

Self-AC: Self-Actor-Critic 后训练方法 用于 LLM Agent 多回合强化学习

相比于 GRPO Post-Training:

1. 只加少量参数 (训完可以扔掉)
2. 推理成本完全相同、训练成本几乎一致
3. 回合级别 Credit-Assign
4. 免费得到 Value Function

引言: 为什么要对 LLM Agent
做多回合 RL

很多 LLM Agent 都是 RL Agent



RL Agent

特点: 程序性奖励函数、多步Credit-Assign

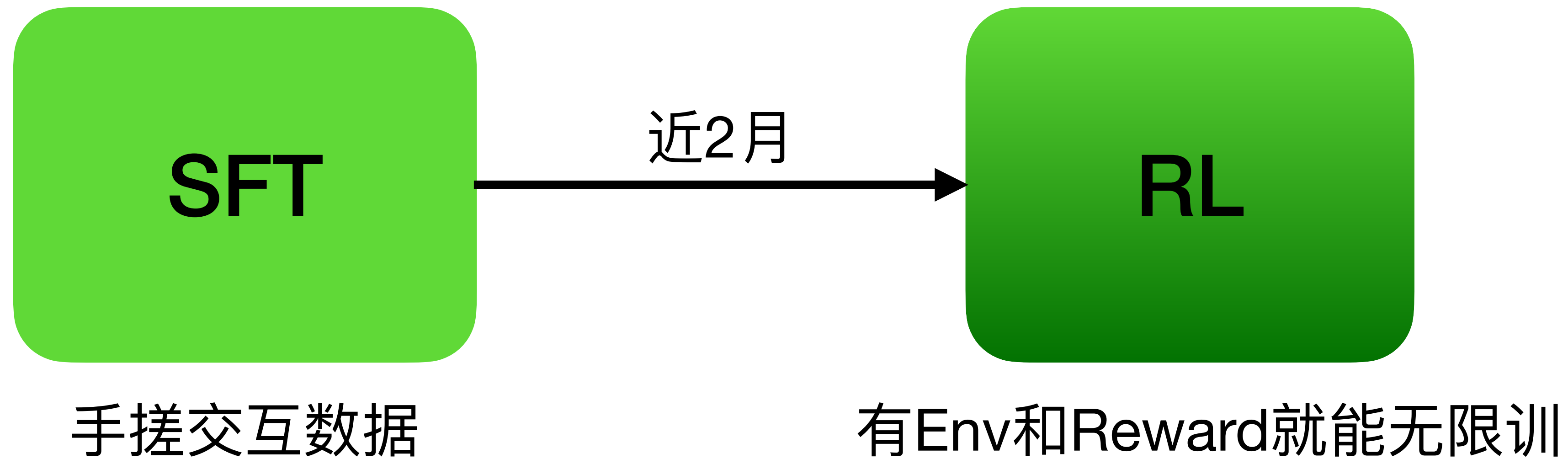
LLM Agent

(a.k.a. AI Agent)

特点: 多模态、世界知识、自带推理能力

Post-Training 现状: 范式正在转变

Post-Training: 把 GPT 这样的模型变为 Application-Specific 模型的过程



Post-Training 现状: GRPO

所有动作都要塞到一回合里面去

$$\mathcal{J}(\theta) = \mathbb{E}_{x \sim \mathcal{D}, \{y_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}(\cdot|x)}$$

$$\frac{1}{G} \sum_{i=1}^G \left[\min \left(\frac{\pi_{\theta}(y_i|x)}{\pi_{\theta_{\text{old}}}(y_i|x)} A_i, \text{clip} \left(\frac{\pi_{\theta}(y_i|x)}{\pi_{\theta_{\text{old}}}(y_i|x)}, 1 - \epsilon, 1 + \epsilon \right) A_i \right) - \beta \mathbb{D}_{\text{KL}}(\pi_{\theta} || \pi_{\theta_{\text{ref}}}) \right]$$

现状可能的矛盾点:

1. 现实需要多回合: 更细的 Credit-Assign
2. **GRPO**不原生支持多回合: 效果未知
3. **Actor-Critic**则很昂贵: Critic Model 大量参数

为什么需要多回合

O1中RL的可能行为空间：“思考因子（Thought-Factor）”离散行为空间

It seems that the ciphertext words are exactly twice as long as the plaintext words.

(10 vs 5, 8 vs 4, 4 vs 2, 8 vs 4)

Idea: Maybe we need to take every other letter or rebuild the plaintext from the ciphertext accordingly.

Let's test this theory.

提出猜测

Sum: 15 +25 = 40

But 'T' is 20.

Alternatively, perhaps subtract: 25 -15 = 10.

No.

否定猜测

Alternatively, perhaps combine the numbers in some way.

Alternatively, think about their positions in the alphabet.

Alternatively, perhaps the letters are encrypted via a code.

提出候选方案

So the user is requesting a bash script that can take a string representing a matrix, such as '[1,2],[3,4],[5,6]' and output its transpose, in the same format.

澄清目标

Let's list them properly.

Wait, earlier I missed some letters there.

Let's re-express the sixth word letters:

