泛在计算与智能系统研究中心
Research Center of Ubiquitous Computing and Intelligent Systems

哈爾濱工業大學
Harbin Institute of Technology

Integrating Intent Understanding and Optimal Behavior Planning for Behavior Tree Generation from Human Instructions

Xinglin Chen∗ , Yishuai Cai ∗ , Yunxin Mao , Minglong Li † ,
Wenjing Yang , Weixia Xu , Ji Wang,

College of Computer Science and Technology, National University of Defense Technology
{chenxinglin, caiyishuai, maoyunxin, liminglong10, wenjing.yang, xuweixia, wj}@nudt.edu.cn

汇报人姓名：王雅斌

汇报日期：2024.10.15

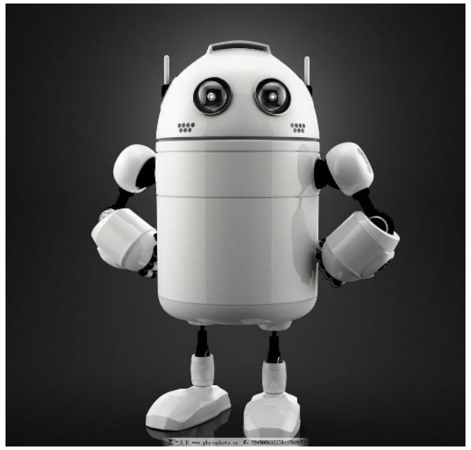
目录

研究背景

背景

行为树 → 自然语言 → 成功

机器人在家庭或工业环境中执行任务时，基本上需要具备适应性和可靠性。行为树（BT）因其模块化和反应性而成为这些场景的合适控制架构。然而，现有的BT生成方法要么不涉及自然语言的解释，要么无法理论上保证BT的成功。



动机

解决问题

低成本

低时耗

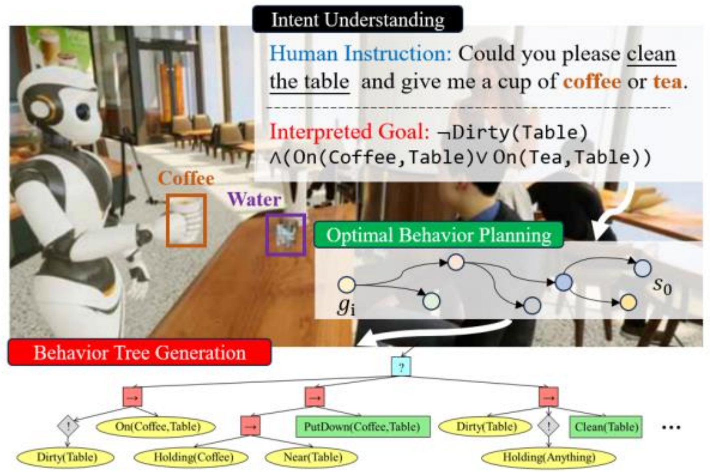
结合了LLM解决意图理解，展示OBTEA在各种指标上优于基线BT扩展算法的优势（具有更低的成本和更少的条件节点计时），并最终确认了框架的实际可部署性。

难点

泛在计算与智能系统研究中心
Research Center of Ubiquitous Computing and Intelligent Systems

技术难点

1. 由于自然语言的开放性，机器人必须理解用户的意图，明确任务的目标，并获得足够的高级技能来完成这些任务。
2. 机器人必须自动生成可靠的计划，并安全、稳健地执行这些计划。

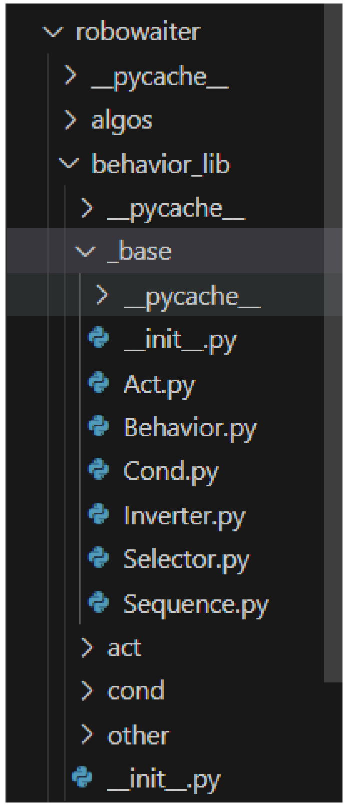


技术方法

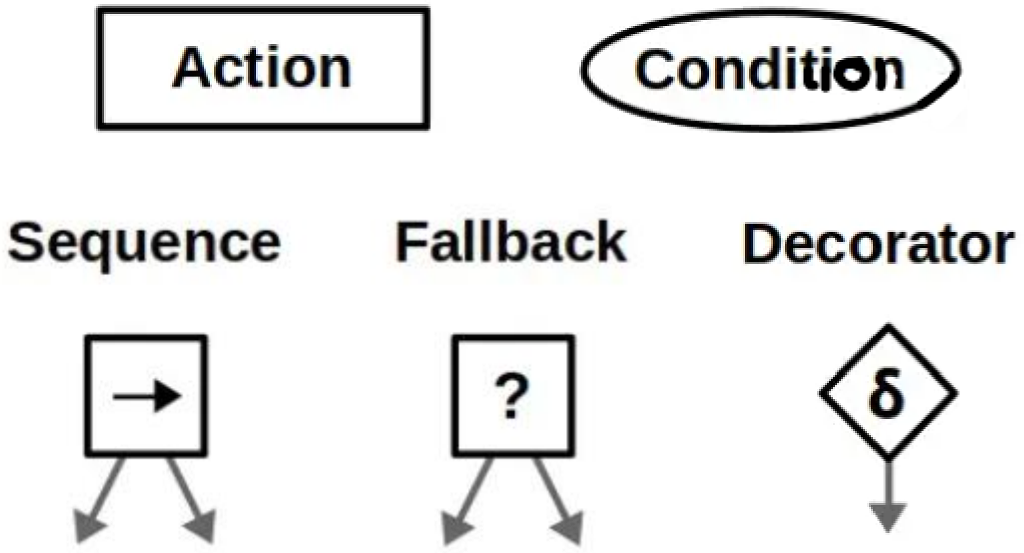
泛在计算与智能系统研究中心
Research Center of Ubiquitous Computing and Intelligent Systems

前置知识：行为树

行为树（BT）是一种有向根树结构，其中叶节点控制机器人的感知和动作，而内部节点则处理这些叶节点的逻辑结构。行为树从其根节点开始执行，根据环境状态的不同，执行过程在树中创建控制流，最终决定机器人将执行的动作。


```
robowaiter
  __pycache__
  algos
  behavior_lib
    __pycache__
    _base
      __pycache__
      __init__.py
      Act.py
      Behavior.py
      Cond.py
      Inverter.py
      Selector.py
      Sequence.py
    act
    cond
    other
    __init__.py
```

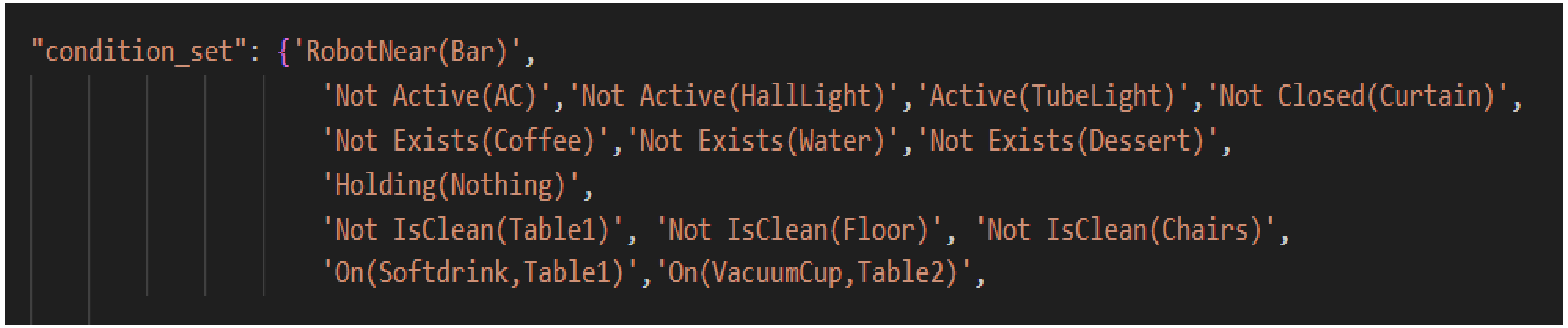
在本文中，我们关注五种典型的行为树节点：



技术方法

前置知识：环境描述

谓词+对象



```
"condition_set": {'RobotNear(Bar)',
                    'Not Active(AC)','Not Active(HallLight)','Active(TubeLight)','Not Closed(Curtain)',
                    'Not Exists(Coffee)','Not Exists(Water)','Not Exists(Dessert)',
                    'Holding(Nothing)',
                    'Not IsClean(Table1)', 'Not IsClean(Floor)', 'Not IsClean(Chairs)',
                    'On(Softdrink,Table1)','On(VacuumCup,Table2)',
```

