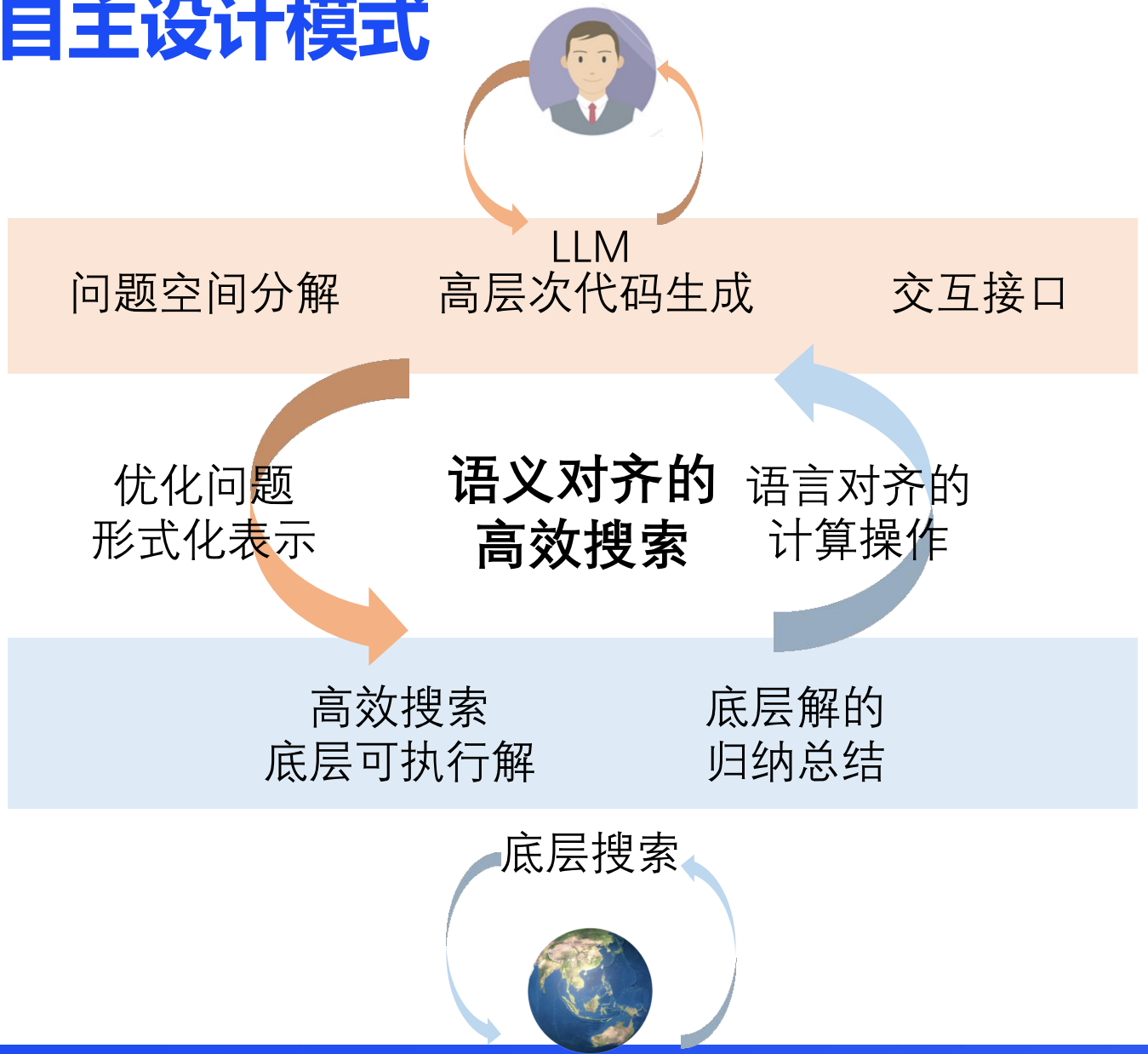
语义世界



模拟环境世界

真实环境世界

智能自主设计模式



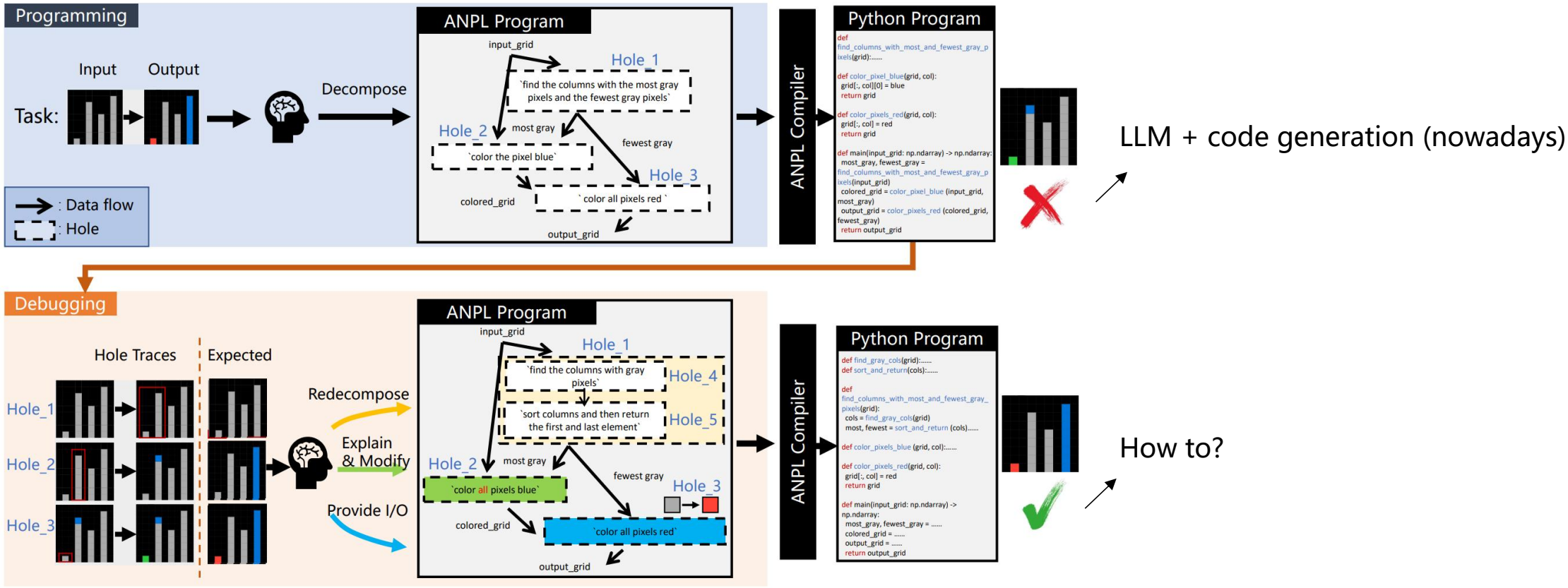
交互基础设施

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