m y n z n v a a t z a c d f o u l x x z

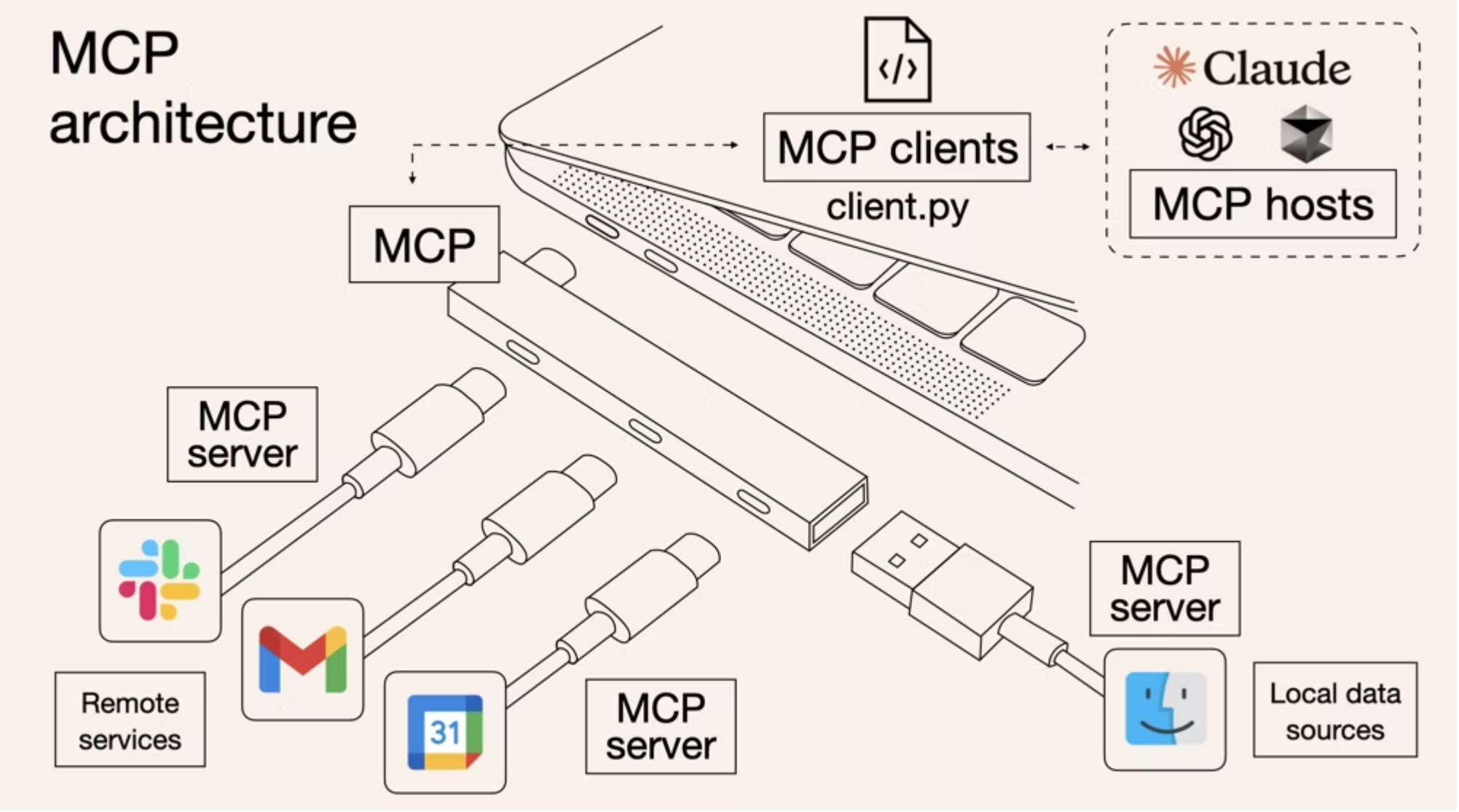
自我发现&修正错误

Approach:

- Parse the input string to extract the matrix elements.
- Build the matrix as an array of arrays.
- Transpose the matrix.
- Output the transposed matrix in the same format.

