SayCan 的执行循环

1. 给定用户指令 `i`
2. 枚举所有技能 π ∈ Π
    - 对每个技能：
        - LLM 打分：**这个技能语义上是不是下一步?**
        - Value Function 打分：**现在能不能成功?**
3. 相乘，选分数最高的技能
4. 执行这个技能
5. 更新环境状态
6. 把"我刚才做了什么"追加进 prompt
7. 回到第 2 步，直到输出 `done`

$$\pi^* = \arg\max_{\pi \in \Pi} \underbrace{p(c_\pi | s, \ell_\pi)}_{\text{Affordance（能做到）}} \times \underbrace{p(\ell_\pi | i)}_{\text{LLM（该做啥）}}$$

符号解释表

符号	含义	机械臂场景举例
i	用户输入的高层指令	"把那个脏的零件挪开"
Π	机器人会的所有技能集合	{抓取, 放置, 移动到A点, ...}
ℓ_π	某个技能的语言描述	"pick up the dirty part"
s	当前环境状态（如摄像头画面）	桌面上有3个零件，1个脏
$p(\ell_\pi \| i)$	LLM：这个技能"对不对"（语义匹配度）	LLM觉得"抓脏零件"符合指令 → 0.9
$p(c_\pi \| s, \ell_\pi)$	Affordance：机器人"能不能"做到	脏零件在臂展内 → 0.8
最终得分	两者相乘	$0.9 \times 0.8 = 0.72$ ✓ 选中

其实就是取这几个集合中的所有可行性的最大值

这个语言消歧SayCan策略，我觉得是这样理解的

1.用户给出最高指令

2.遍历机器可执行动作

3.每一个动作和最高指令的匹配评分

4.每一个动作在当前状况下的可执行性评分

5.两者相乘，求取最大值，对应的机器可执行动作，为机器的真实执行动作

6.执行动作之后S，即环境状态发生改变，需更改机器可执行动作的affordance评分

7.循环该过程

机器不是创造出来动作，而是在已经设计好的这个动作之中进行选择

5️⃣ 两者相乘，求取最大值，对应的机器可执行动作，为机器的真实执行动作

✅ 100% 正确，这是 SayCan 的决策核心

$$\pi^{*} = \arg\max_{\pi} \left(p(\ell_\pi \mid i) \times p(c_\pi \mid s, \ell_\pi) \right)$$

你这一步理解得**非常成熟**，因为你已经默认了两件事：

- ❌ 不是先语言选，再看能不能做
- ❌ 也不是先看能不能做，再想理由

好一个联合最优，好一个世界的普遍联系和永恒发展

好一个动态消歧，在执行的过程中，这个世界的状态也会不断的发生改变，也就是改变公式里面的这个S参数，世界状态，也就是这个affordance参数

```arduino
for step = 1,2,3,...
    重新遍历所有可执行动作
    重新算语言匹配
    重新算 affordance
    选一个
    执行
    更新世界状态
```

👉 **状态在变，歧义会自动"坍缩"**

举个工业级直觉例子
用户说：

> "把那个螺丝拧紧"

- 一开始：
  - 多个螺丝 affordance 都不低
- 机器人靠近某一个后：
  - 其他螺丝 affordance ↓
  - 当前螺丝 affordance ↑

👉 **不是语言理解突然变准了**
👉 **而是世界状态替你做了消歧**

好一个歧义坍缩

🔁 隐藏的第 6 步：执行 → 更新状态 → 重新评分

SayCan 不是一次性做完的。

真实流程是：

```arduino
for step = 1,2,3,...
    重新遍历所有可执行动作
    重新算语言匹配
    重新算 affordance
    选一个
    执行
    更新世界状态
```

👉 状态在变，歧义会自动"坍缩"

在这个SayCan中这个prompt指的是一种模版，我们把它翻译成"规划上下文"

工业版 Prompt 模板（建议你以后就用这个骨架）

[系统角色/规则]
Robot: 你是工业机械臂的"任务规划器"。
你只能从【技能列表】中选择下一步动作。
输出必须严格为：`1. <技能>`，不要输出多余解释。

[少量示例]（教它"模糊→先对齐/先确认/先靠近"的套路）
Human: 把这个零件装上去。
Robot: 1. go_to(assembly_station)

Human: 把那个螺丝拧紧。
Robot: 1. align_to(screw_candidate)

[当前任务 + （可选）历史]
Human: 把那个螺丝拧到那里。
Robot: 1.

然后你做 scoring：

果然，只要理解了这个prompt是事先写好写进这个模版的，我们就理解了这个规划上下文的本质

Prompt 是预先设计好写进系统的

一句话回答

对，Prompt 是研究人员事先精心设计好的"剧本模板"，在系统运行前就写死在代码里了。

为什么 SayCan 无法处理"两个脏零件"的消歧问题?