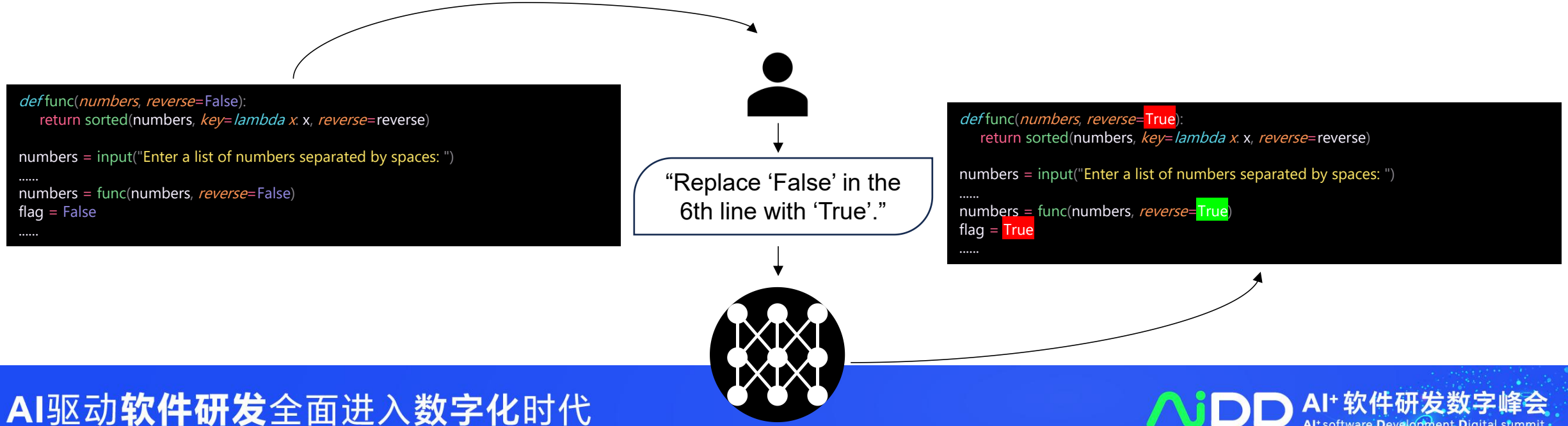
大模型的难预测性和不稳定性导致无法通过交互保证正确性收敛

普通程序：随着交互进行程序向正确性收敛

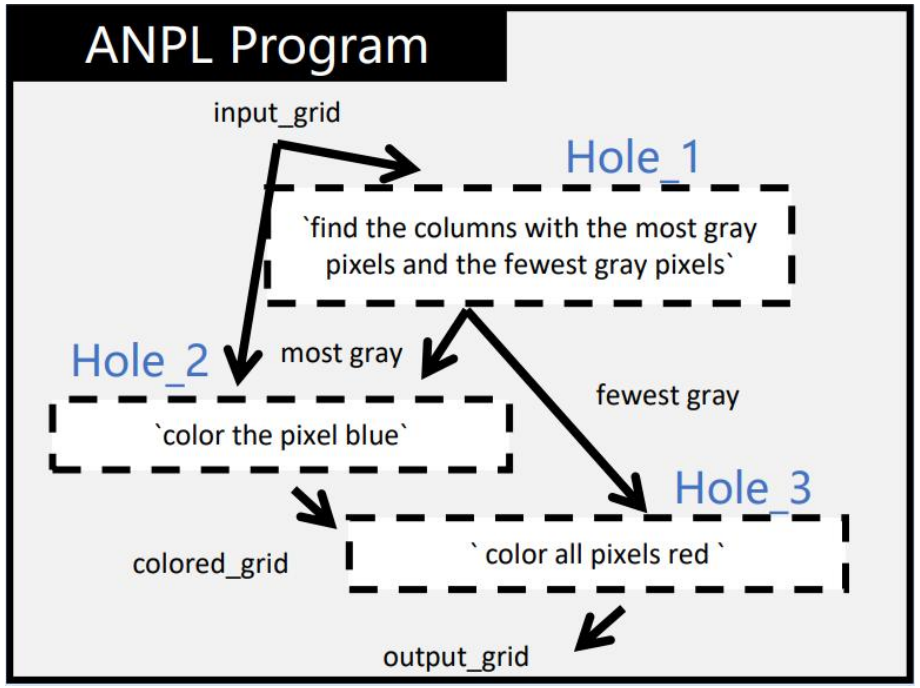
自然语言程序：大模型具有难预测性和不稳定性