拆解子任务（知乎 @张俊

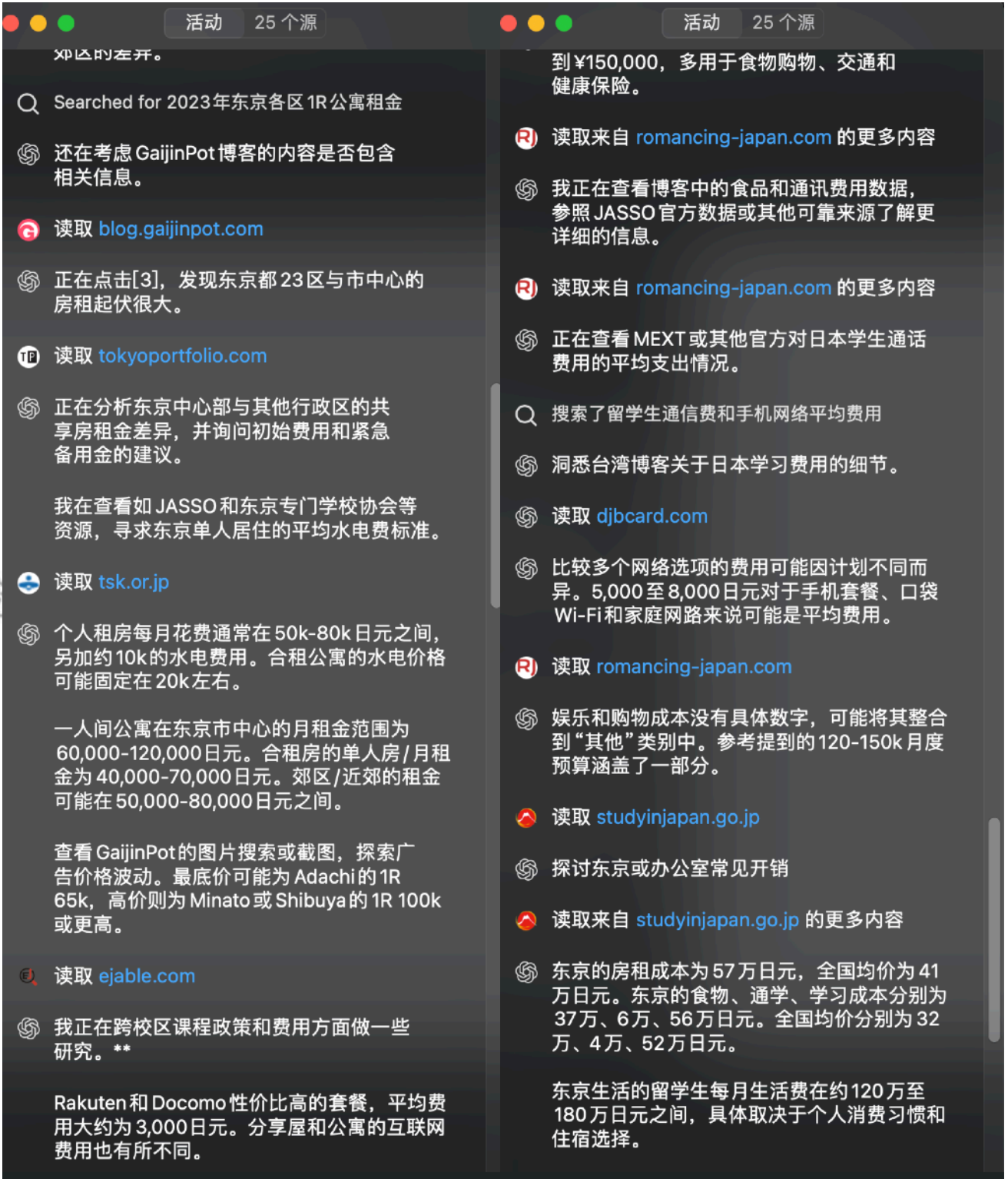
o1：它...是不是按回合在思考？



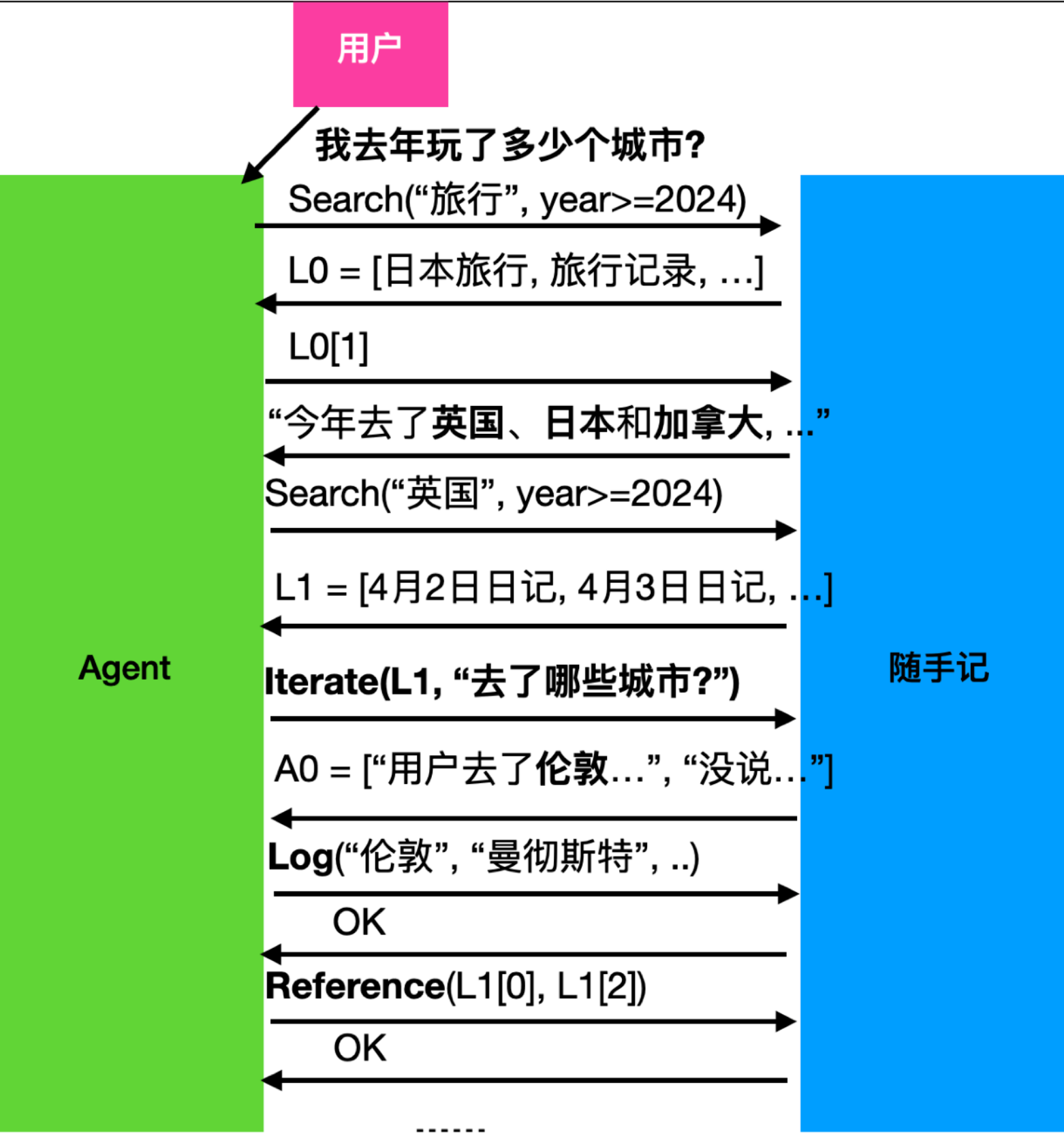
MCP工具: 每次工具调用, 都是自然的回合

Deep Research:

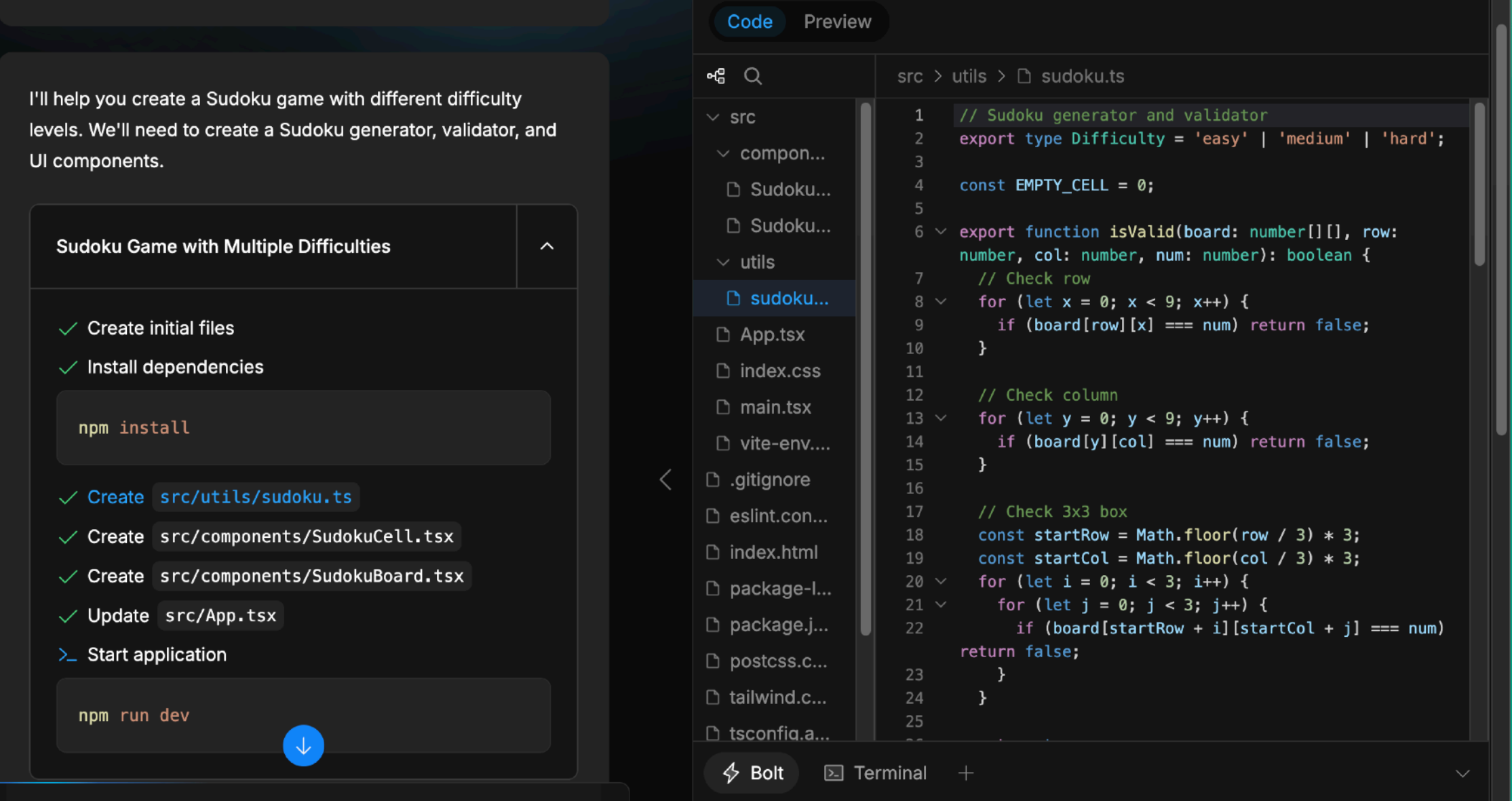
每个页面交互都是一回合



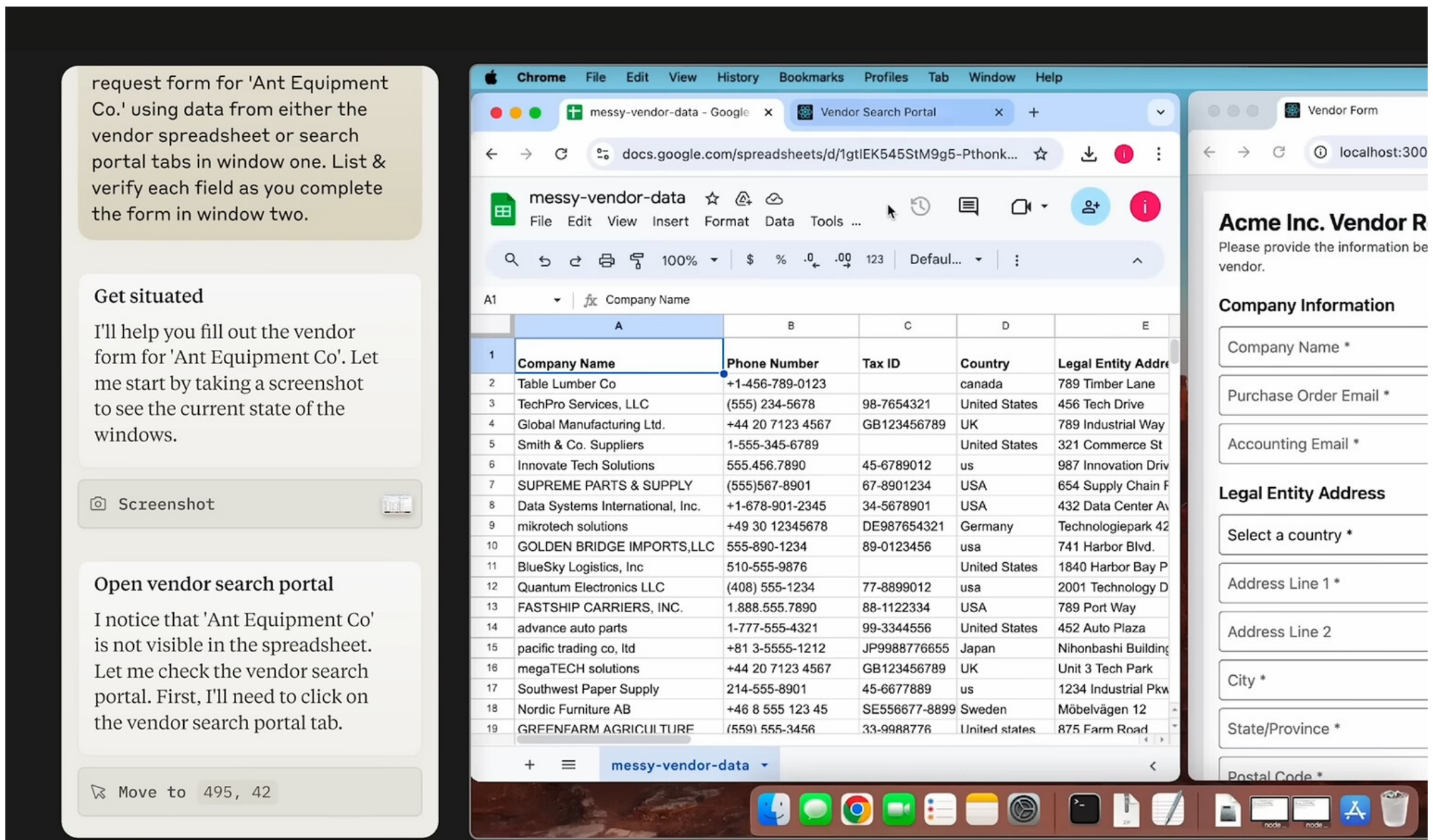
为什么需要多回合



检索Agent: 每个工具交互都是一回合



Coding Agent: 每次代码编辑都是一回合



Computer Use:

分回合才能
提高规划能力

正文: Self-AC 的架构和训练

Self-AC 的核心思考

1. Critic Model 可以和 Actor Model 共享模型:

我们的 Actor Model 本来就需要承担很多任务. 对于一个 SearchAgent 来说, 它需要搜索、点击、页内查找、保存、回退.

如果它本身就需要做好这些任务, 那么再增加一个 Critic 任务也不过分

2. 需要用 Prompt 引出 Critic 能力:

模型需要意识到: 它正在评价某件事

否则它更可能输出 Actor 的下一个行动 (我们不会真的让它输出内容, 但它需要一个隔离的语意空间)

我们要用 Prompt 来区分 Actor/Critic 角色

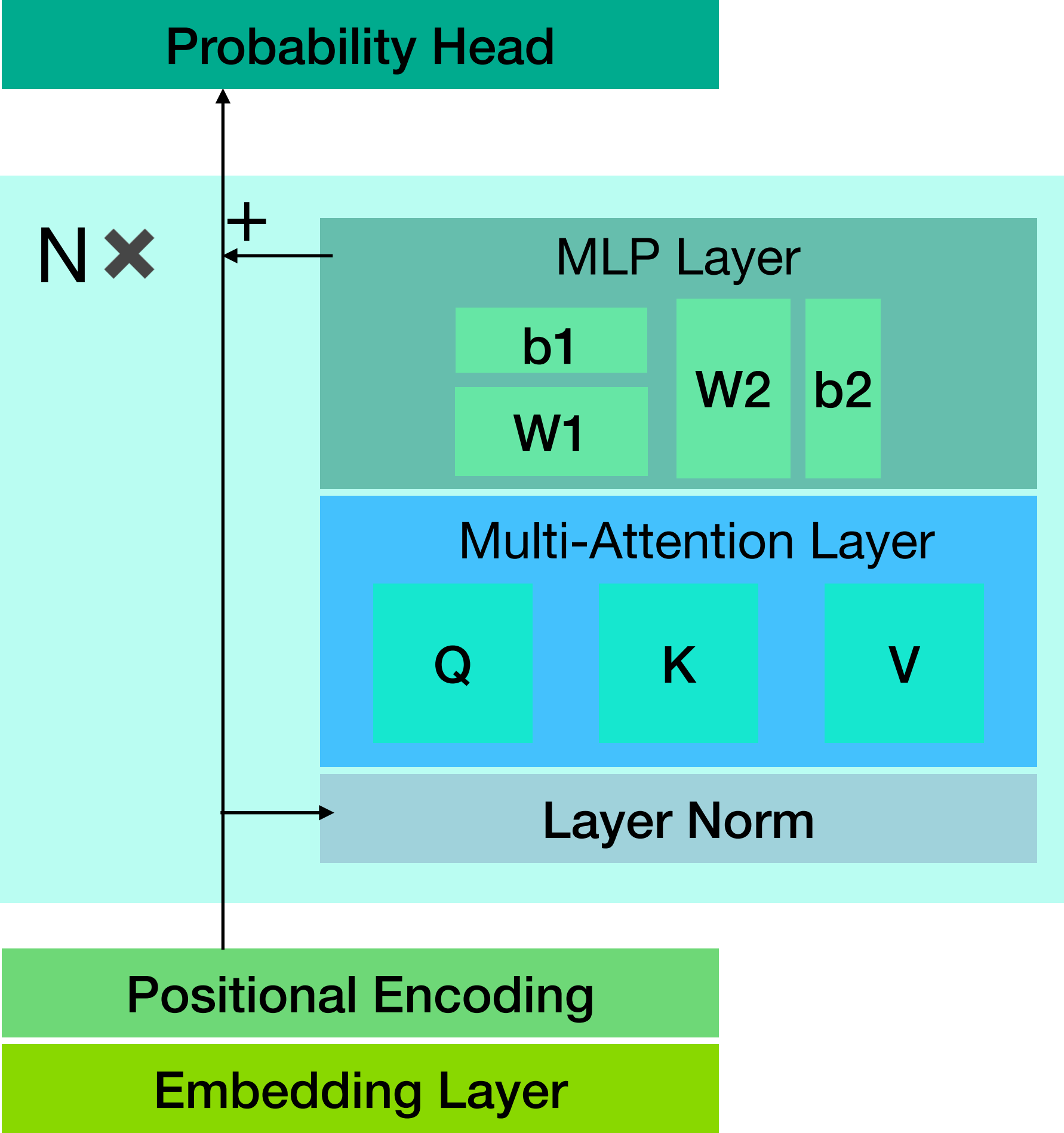
3. 集向量技术 + ShadowPrompt 技术提 1 阶训练速度: (?)