先回顾 SayCan 的工作流程

匹配技能指令，匹配可行性

SayCan策略的问题有很多，不仅仅是在这个模糊指令上面，还有的是这个skill库数据爆炸的问题

5) 技能数量扩展性：550 个技能还行，上千个怎么办？

先纠正一个数字：论文里说他们提出了 **551 个 skills**（7 个技能家族、17 个物体等）Ahn 等 - 2022 - Do As I Can, Not...；但真正用于长时规划的，是其中更稳定、更适合组合的一部分（某些技能因为不稳定或不常用被排除）。

扩展性问题的本质是：**每一步要对候选技能逐个打分 + 逐个算 affordance（value）**，复杂度大体是 $O(|\Pi|)$ 每步。

如果上千甚至上万技能，常见解法（论文没有完整展开，但你做工业场景很实用）：

- **两阶段检索**：先用便宜的 embedding 相似度/规则把候选缩到 top-k，再让 LLM 精排（k=20/50 这种量级）
- **层级技能库**：先选大类（导航/抓取/放置/装配），再在类内选具体技能
- **参数化技能**：减少技能离散枚举数量（把"对象/位置/姿态"变成参数槽）

你做"模糊指令语义消歧"时，这个两阶段尤其重要：先用视觉/检索把"可能的对象/动作集合"缩小，再用 LLM 做最终消歧决策。

1) 一句话定义（最常用，最稳）

> **SayCan（Ahn et al., 2022）是一种技能级任务规划框架/方法**：在给定高层自然语言指令 i 和技能集合 Π 时，它用语言模型给每个技能文本 ℓ_π 打"该不该做"的分数 $p(\ell_\pi \mid i)$，用可执行性/affordance 给"能不能做成"的分数 $p(c_\pi \mid s, \ell_\pi)$，两者相乘选择下一步技能并迭代执行直到 "done"。Ahn 等 - 2022 - Do As I Can, Not...Ahn 等 - 2022 - Do As I Can, Not...

（你下次就可以说：**"SayCan 框架/方法"** 或 **"SayCan 规划算法"**）

---

2) 两句话讲创新点（你写 related work 够用）

> **创新点 1：把"语言合理性"和"世界可行性"解耦**：语言侧做 task-grounding（下一步是否有用），世界侧做 world-grounding（当前状态下是否能成功），并在同一概率接口中融合。Ahn 等 - 2022 - Do As I Can, Not... Ahn 等 - 2022 - Do As I Can, Not...

> **创新点 2：用"评分式 LLM + 严格响应结构"做技能选择**：不是让 LLM 自由生成计划，而是对候选技能进行评分，并把已选技能追加到上下文里做迭代规划。Ahn 等 - 2022 - Do As I Can, Not...Ahn 等 - 2022 - Do As I Can, Not...

---

3) 两句话讲局限（你写 critique/讨论部分够用）

> **局限 1：能力上限受限于技能库与低层技能可靠性**：只有当技能足够多且足够稳定，SayCan 的长序列规划才真正有用；不稳定或不适合长任务的技能会被弃用。 Ahn 等 - 2022 - Do As I Can, Not...Ahn 等 - 2022 - Do As I Can, Not...

> **局限 2：闭环反馈有限**：在原始设定中，系统主要通过当前决策步的 value/affordance 得到环境反馈；如果技能失败或环境改变，反馈可能不足，后续工作用 inner monologue 等方式补强闭环。 Ahn 等 - 2022 - Do As I Can, Not...Ahn 等 - 2022 - Do As I Can, Not...

---

4) "落地结果"一句话（可选，写实验动机时用）

> 作者在真实厨房验证了从 mock 到 real 的迁移：规划成功率下降约 3%，执行成功率下降约 14%，显示底层策略与 value/affordance 具有一定泛化能力。 Ahn 等 - 2022 - Do As I Can, Not...

---

5) 你课题专用版本（工业机械臂 + 模糊指令语义消歧）

如果你要把 SayCan 这套话术改成"你的论文会用的版本"，我建议这样写：

> **我们借鉴 SayCan 的技能级规划思想，将模糊高层指令转化为"候选技能集合上的排序问题"**：语言侧提供"语义上更像下一步"的评分，世界侧用工业约束/可达性/碰撞/对准成功率等构造"可执行性评分"，两者融合选择下一步动作并迭代更新，从而在执行过程中逐步消歧。 （对应 SayCan 的 $p(\ell_\pi \mid i) \times p(c_\pi \mid s, \ell_\pi)$ 结构）Ahn 等 - 2022 - Do As I Can, Not...

确实，LLM模型就是用来做这个推理的，那么不同的LLM模型在推理上面的不同体现着他们之间的优劣

接下来阅读inner monologue论文