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

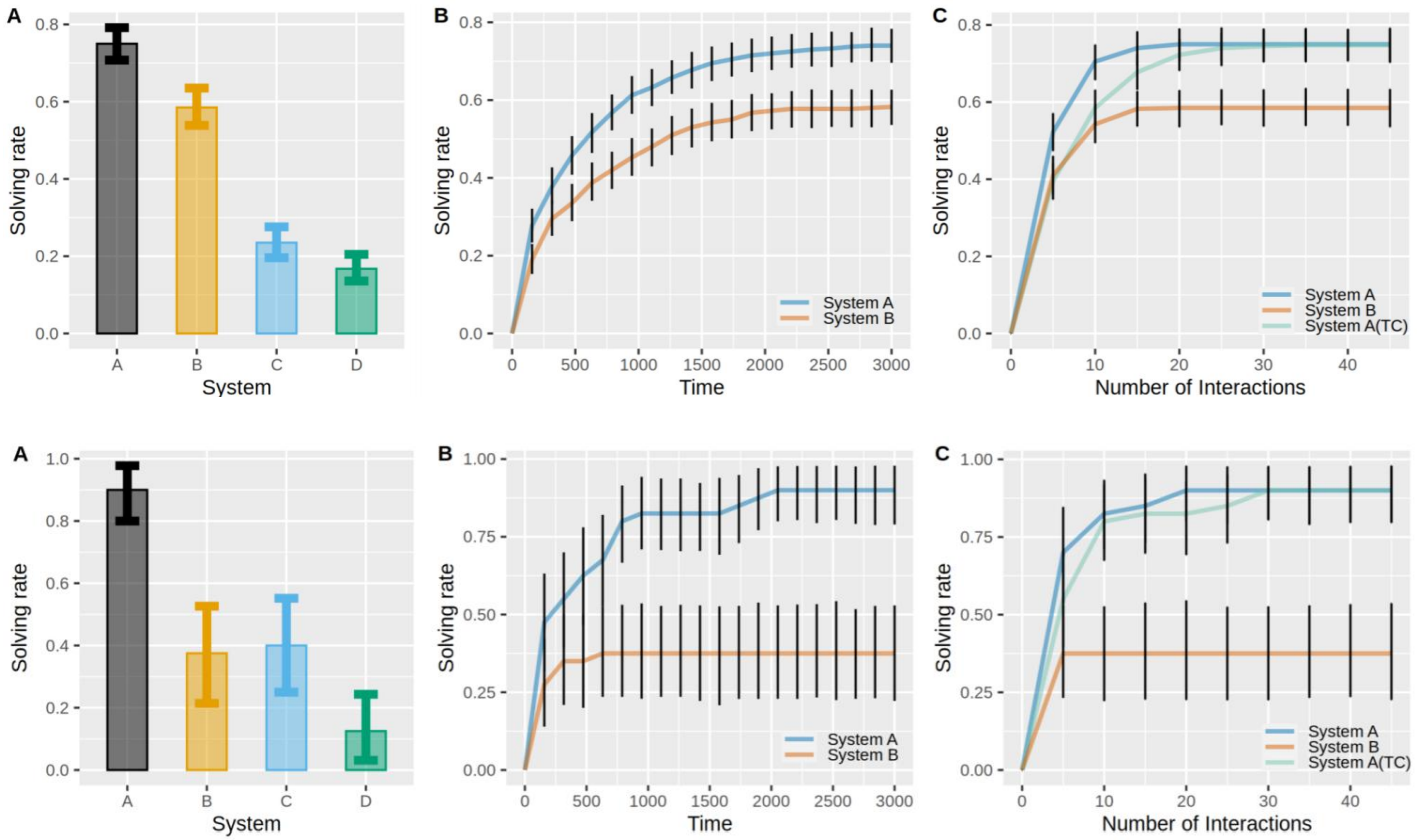
- 19 primary Pythoners
  - System A: ANPL
  - System B: ChatGPT