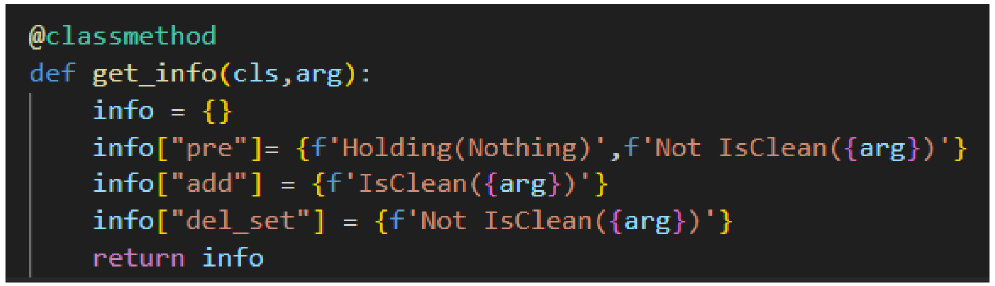
三元组 ⟨O, Pc, Pa⟩

每个谓词的对象是O的子集，环境描述可以通过pc(o1, ..., ok)以∧（合取）、∨（析取）和￢（否定）的形式组合表示。

技术方法

泛在计算与智能系统研究中心
Research Center of Ubiquitous Computing and Intelligent Systems

前置知识：行为规划



```
@classmethod
def get_info(cls,arg):
    info = {}
    info["pre"]= {f'Holding(Nothing)',f'Not IsClean({arg})'}
    info["add"] = {f'IsClean({arg})'}
    info["del_set"] = {f'Not IsClean({arg})'}
    return info
```

行为规划问题描述为一个元组：

< S, A, P, s0, g >

定义为三元组

P(a) =< pre(a), add(a), del(a) >

状态转移

s ′ = s ∪ add(a) \ del(a)

$$D(T) = \sum_{i=1}^{n} D(a_i).$$

描述

转移

目标

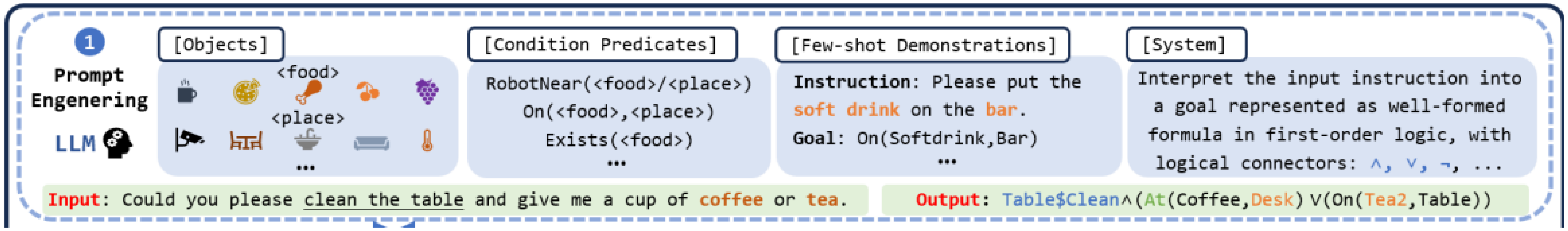
行为规划

技术方法

LLM：提示工程

四个组成部分构成：

（1）对象。
（2）条件谓词。
（3）少量示例演示。
（4）系统。



RobotNear(<food>/<place>)
On(<food>,<place>)
Exists(<food>)
...

Instruction: Please put the soft drink on the bar.
Goal: On(Softdrink,Bar)
...

Interpret the input instruction into a goal represented as well-formed formula in first-order logic, with logical connectors: ∧, ∨, ¬, ...

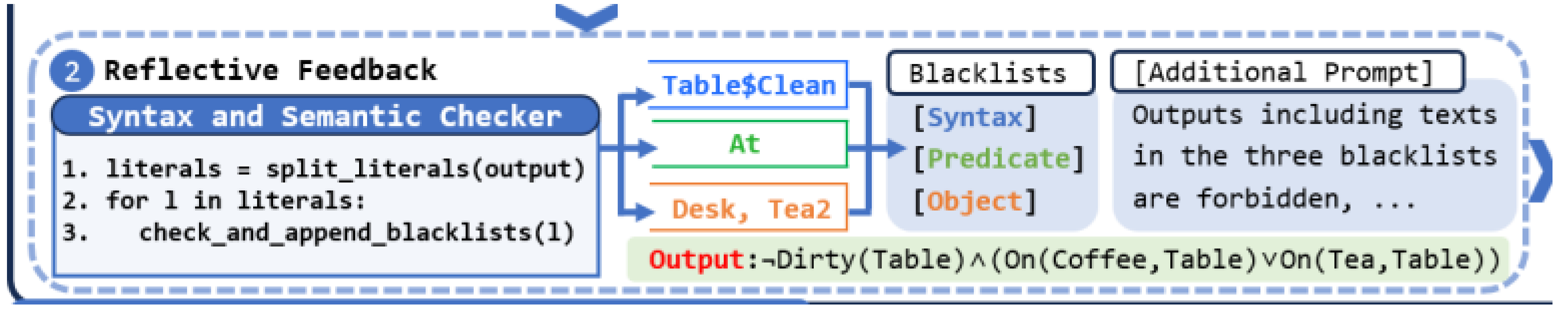
Input: Could you please clean the table and give me a cup of coffee or tea.

Output: Table$Clean∧(At(Coffee,Desk)∨(On(Tea2,Table))

技术方法

泛在计算与智能系统研究中心
Research Center of Ubiquitous Computing and Intelligent Systems

LLM：反思性反馈


2 Reflective Feedback
Syntax and Semantic Checker
Table$Clean
At
Desk, Tea2
Blacklists
[Syntax]
[Predicate]
[Object]
[Additional Prompt]
Outputs including texts in the three blacklists are forbidden, ...
Output:¬Dirty(Table)∧(On(Coffee,Table)∨On(Tea,Table))

```
1. literals = split_literals(output)
2. for l in literals:
3.    check_and_append_blacklists(l)
```