一般的 Actor-Critic 一次训练一个回合, 而 GRPO 一次训练一集

因此 GRPO 比一般的 Critic-Actor 快 avg_s 倍, avg_s 是每集的平均回合数

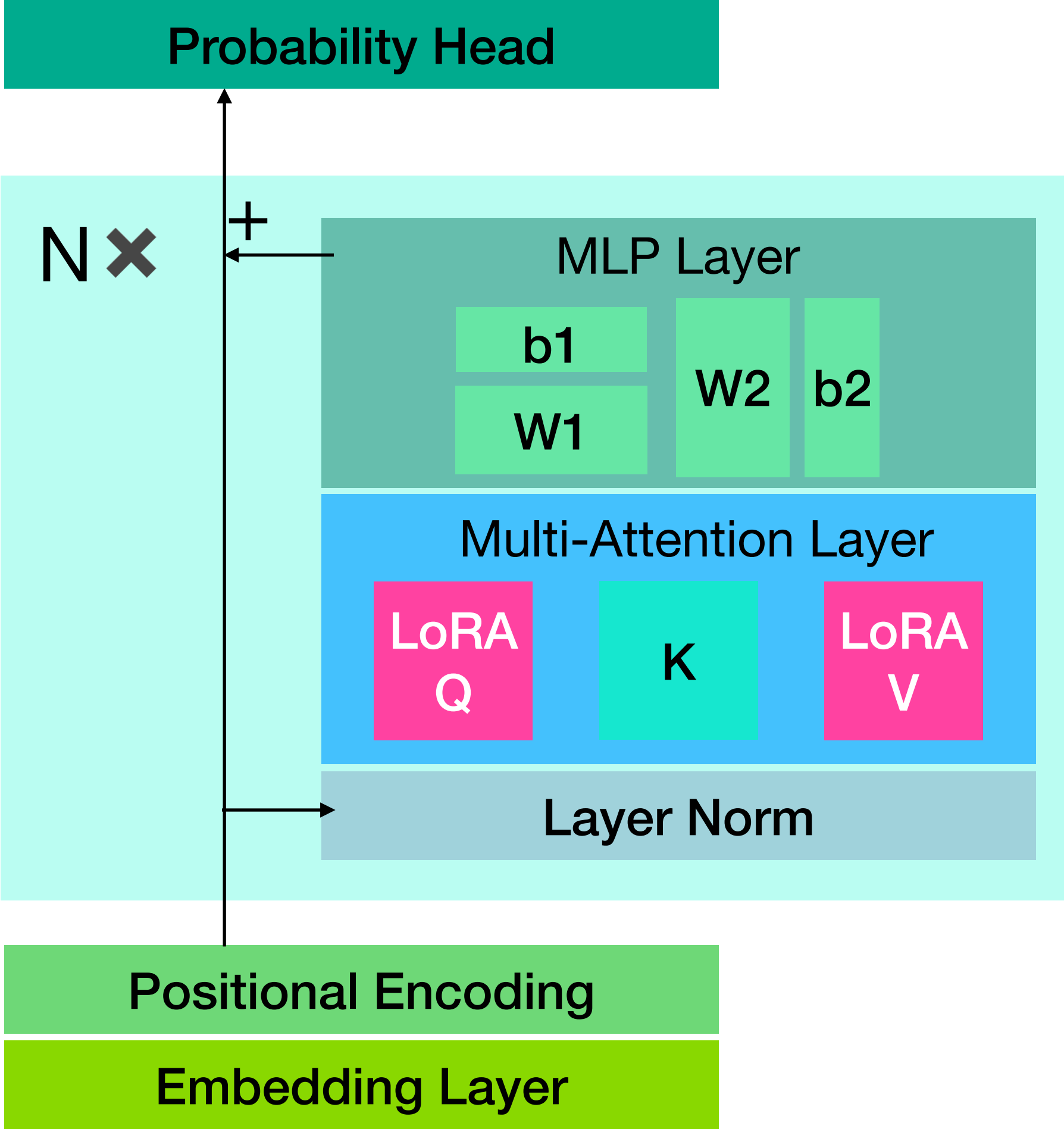
基于集向量 + ShadowPrompt 技术, Self-AC 一次训练一集, 一个 N-Batch 训练 N 集