  - Random orders of A/B

- 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

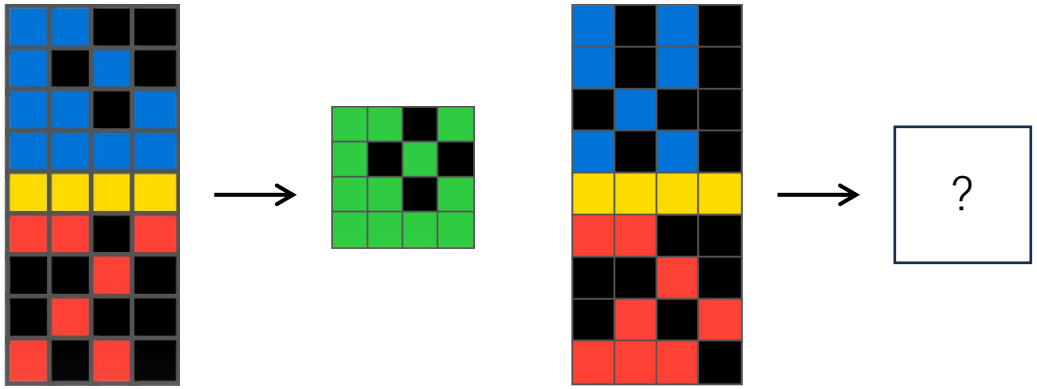
DARC: A Recursive Decomposition Dataset of ARC Tasks