语法检查器评估输出是否构成良好公式

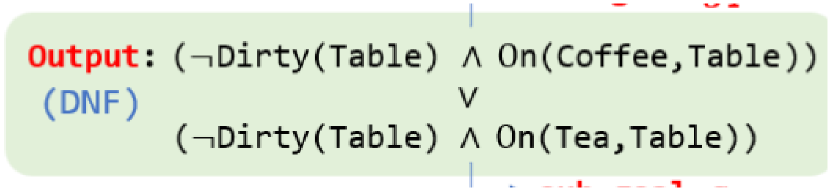
语义检查器则进一步验证谓词和对象的有效性。

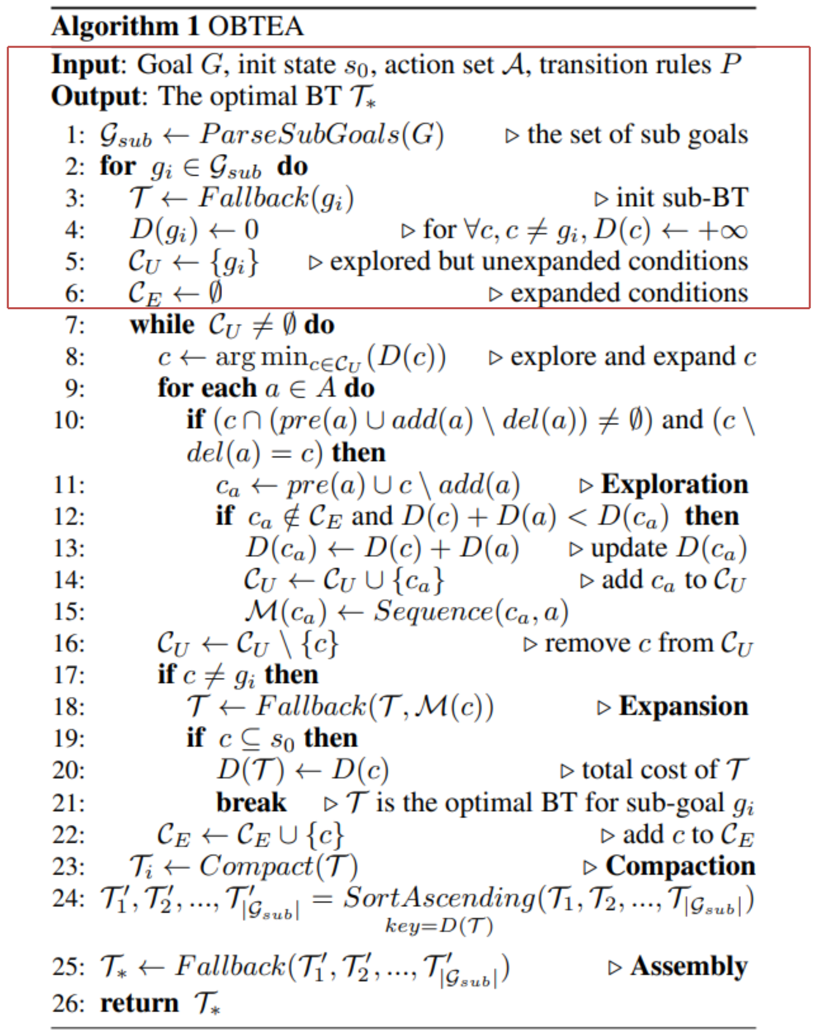
增加了黑名单和额外的上下文，以指示LLM不要在输出中再次包含黑名单中的文本。

技术方法

泛在计算与智能系统研究中心
Research Center of Ubiquitous Computing and Intelligent Systems

OBTEA


```
Output: (¬Dirty(Table) ∧ On(Coffee,Table))
(DNF)                      ∨
        (¬Dirty(Table) ∧ On(Tea,Table))
```


Algorithm 1 OBTEA

Input: Goal G, init state s_0, action set $\mathcal{A}$, transition rules P
Output: The optimal BT $\mathcal{T}_*$

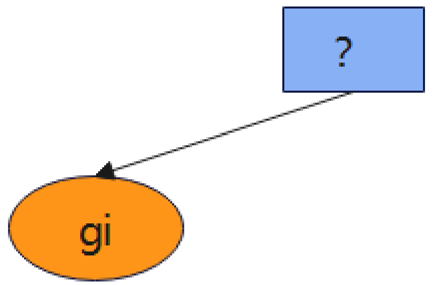
```
1:  $\mathcal{G}_{sub} \leftarrow ParseSubGoals(G)$    ▷ the set of sub goals
2:  for $g_i \in \mathcal{G}_{sub}$ do
3:      $\mathcal{T} \leftarrow Fallback(g_i)$    ▷ init sub-BT
4:      $D(g_i) \leftarrow 0$    ▷ for $\forall c, c \neq g_i, D(c) \leftarrow +\infty$
5:      $\mathcal{C}_U \leftarrow \{g_i\}$    ▷ explored but unexpanded conditions
6:      $\mathcal{C}_E \leftarrow \emptyset$    ▷ expanded conditions
7:      while $\mathcal{C}_U \neq \emptyset$ do
8:          $c \leftarrow \arg\min_{c \in \mathcal{C}_U}(D(c))$    ▷ explore and expand $c$
9:          for each $a \in \mathcal{A}$ do
10:             if $(c \cap (pre(a) \cup add(a) \setminus del(a)) \neq \emptyset)$ and $(c \setminus del(a) = c)$ then
11:                 $c_a \leftarrow pre(a) \cup c \setminus add(a)$    ▷ Exploration
12:                 if $c_a \notin \mathcal{C}_E$ and $D(c) + D(a) < D(c_a)$ then
13:                     $D(c_a) \leftarrow D(c) + D(a)$    ▷ update $D(c_a)$
14:                     $\mathcal{C}_U \leftarrow \mathcal{C}_U \cup \{c_a\}$    ▷ add $c_a$ to $\mathcal{C}_U$
15:                     $\mathcal{M}(c_a) \leftarrow Sequence(c_a, a)$
16:         $\mathcal{C}_U \leftarrow \mathcal{C}_U \setminus \{c\}$    ▷ remove $c$ from $\mathcal{C}_U$
17:         if $c \neq g_i$ then
18:             $\mathcal{T} \leftarrow Fallback(\mathcal{T}, \mathcal{M}(c))$    ▷ Expansion
19:             if $c \subseteq s_0$ then
20:                 $D(\mathcal{T}) \leftarrow D(c)$    ▷ total cost of $\mathcal{T}$
21:                 break    ▷ $\mathcal{T}$ is the optimal BT for sub-goal $g_i$
22:         $\mathcal{C}_E \leftarrow \mathcal{C}_E \cup \{c\}$    ▷ add $c$ to $\mathcal{C}_E$
23:     $\mathcal{T}_i \leftarrow Compaction(\mathcal{T})$    ▷ Compaction
24: $\mathcal{T}'_1, \mathcal{T}'_2, ..., \mathcal{T}'_{|\mathcal{G}_{sub}|} = \underset{key=D(\mathcal{T})}{SortAscending}(\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_{|\mathcal{G}_{sub}|})$
25: $\mathcal{T}_* \leftarrow Fallback(\mathcal{T}'_1, \mathcal{T}'_2, ..., \mathcal{T}'_{|\mathcal{G}_{sub}|})$    ▷ Assembly
26: return $\mathcal{T}_*$
```

输入：LLM最后的输出G，初始状态，动作谓词，状态转移规则

步骤1：初始化

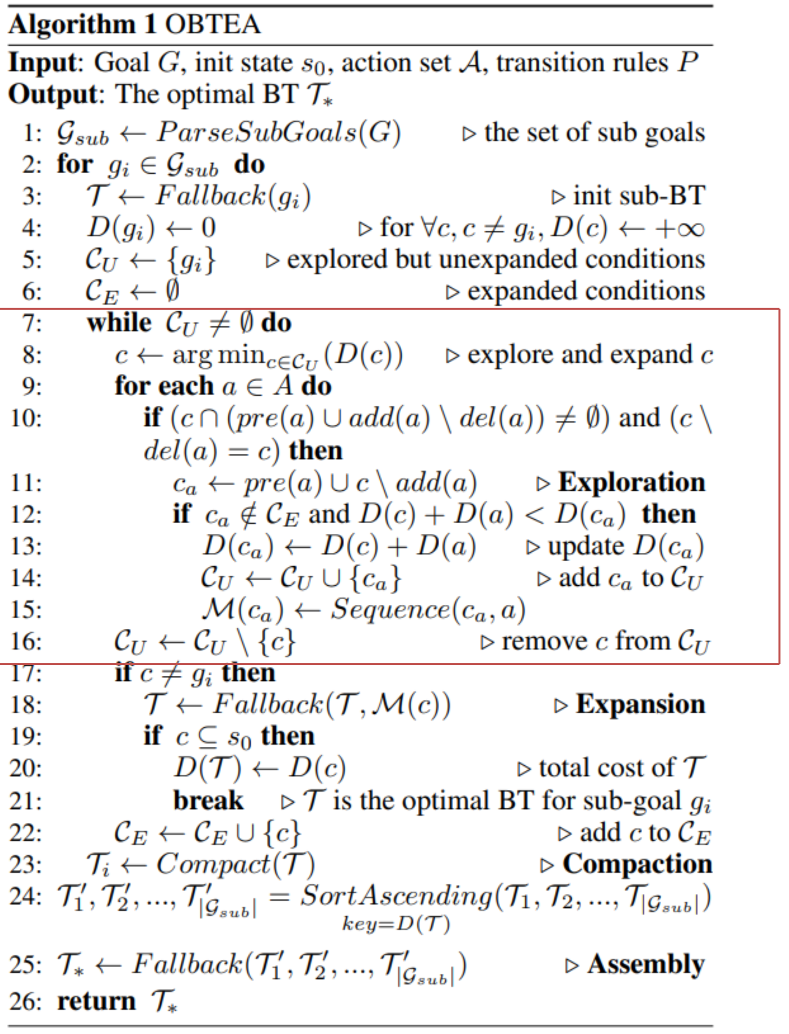
目标：G=(l1∧l2)∨(l1∧l3)
Gsub={g1，g2}
g1={l1，l2}，g2={l1，l3}

T，D，CU，CE



技术方法

OBTEA



Algorithm 1 OBTEA

Input: Goal G, init state s_0, action set $\mathcal{A}$, transition rules P
Output: The optimal BT $\mathcal{T}_*$