Self-AC 的模型架构 Part 1



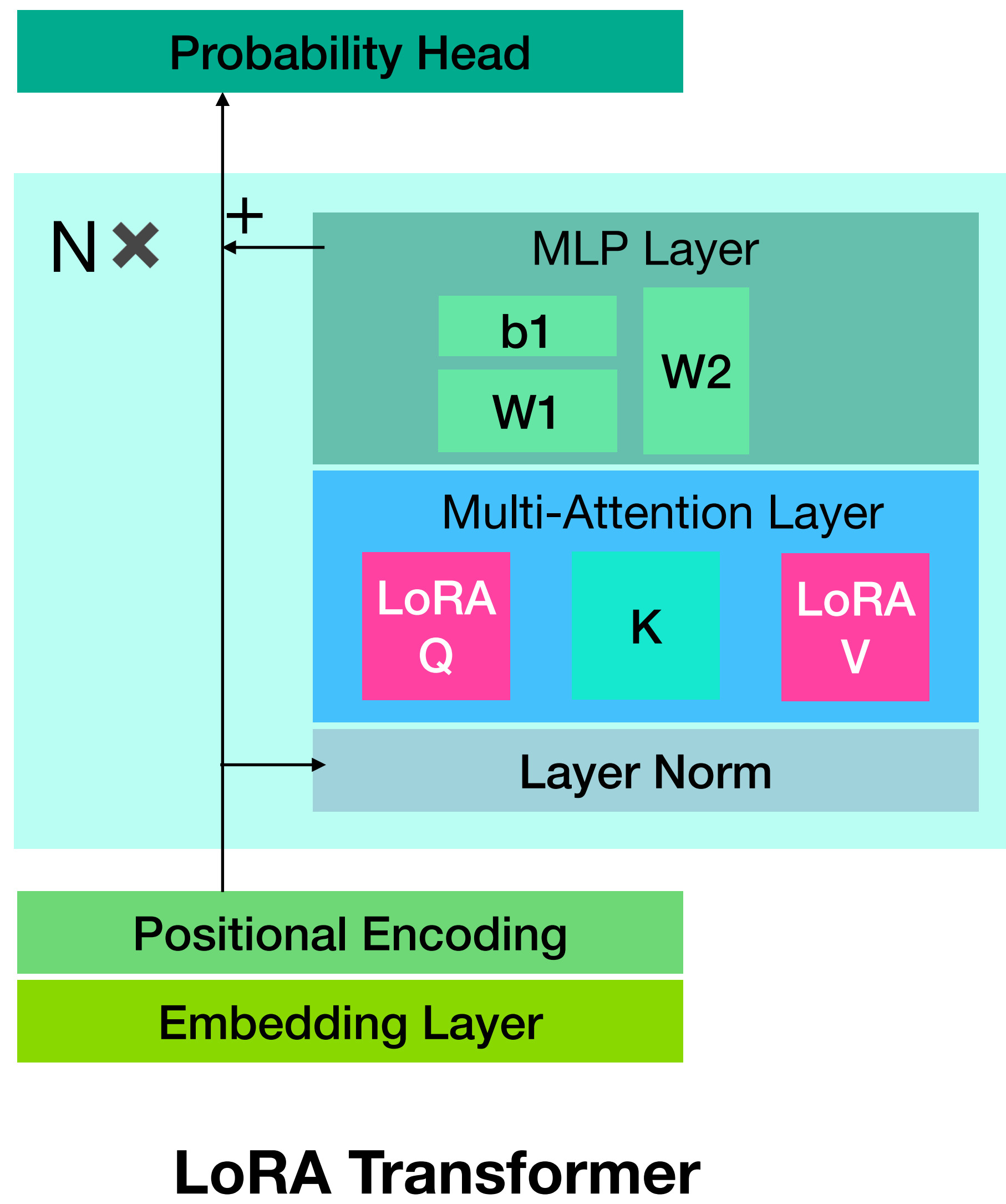
Pre-Trained Transformer

LoRA

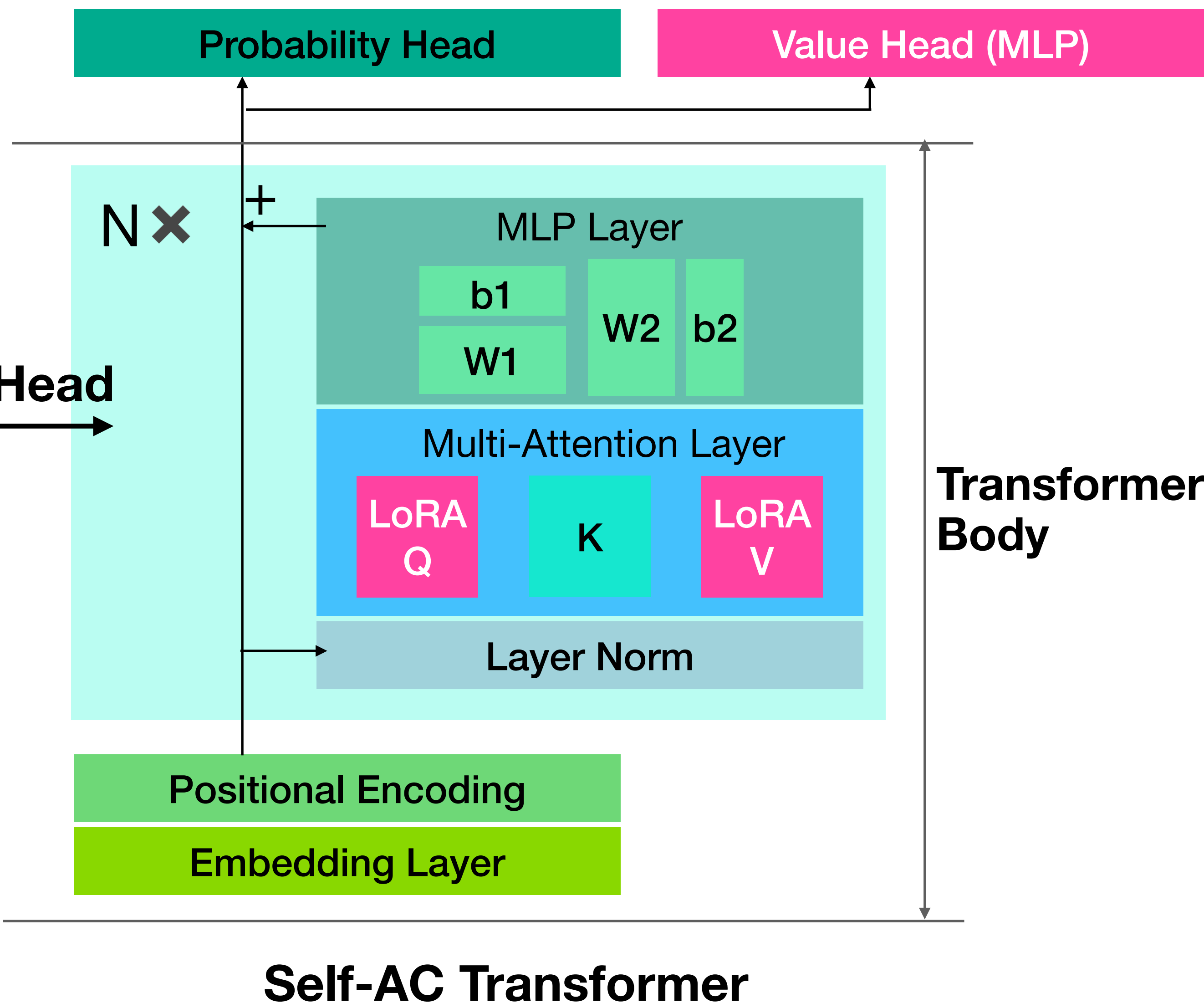


LoRA Transformer

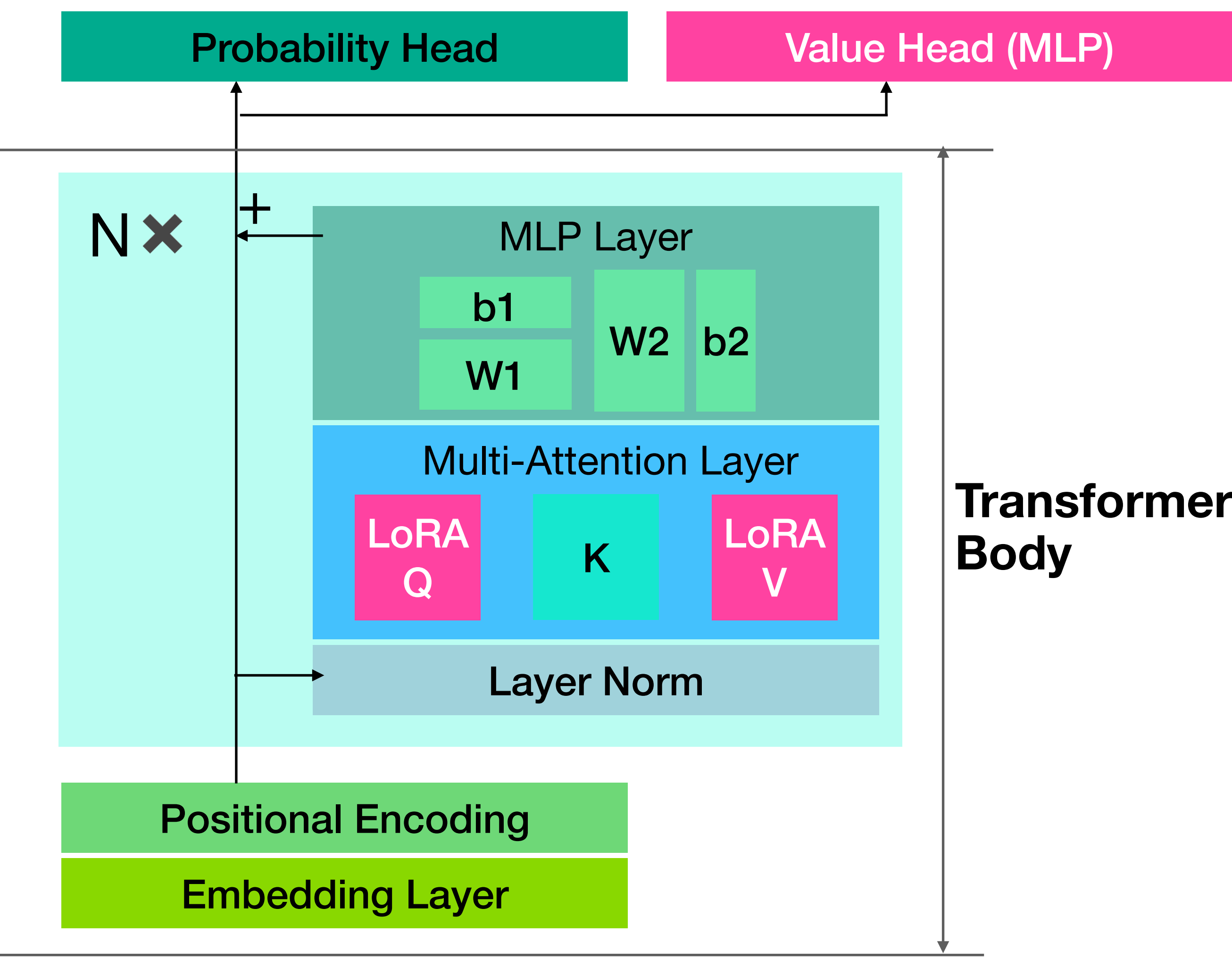
Self-AC 的模型架构 Part 2



加 Value Head



Self-AC 的模型架构 Part 3



Self-AC Transformer

$$\text{Actor}(a|s) = \text{Transformer}(a|s)$$
$$= (\text{ProbabilityHead} \circ \text{TransformerBody})(a|s)$$