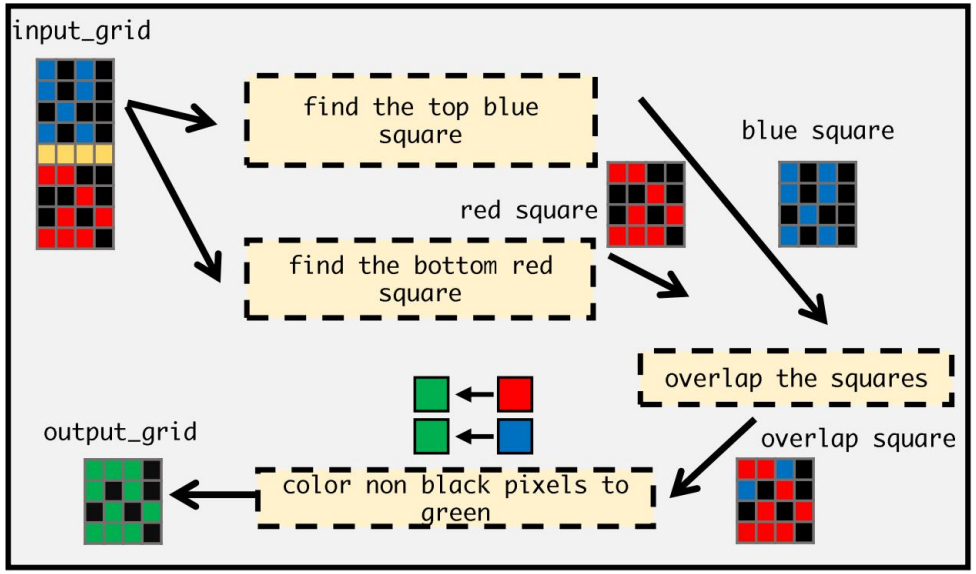
human分解策略

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可以用于编写相对的复杂任务

- 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）

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.

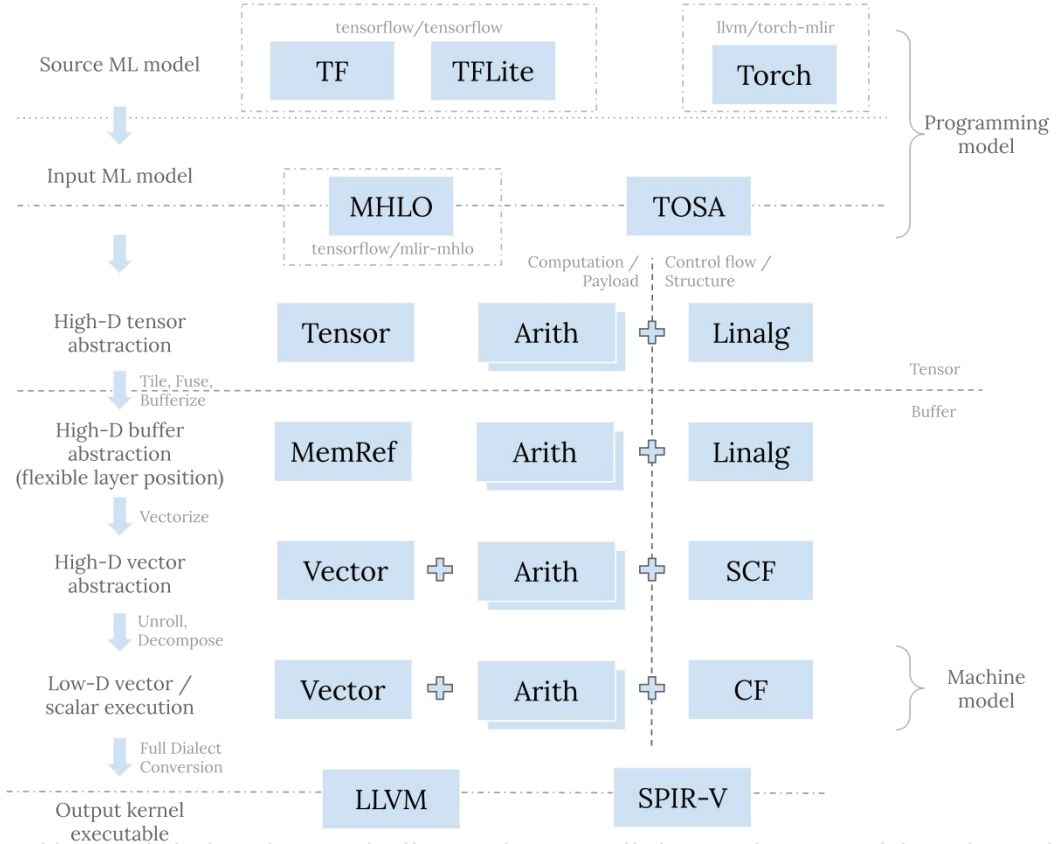
[NeurIPS-23] Di Huang, Ziyuan Nan, Xing Hu, Pengwei Jin, Shaohui Peng, Yuanbo Wen, Rui Zhang, Zidong Du, Qi Guo, Yewen Pu, Yunji Chen. "ANPL: Compiling Natural Programs with Interactive Decomposition". Neural Information Processing Systems (NeurIPS), 2023