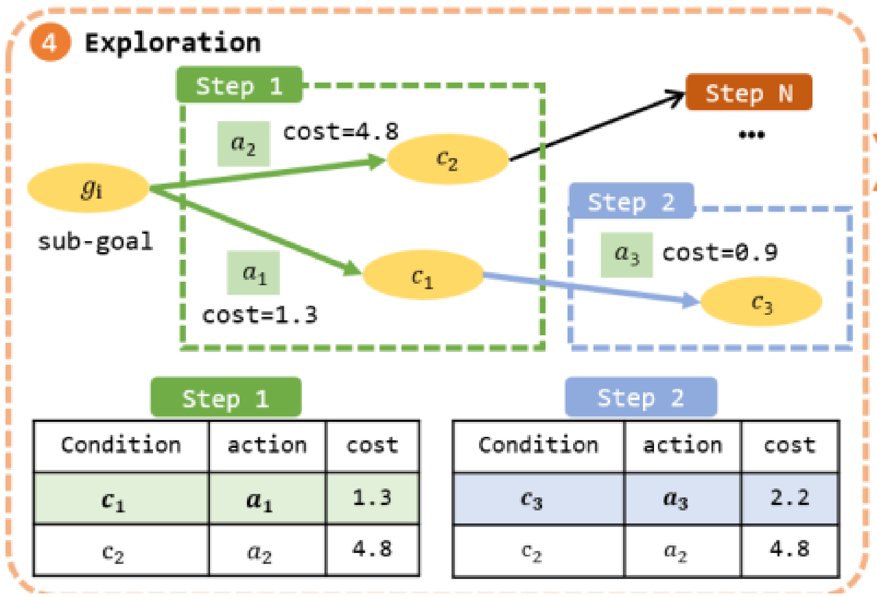
```
1:  $\mathcal{G}_{sub} \leftarrow ParseSubGoals(G)$    ▷ the set of sub goals
2:  for $g_i \in \mathcal{G}_{sub}$ do
3:      $\mathcal{T} \leftarrow Fallback(g_i)$    ▷ init sub-BT
4:      $D(g_i) \leftarrow 0$    ▷ for $\forall c, c \neq g_i, D(c) \leftarrow +\infty$
5:      $\mathcal{C}_U \leftarrow \{g_i\}$    ▷ explored but unexpanded conditions
6:      $\mathcal{C}_E \leftarrow \emptyset$    ▷ expanded conditions
7:      while $\mathcal{C}_U \neq \emptyset$ do
8:          $c \leftarrow \arg\min_{c \in \mathcal{C}_U}(D(c))$    ▷ explore and expand $c$
9:          for each $a \in \mathcal{A}$ do
10:             if $(c \cap (pre(a) \cup add(a) \setminus del(a)) \neq \emptyset)$ and $(c \setminus del(a) = c)$ then
11:                 $c_a \leftarrow pre(a) \cup c \setminus add(a)$    ▷ Exploration
12:                 if $c_a \notin \mathcal{C}_E$ and $D(c) + D(a) < D(c_a)$ then
13:                     $D(c_a) \leftarrow D(c) + D(a)$    ▷ update $D(c_a)$
14:                     $\mathcal{C}_U \leftarrow \mathcal{C}_U \cup \{c_a\}$    ▷ add $c_a$ to $\mathcal{C}_U$
15:                     $\mathcal{M}(c_a) \leftarrow Sequence(c_a, a)$
16:         $\mathcal{C}_U \leftarrow \mathcal{C}_U \setminus \{c\}$    ▷ remove $c$ from $\mathcal{C}_U$
17:         if $c \neq g_i$ then
18:             $\mathcal{T} \leftarrow Fallback(\mathcal{T}, \mathcal{M}(c))$    ▷ Expansion
19:             if $c \subseteq s_0$ then
20:                 $D(\mathcal{T}) \leftarrow D(c)$    ▷ total cost of $\mathcal{T}$
21:                 break    ▷ $\mathcal{T}$ is the optimal BT for sub-goal $g_i$
22:         $\mathcal{C}_E \leftarrow \mathcal{C}_E \cup \{c\}$    ▷ add $c$ to $\mathcal{C}_E$
23:     $\mathcal{T}_i \leftarrow Compact(\mathcal{T})$    ▷ Compaction
24: $\mathcal{T}'_1, \mathcal{T}'_2, ..., \mathcal{T}'_{|\mathcal{G}_{sub}|} = \underset{key=D(\mathcal{T})}{SortAscending}(\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_{|\mathcal{G}_{sub}|})$
25: $\mathcal{T}_* \leftarrow Fallback(\mathcal{T}'_1, \mathcal{T}'_2, ..., \mathcal{T}'_{|\mathcal{G}_{sub}|})$    ▷ Assembly
26: return $\mathcal{T}_*$
```

步骤2：探索

CU非空

邻近条件的生成Ca，忽略条件
1>ca ∈ CE，
2>D(c) + D(a) ≥ D(ca)

否则更新 D(ca)，CU



Step 1

Condition	action	cost
c_1	a_1	1.3
c_2	a_2	4.8

Step 2

Condition	action	cost
c_3	a_3	2.2
c_2	a_2	4.8

技术方法

OBTEA

Algorithm 1 OBTEA

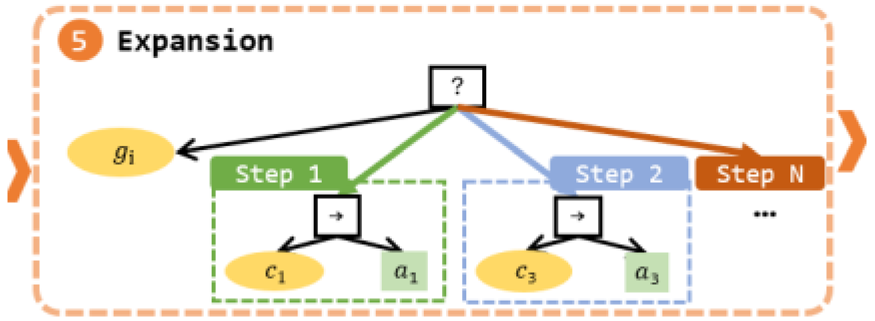
```
**Input**: Goal $G$, init state $s_0$, action set $\mathcal{A}$, transition rules $P$
**Output**: The optimal BT $\mathcal{T}_*$
 1: $\mathcal{G}_{sub} \leftarrow ParseSubGoals(G)$                ▷ the set of sub goals
 2: for $g_i \in \mathcal{G}_{sub}$ do
 3:   $\mathcal{T} \leftarrow Fallback(g_i)$                              ▷ init sub-BT
 4:   $D(g_i) \leftarrow 0$                  ▷ for $\forall c, c \neq g_i, D(c) \leftarrow +\infty$
 5:   $\mathcal{C}_U \leftarrow \{g_i\}$            ▷ explored but unexpanded conditions
 6:   $\mathcal{C}_E \leftarrow \emptyset$                          ▷ expanded conditions
 7:   while $\mathcal{C}_U \neq \emptyset$ do
 8:     $c \leftarrow \arg\min_{c \in \mathcal{C}_U}(D(c))$          ▷ explore and expand $c$
 9:     for each $a \in \mathcal{A}$ do
10:       if $(c \cap (pre(a) \cup add(a) \setminus del(a)) \neq \emptyset)$ and $(c \setminus del(a) = c)$ then
11:         $c_a \leftarrow pre(a) \cup c \setminus add(a)$              ▷ **Exploration**
12:         if $c_a \notin \mathcal{C}_E$ and $D(c) + D(a) < D(c_a)$ then
13:           $D(c_a) \leftarrow D(c) + D(a)$              ▷ update $D(c_a)$
14:           $\mathcal{C}_U \leftarrow \mathcal{C}_U \cup \{c_a\}$                ▷ add $c_a$ to $\mathcal{C}_U$
15:           $\mathcal{M}(c_a) \leftarrow Sequence(c_a, a)$
16:     $\mathcal{C}_U \leftarrow \mathcal{C}_U \setminus \{c\}$                      ▷ remove $c$ from $\mathcal{C}_U$
17:     if $c \neq g_i$ then
18:       $\mathcal{T} \leftarrow Fallback(\mathcal{T}, \mathcal{M}(c))$               ▷ **Expansion**
19:       if $c \subseteq s_0$ then
20:         $D(\mathcal{T}) \leftarrow D(c)$                          ▷ total cost of $\mathcal{T}$
21:         break     ▷ $\mathcal{T}$ is the optimal BT for sub-goal $g_i$
22:     $\mathcal{C}_E \leftarrow \mathcal{C}_E \cup \{c\}$                          ▷ add $c$ to $\mathcal{C}_E$
23:   $\mathcal{T}_i \leftarrow Compact(\mathcal{T})$                           ▷ **Compaction**
24: $\mathcal{T}'_1, \mathcal{T}'_2, ..., \mathcal{T}'_{|\mathcal{G}_{sub}|} = \underset{key=D(\mathcal{T})}{SortAscending}(\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_{|\mathcal{G}_{sub}|})$
25: $\mathcal{T}_* \leftarrow Fallback(\mathcal{T}'_1, \mathcal{T}'_2, ..., \mathcal{T}'_{|\mathcal{G}_{sub}|})$                ▷ **Assembly**
26: return $\mathcal{T}_*$
```