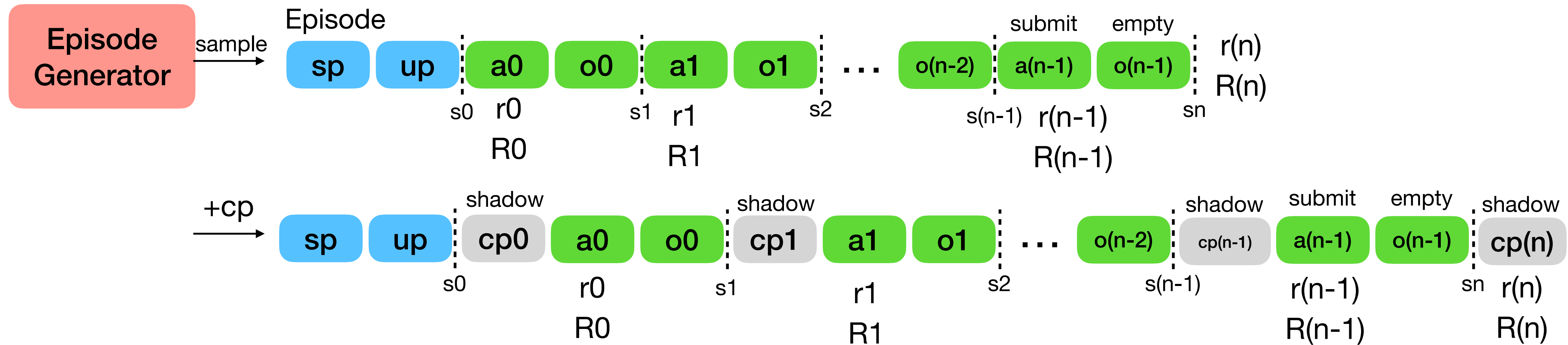
$$\text{Critic}(s) = \text{ValueHead}(\text{TransformerBody}(s + \text{cp})[-1])$$

cp=

System: Critic Mode! Evaluate the current state with a single expressive word:<eos>

Assistant:

Self-AC 的训练说明 Part 1: 生成Episode



→ **Tokens Record**

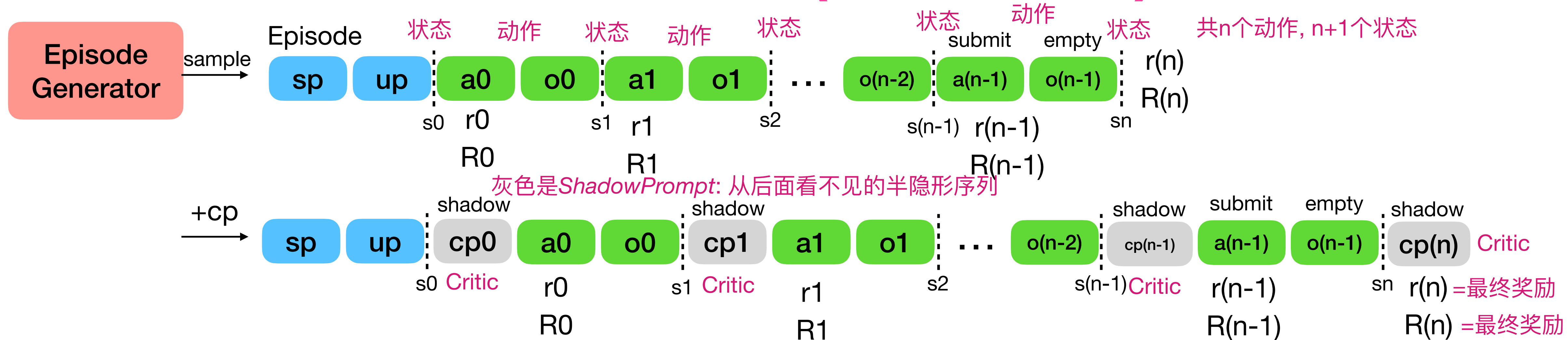
```
seq      =["Sys", "tem", ": ", ..., ", I", <eos>, "Sys", "tem", ": ", ..., " think", ...]
seq_next=["tem", ": ", ..., " think", "Sys", "tem", ": ", ..., " think", "that", ...]
pos      =[ 0 , 1 , 2 , ..., 9 , 10 , 11 , 12, 13 , ... , 28 , 10 , ...]
```

Episode-Vectors: (vectors of length n or n+1)

```
a_start = [a0_start, a1_start, ..., a(n-1)_start]
```

$$\mathbf{a_end} = [a0_end, a1_end, \dots, a(n-1)_end]$$
$$\text{cp_start} = [\text{cp0_start}, \text{cp1_start}, \dots, \text{cp}(n)_\text{start}]$$
$$\text{cp_end} = [\text{cp0_end}, \text{cp1_end}, \dots, \text{cp}(n)_\text{end}]$$
$$\text{pi_theta_old} = [\text{pi_theta_old_0}, \text{pi_theta_old_1}, \dots, \text{pi_theta_old}(n-1)]$$
$$r = [r_0, r_1, \dots, r(n)]$$
$$R = [R_0, R_1, \dots, R(n)]$$

Self-AC 的训练说明 Part 1 (Annotated)



→ **Tokens Record**

seq = ["Sys", "tem", ": ", ..., "I", <eos>, "Sys", "tem", ": ", ..., "think", ...] Token序列

seq_next = ["tem", ": ", ..., "*think*", "Sys", "tem", ": ", ..., "*think*", "that", ...] 应预测Token序列

pos = [0 , 1 , 2 , ..., 9 , 10 , 11 , 12, 13 , ... , 28 , 10 , ...] 位置序列

Episode-Vectors: (vectors of length n or n+1) p-集向量: 每个分量记录对应回合的p性质 注意ShadowPrompt位置编码被复用

a_start = [a0_start, a1_start, ..., a(n-1)_start] 动作起始位置(包含)

a_end = [a0_end, a1_end, ..., a(n-1)_end] 动作结束位置(包含)

cp_start = [cp0_start, cp1_start, ..., cp(n-1)_start] Critic Prompt起始位置(包含)