对齐基础设施

系统领域带来的启发

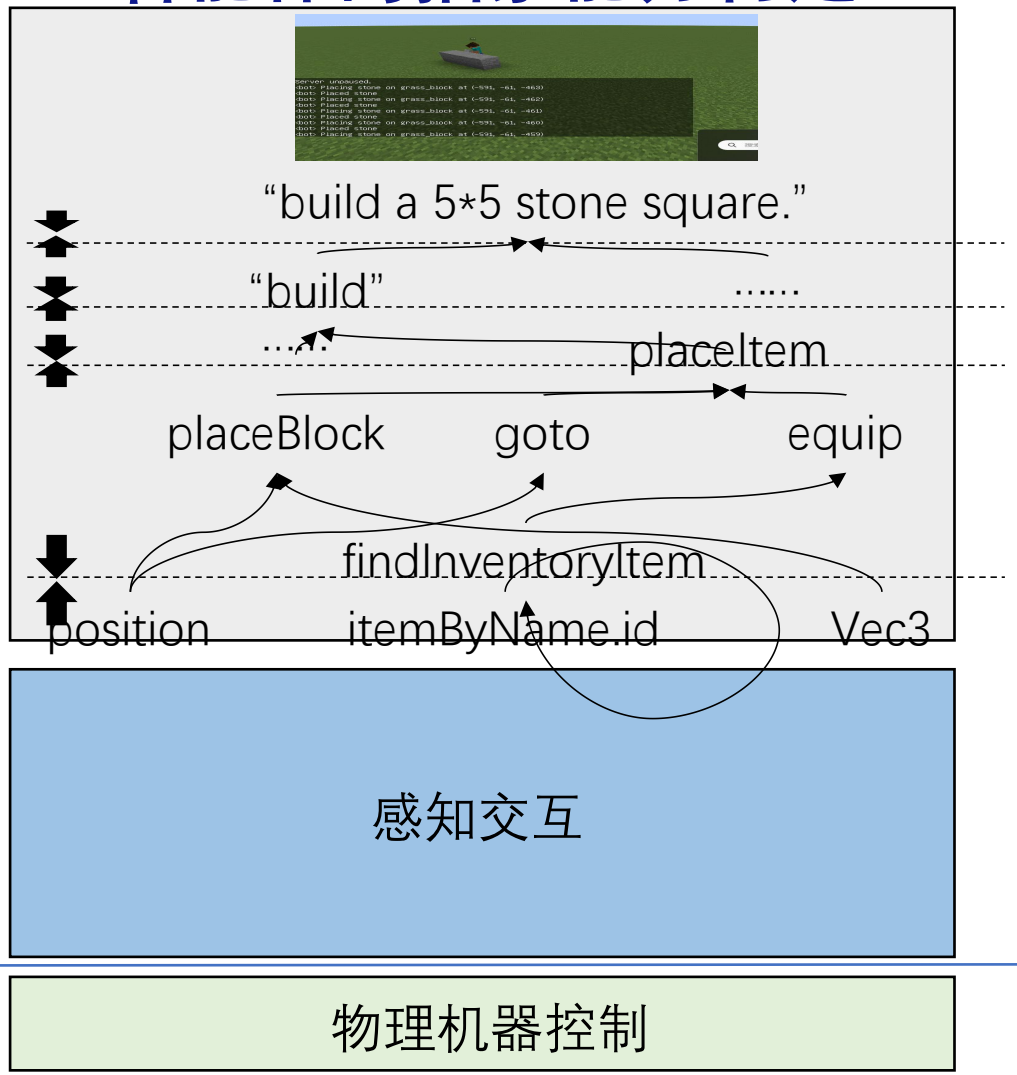
MLIR (Multi-Level Intermediate Representation)