步骤3：拓展

添加子节点M(c)

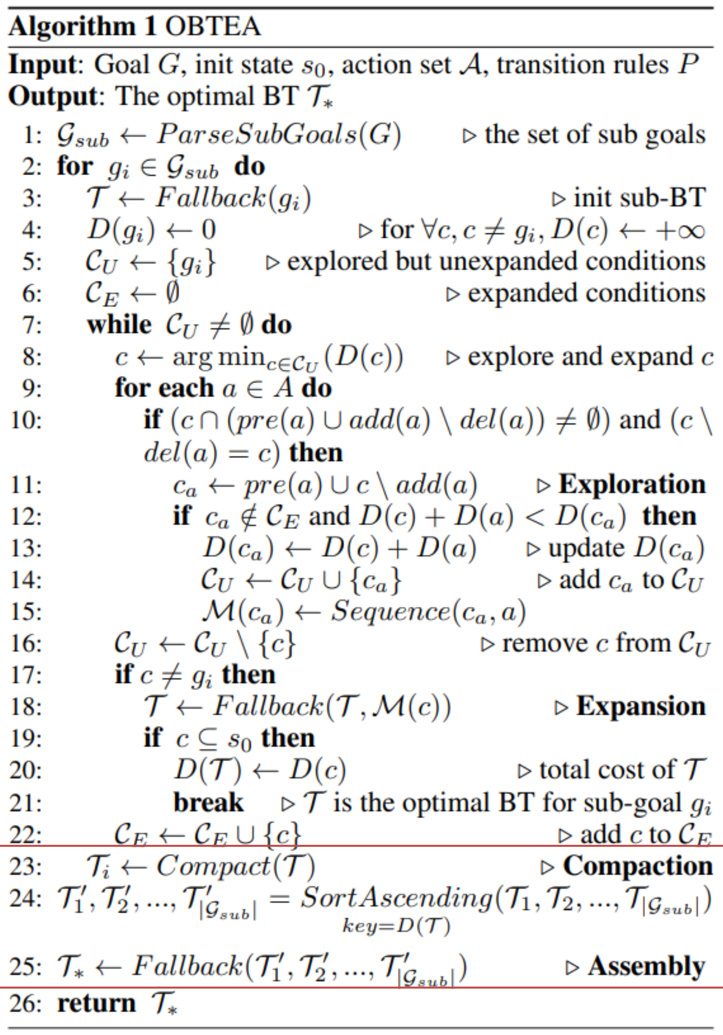
如果未达到 s0，我们将 c 添加到扩展集合 CE 中

继续探索条件空间，直到扩展到s0



技术方法

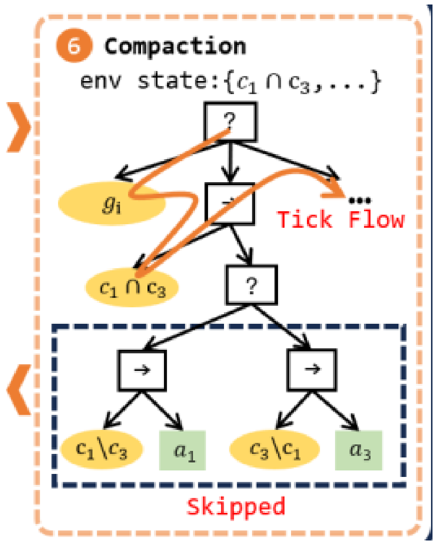
OBTEA



```
Algorithm 1 OBTEA
Input: Goal $G$, init state $s_0$, action set $\mathcal{A}$, transition rules $P$
Output: The optimal BT $\mathcal{T}_*$
 1: $\mathcal{G}_{sub} \leftarrow ParseSubGoals(G)$                ▷ the set of sub goals
 2: for $g_i \in \mathcal{G}_{sub}$ do
 3:     $\mathcal{T} \leftarrow Fallback(g_i)$                    ▷ init sub-BT
 4:     $D(g_i) \leftarrow 0$                  ▷ for $\forall c, c \neq g_i, D(c) \leftarrow +\infty$
 5:     $\mathcal{C}_U \leftarrow \{g_i\}$        ▷ explored but unexpanded conditions
 6:     $\mathcal{C}_E \leftarrow \emptyset$                       ▷ expanded conditions
 7:     while $\mathcal{C}_U \neq \emptyset$ do
 8:         $c \leftarrow \arg\min_{c \in \mathcal{C}_U}(D(c))$       ▷ explore and expand $c$
 9:         for each $a \in \mathcal{A}$ do
10:             if $(c \cap (pre(a) \cup add(a) \setminus del(a)) \neq \emptyset)$ and $(c \setminus del(a) = c)$ then
11:                 $c_a \leftarrow pre(a) \cup c \setminus add(a)$          ▷ Exploration
12:                 if $c_a \notin \mathcal{C}_E$ and $D(c) + D(a) < D(c_a)$ then
13:                     $D(c_a) \leftarrow D(c) + D(a)$           ▷ update $D(c_a)$
14:                     $\mathcal{C}_U \leftarrow \mathcal{C}_U \cup \{c_a\}$            ▷ add $c_a$ to $\mathcal{C}_U$
15:                     $\mathcal{M}(c_a) \leftarrow Sequence(c_a, a)$
16:         $\mathcal{C}_U \leftarrow \mathcal{C}_U \setminus \{c\}$                ▷ remove $c$ from $\mathcal{C}_U$
17:         if $c \neq g_i$ then
18:             $\mathcal{T} \leftarrow Fallback(\mathcal{T}, \mathcal{M}(c))$           ▷ Expansion
19:             if $c \subseteq s_0$ then
20:                 $D(\mathcal{T}) \leftarrow D(c)$                  ▷ total cost of $\mathcal{T}$
21:                 break      ▷ $\mathcal{T}$ is the optimal BT for sub-goal $g_i$
22:         $\mathcal{C}_E \leftarrow \mathcal{C}_E \cup \{c\}$                  ▷ add $c$ to $\mathcal{C}_E$
23:     $\mathcal{T}_i \leftarrow Compaction(\mathcal{T})$                ▷ Compaction
24: $\mathcal{T}'_1, \mathcal{T}'_2, ..., \mathcal{T}'_{|\mathcal{G}_{sub}|} = \underset{key=D(\mathcal{T})}{SortAscending}(\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_{|\mathcal{G}_{sub}|})$
25: $\mathcal{T}_* \leftarrow Fallback(\mathcal{T}'_1, \mathcal{T}'_2, ..., \mathcal{T}'_{|\mathcal{G}_{sub}|})$          ▷ Assembly
26: return $\mathcal{T}_*$
```

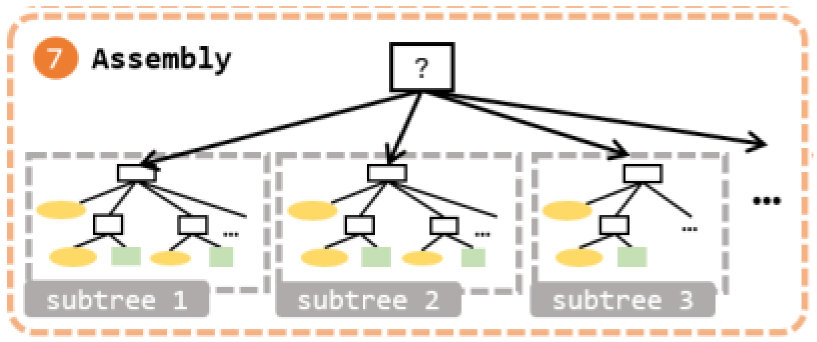
步骤4：压缩

压缩树结构并提高运行时效率。



步骤5：组装

所有子树生成完毕，我们首先按照总成本的升序对所有子树进行排序



效果

部署场景

酒店行业场景中部署了该框架，其中机器人充当咖啡馆的服务员

80 种可用物体

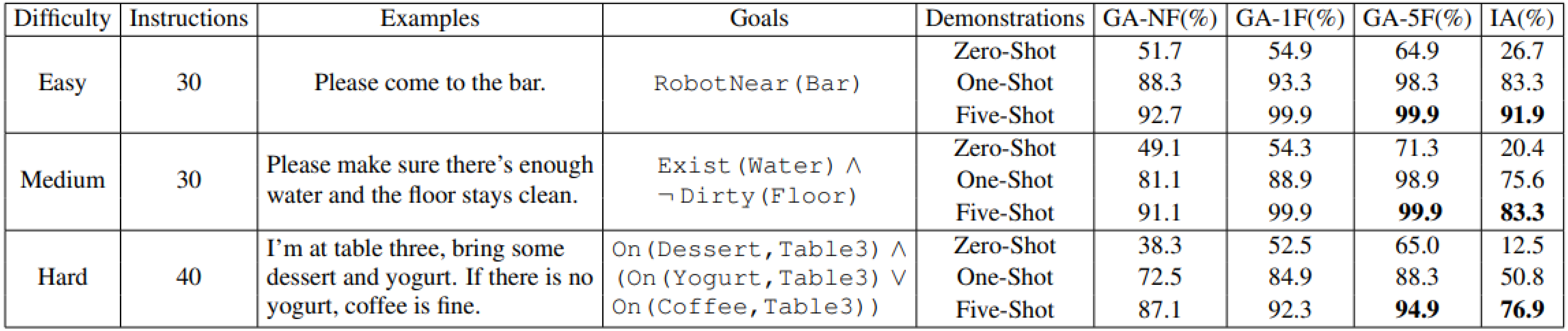
8 个条件谓词：Pc = {RobotNear, Holding, On, Closed, Exists, Dirty, Activate, Low}