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**



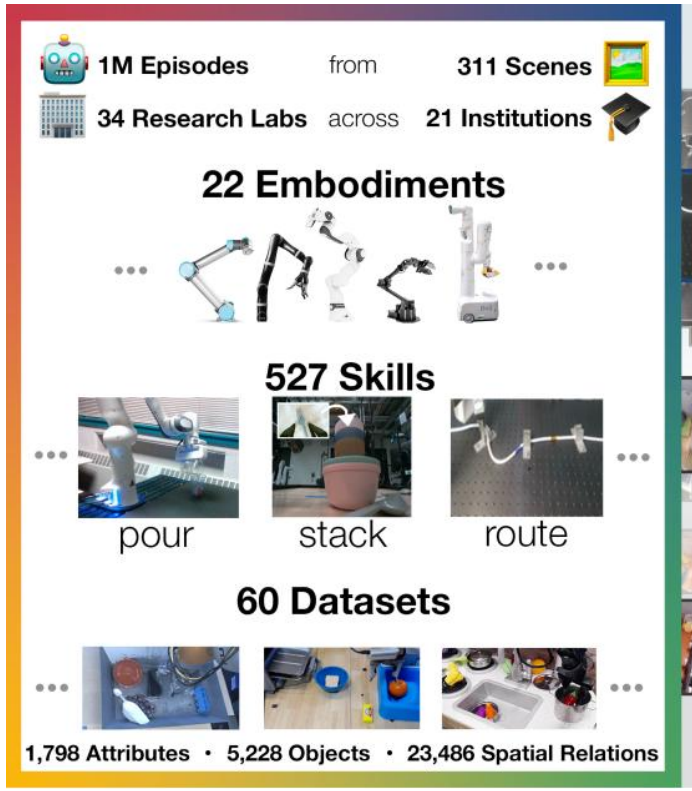
https://www.lei.chat/posts/mlir-codegen-dialects-for-machine-learning-compilers/