6 个动作谓词 Pa = {Make, MoveTo, PickUp, PutDown, Clean, Turn}

效果

泛在计算与智能系统研究中心
Research Center of Ubiquitous Computing and Intelligent Systems

LLM



Difficulty	Instructions	Examples	Goals	Demonstrations	GA-NF(%)	GA-1F(%)	GA-5F(%)	IA(%)
Easy	30	Please come to the bar.	`RobotNear(Bar)`	Zero-Shot	51.7	54.9	64.9	26.7
				One-Shot	88.3	93.3	98.3	83.3
				Five-Shot	92.7	99.9	**99.9**	**91.9**
Medium	30	Please make sure there's enough water and the floor stays clean.	`Exist(Water)` ∧ ¬ `Dirty(Floor)`	Zero-Shot	49.1	54.3	71.3	20.4
				One-Shot	81.1	88.9	98.9	75.6
				Five-Shot	91.1	99.9	**99.9**	**83.3**
Hard	40	I'm at table three, bring some dessert and yogurt. If there is no yogurt, coffee is fine.	`On(Dessert,Table3)` ∧ `(On(Yogurt,Table3)` ∨ `On(Coffee,Table3))`	Zero-Shot	38.3	52.5	65.0	12.5
				One-Shot	72.5	84.9	88.3	50.8
				Five-Shot	87.1	92.3	**94.9**	**76.9**

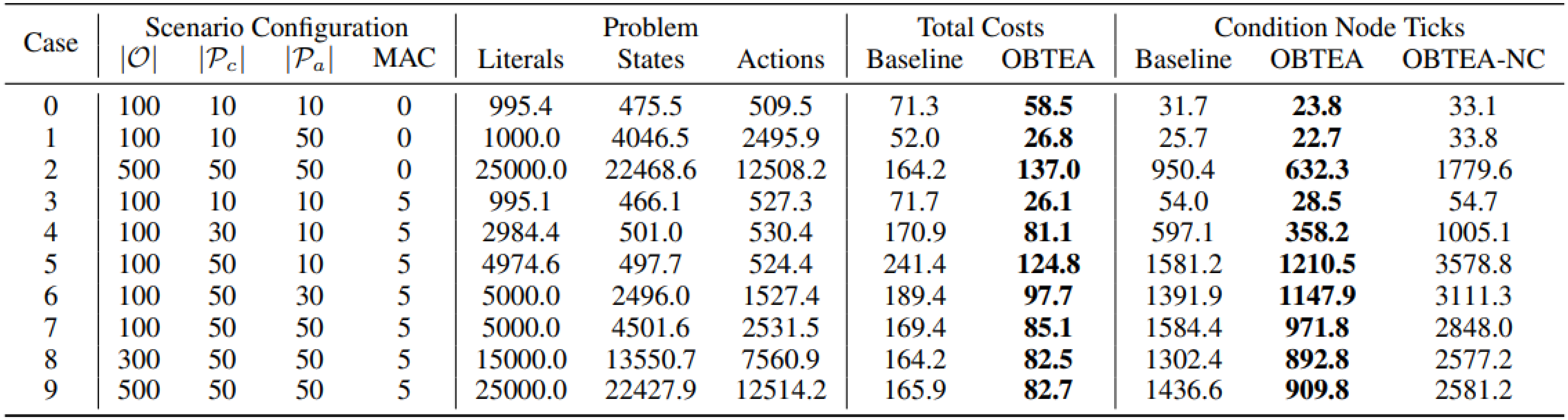
评估数据集：三个难度级别进行了分类。

两个指标进行评估：语法准确性（GA）和解释准确性（IA）。

研究了两个因素对我们方法的影响：示例的数量和最大重试次数。

效果

OBTEA



Case	Scenario Configuration				Problem			Total Costs		Condition Node Ticks		
	$\lvert\mathcal{O}\rvert$	$\lvert\mathcal{P}_c\rvert$	$\lvert\mathcal{P}_a\rvert$	MAC	Literals	States	Actions	Baseline	OBTEA	Baseline	OBTEA	OBTEA-NC
0	100	10	10	0	995.4	475.5	509.5	71.3	**58.5**	31.7	**23.8**	33.1
1	100	10	50	0	1000.0	4046.5	2495.9	52.0	**26.8**	25.7	**22.7**	33.8
2	500	50	50	0	25000.0	22468.6	12508.2	164.2	**137.0**	950.4	**632.3**	1779.6
3	100	10	10	5	995.1	466.1	527.3	71.7	**26.1**	54.0	**28.5**	54.7
4	100	30	10	5	2984.4	501.0	530.4	170.9	**81.1**	597.1	**358.2**	1005.1
5	100	50	10	5	4974.6	497.7	524.4	241.4	**124.8**	1581.2	**1210.5**	3578.8
6	100	50	30	5	5000.0	2496.0	1527.4	189.4	**97.7**	1391.9	**1147.9**	3111.3
7	100	50	50	5	5000.0	4501.6	2531.5	169.4	**85.1**	1584.4	**971.8**	2848.0
8	300	50	50	5	15000.0	13550.7	7560.9	164.2	**82.5**	1302.4	**892.8**	2577.2
9	500	50	50	5	25000.0	22427.9	12514.2	165.9	**82.7**	1436.6	**909.8**	2581.2

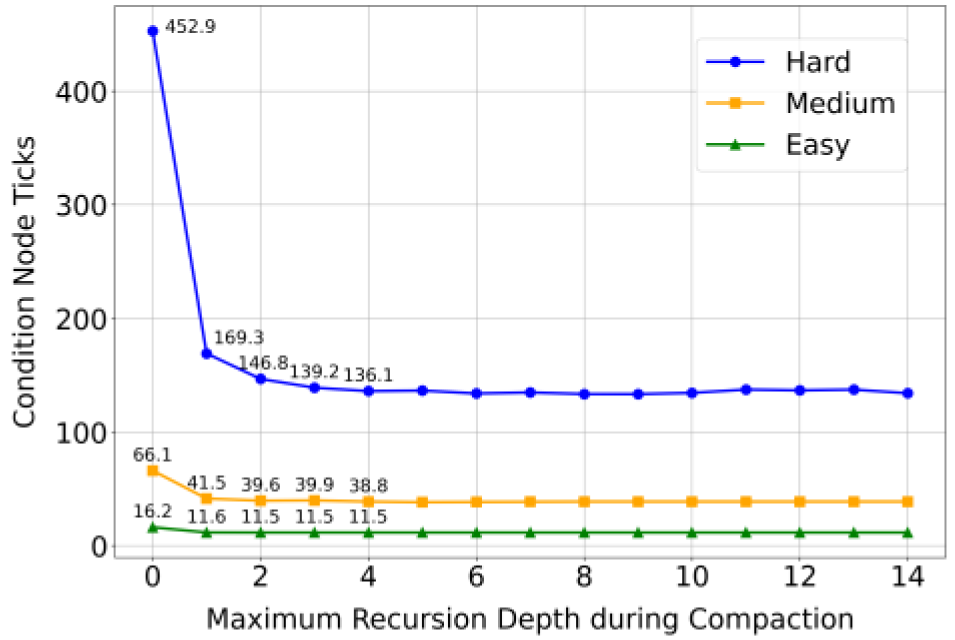
动作成本是基于人类经验设定的。

关注两个关键指标：BT的总成本和BT执行过程中条件节点的计数。

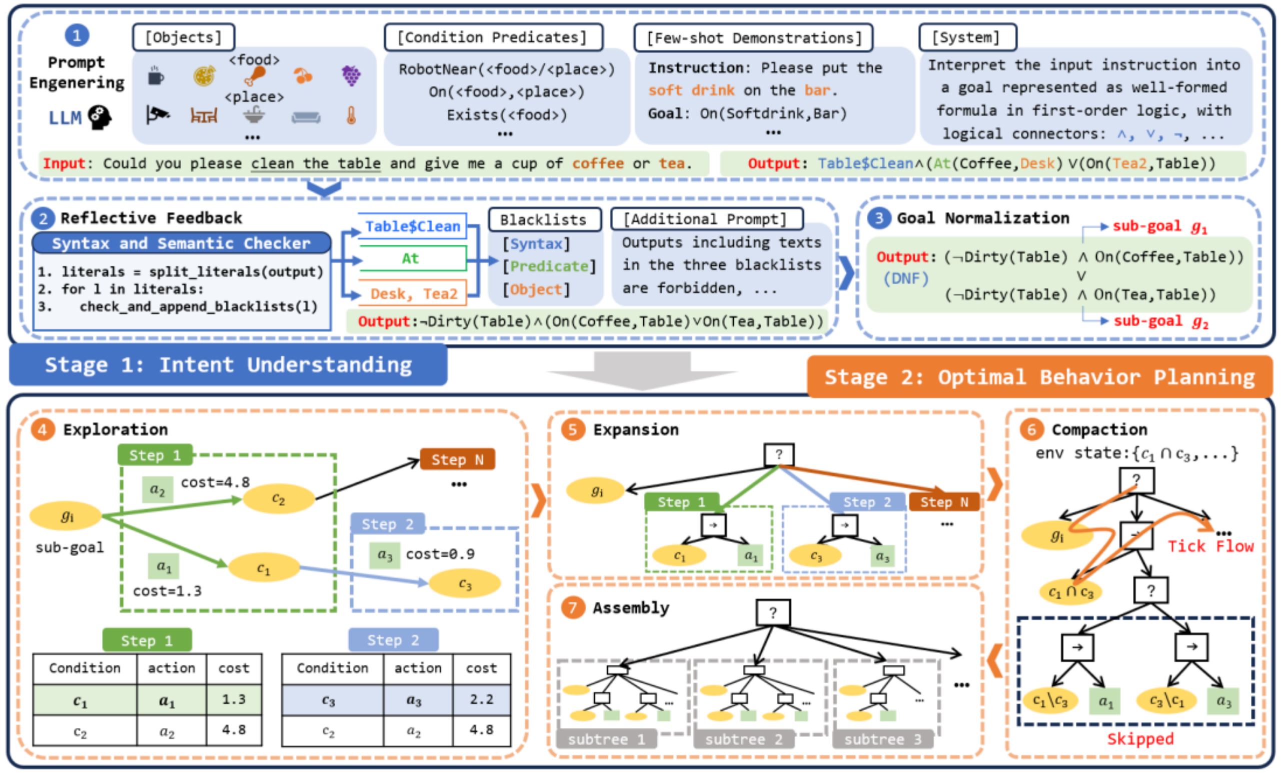
效果

OBTEA

对压缩过程中最大递归深度的影响进行了消融分析



总结


1 Prompt Engenering
LLM
[Objects]
<food>
<place>
[Condition Predicates]
RobotNear(<food>/<place>)
On(<food>,<place>)
Exists(<food>)
[Few-shot Demonstrations]
Instruction: Please put the soft drink on the bar.
Goal: On(Softdrink,Bar)
[System]
Interpret the input instruction into a goal represented as well-formed formula in first-order logic, with logical connectors: ∧, ∨, ¬, ...
Input: Could you please clean the table and give me a cup of coffee or tea.
Output: Table$Clean∧(At(Coffee,Desk)∨(On(Tea2,Table))
2 Reflective Feedback
Syntax and Semantic Checker
Table$Clean
At
Desk, Tea2
Blacklists
[Syntax]
[Predicate]
[Object]
[Additional Prompt]
Outputs including texts in the three blacklists are forbidden, ...
Output:¬Dirty(Table)∧(On(Coffee,Table)∨On(Tea,Table))
3 Goal Normalization
sub-goal g1
Output: (¬Dirty(Table) ∧ On(Coffee,Table))
(DNF) ∨
(¬Dirty(Table) ∧ On(Tea,Table))
sub-goal g2
Stage 1: Intent Understanding
Stage 2: Optimal Behavior Planning
4 Exploration
gi
sub-goal
Step 1
a2 cost=4.8 c2
a1 c1 cost=1.3
Step 2
a3 cost=0.9 c3
Step N
5 Expansion
?
gi
Step 1
Step 2
Step N
c1 a1
c3 a3
6 Compaction
env state:{c1 ∩ c3,...}
gi
Tick Flow
c1 ∩ c3
c1\c3 a1 c3\c1 a3
Skipped
7 Assembly
subtree 1
subtree 2
subtree 3

```
1. literals = split_literals(output)
2. for l in literals:
3.     check_and_append_blacklists(l)
```

Step 1

Condition	action	cost
c_1	a_1	1.3
c_2	a_2	4.8

Step 2

Condition	action	cost
c_3	a_3	2.2
c_2	a_2	4.8

在第一阶段，输入指令通过大语言模型（LLM）与提示工程和反思反馈转化为逻辑表达的目标。该目标随后被规范化为合取范式（DNF）并分解为子目标。在第二阶段，通过探索、扩展和压缩为每个子目标生成一个子树。这些子树最终被组装以创建最终的最优决策树