智能体的抽象能力构建

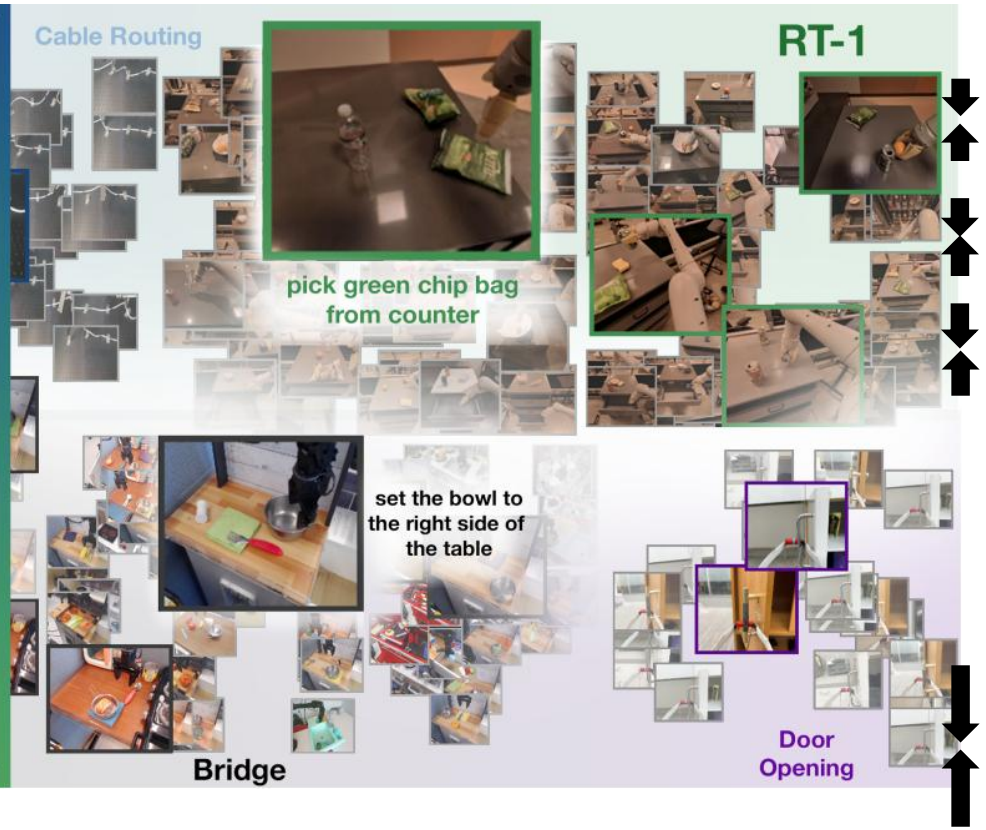


数据基础设施

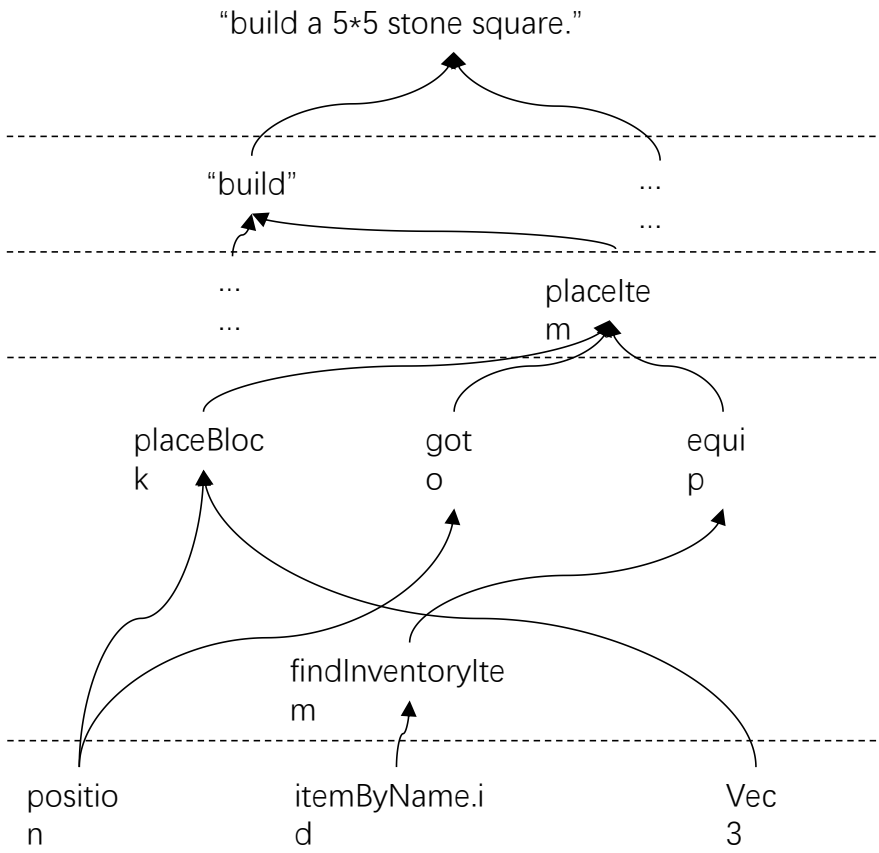
任务数据



环境数据



程序数据



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

AiDD AI+ 软件研发数字峰会
AI+ software Development Digital summit

THANKS