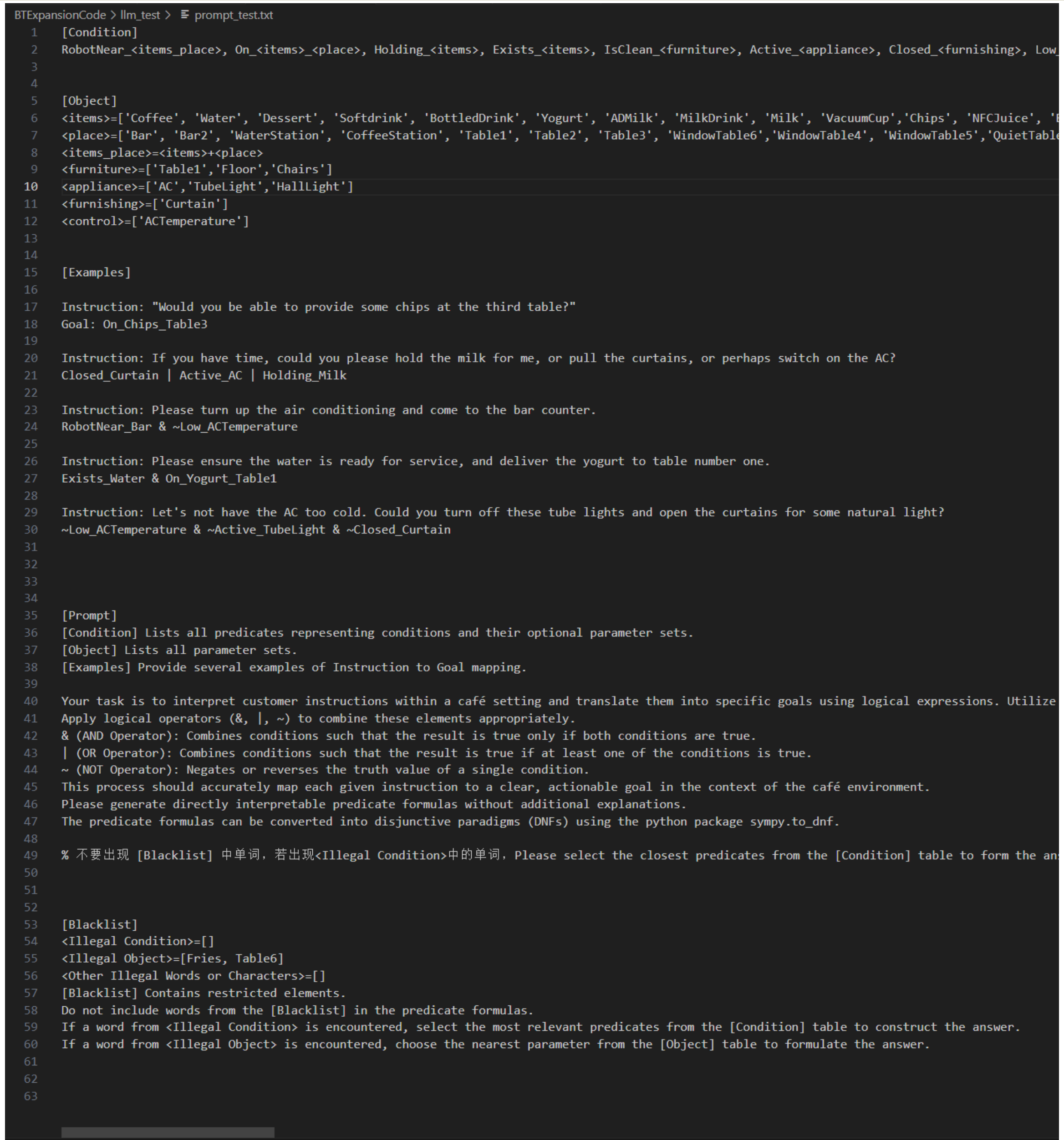
结束

问题：

泛在计算与智能系统研究中心
Research Center of Ubiquitous Computing and Intelligent Systems

1.传参时是否有把动作，环境信息传入

2.黑名单的使用



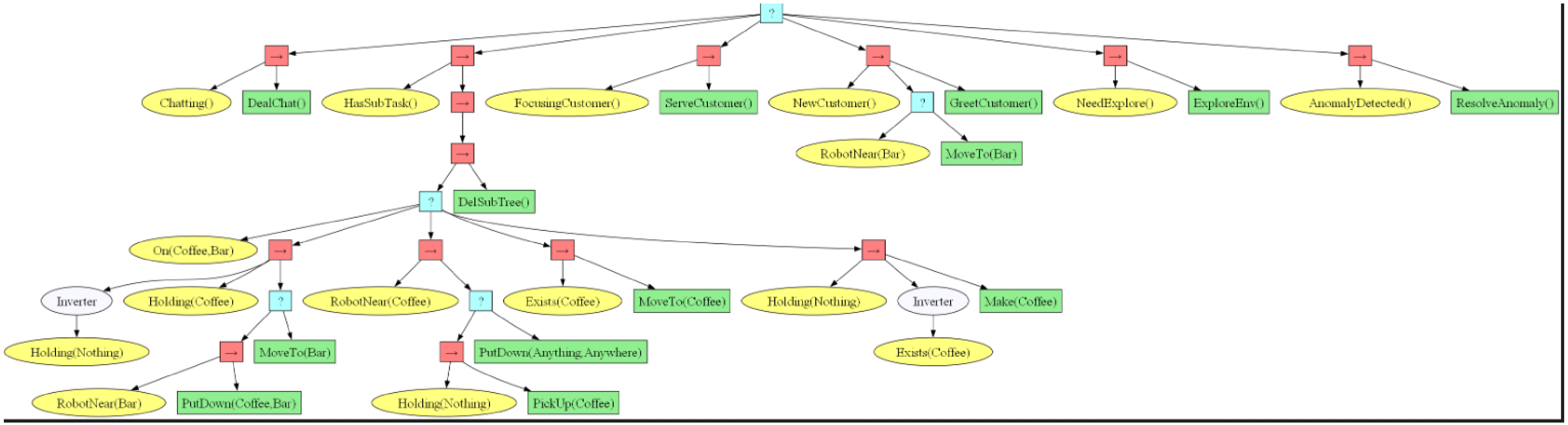
BTExpansionCode > llm_test > prompt_test.txt

```
[Condition]
RobotNear_<items_place>, On_<items>_<place>, Holding_<items>, Exists_<items>, IsClean_<furniture>, Active_<appliance>, Closed_<furnishing>, Low_


[Object]
<items>=['Coffee', 'Water', 'Dessert', 'Softdrink', 'BottledDrink', 'Yogurt', 'ADMilk', 'MilkDrink', 'Milk', 'VacuumCup','Chips', 'NFCJuice', '
<place>=['Bar', 'Bar2', 'WaterStation', 'CoffeeStation', 'Table1', 'Table2', 'Table3', 'WindowTable6','WindowTable4', 'WindowTable5','QuietTable
<items_place>=<items>+<place>
<furniture>=['Table1','Floor','Chairs']
<appliance>=['AC','TubeLight','HallLight']
<furnishing>=['Curtain']
<control>=['ACTemperature']


[Examples]

Instruction: "Would you be able to provide some chips at the third table?"
Goal: On_Chips_Table3

Instruction: If you have time, could you please hold the milk for me, or pull the curtains, or perhaps switch on the AC?
Closed_Curtain | Active_AC | Holding_Milk

Instruction: Please turn up the air conditioning and come to the bar counter.
RobotNear_Bar & ~Low_ACTemperature

Instruction: Please ensure the water is ready for service, and deliver the yogurt to table number one.
Exists_Water & On_Yogurt_Table1

Instruction: Let's not have the AC too cold. Could you turn off these tube lights and open the curtains for some natural light?
~Low_ACTemperature & ~Active_TubeLight & ~Closed_Curtain




[Prompt]
[Condition] Lists all predicates representing conditions and their optional parameter sets.
[Object] Lists all parameter sets.
[Examples] Provide several examples of Instruction to Goal mapping.

Your task is to interpret customer instructions within a café setting and translate them into specific goals using logical expressions. Utilize
Apply logical operators (&, |, ~) to combine these elements appropriately.
& (AND Operator): Combines conditions such that the result is true only if both conditions are true.
| (OR Operator): Combines conditions such that the result is true if at least one of the conditions is true.
~ (NOT Operator): Negates or reverses the truth value of a single condition.
This process should accurately map each given instruction to a clear, actionable goal in the context of the café environment.
Please generate directly interpretable predicate formulas without additional explanations.
The predicate formulas can be converted into disjunctive paradigms (DNFs) using the python package sympy.to_dnf.

% 不要出现 [Blacklist] 中单词，若出现<Illegal Condition>中的单词，Please select the closest predicates from the [Condition] table to form the an



[Blacklist]
<Illegal Condition>=[]
<Illegal Object>=[Fries, Table6]
<Other Illegal Words or Characters>=[]
[Blacklist] Contains restricted elements.
Do not include words from the [Blacklist] in the predicate formulas.
If a word from <Illegal Condition> is encountered, select the most relevant predicates from the [Condition] table to construct the answer.
If a word from <Illegal Object> is encountered, choose the nearest parameter from the [Object] table to formulate the answer.
```

问题：

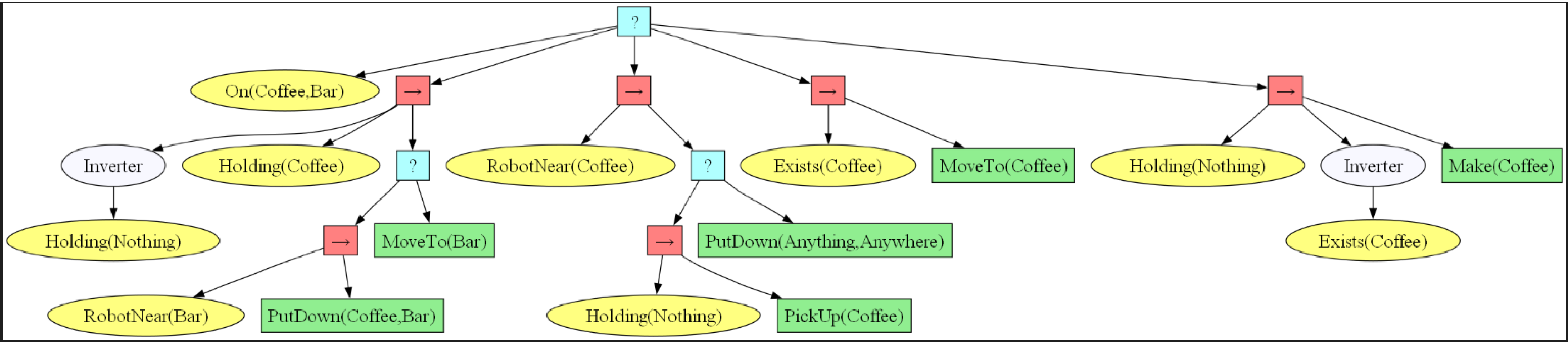
3.生成的逻辑与优先级有关吗

给我一杯咖啡



问题：

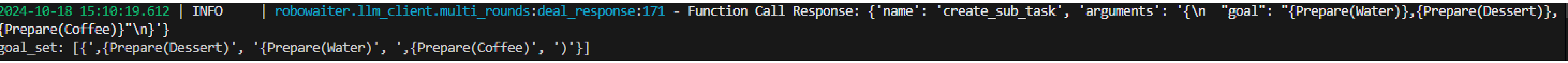
4.行为树的执行步骤



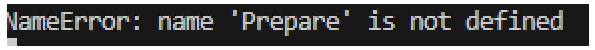
问题：

泛在计算与智能系统研究中心
Research Center of Ubiquitous Computing and Intelligent Systems

5.涉及到环境中没有的物体或者没有的动作他会怎么做（稳定性）



```
2024-10-18 15:10:19.612 | INFO     | robowaiter.llm_client.multi_rounds:deal_response:171 - Function Call Response: {'name': 'create_sub_task', 'arguments': '{\n  "goal": "{Prepare(Water)},{Prepare(Dessert)},
{Prepare(Coffee)}"\n}'}
goal_set: [{',{Prepare(Dessert)', '{Prepare(Water)', ',{Prepare(Coffee)', ')'}]
```



```
NameError: name 'Prepare' is not defined
```