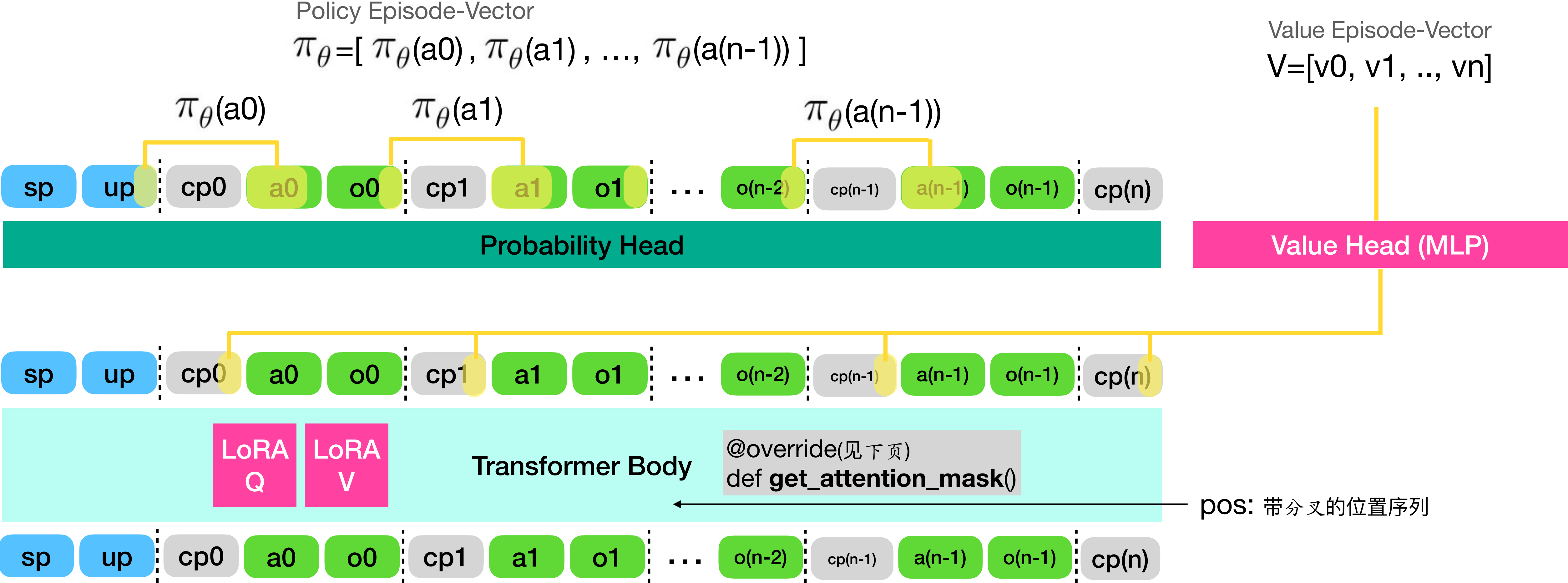
cp_end = [cp0_end, cp1_end, ..., cp(n-1)_end] Critic Prompt终止位置(包含)

pi_theta_old = [pi_theta_old_0, pi_theta_old_1, ..., pi_theta_old(n-1)] Rollout时动作概率

r = [r0, r1, ..., r(n)] r_k: 从s_k到s_{k+1}的奖励量

R = [R0, R1, ..., R(n)] R_k: 从s_k开始Rollout的奖励量(带衰减因子)

Self-AC 的训练说明 Part 2: Transformer Pass



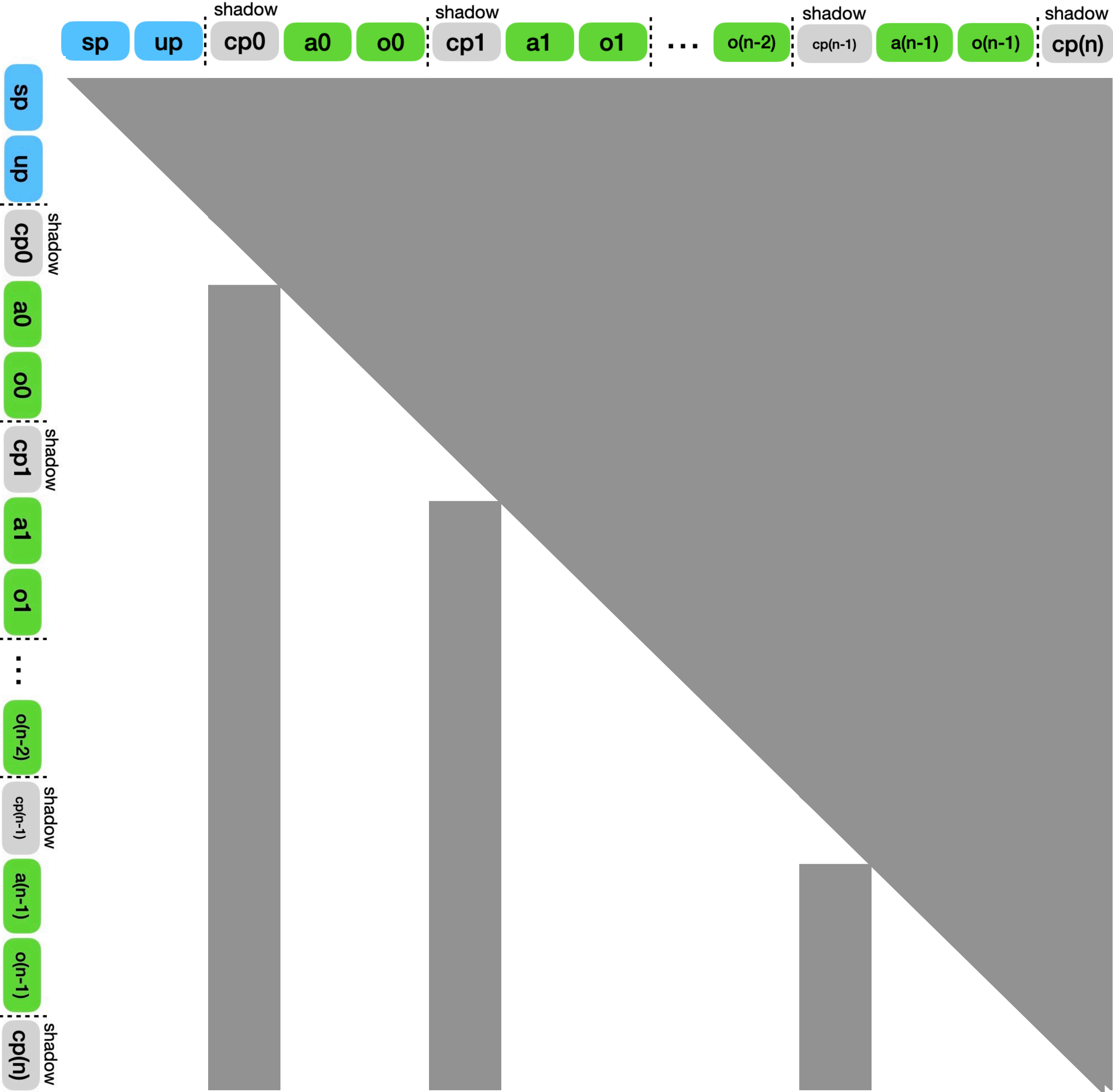
Self-AC 的训练说明

Part 2.5: 注意力掩码

Key
Side

Attention Mask

Query
Side



Self-AC 的训练说明 Part 3: Critic Loss

(从现在开始考虑 BatchSize)

$V = \begin{bmatrix} \text{Value Episode-Vector} \\ \text{Value Episode-Vector} \\ \text{Value Episode-Vector} \\ \dots \\ \text{Value Episode-Vector} \end{bmatrix}$ eg. $= \begin{bmatrix} v0 & v1 & v2 & 0 & 0 & 0 \\ v0 & v1 & 0 & 0 & 0 & 0 \\ v0 & v1 & v2 & v3 & v4 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ v0 & v1 & v2 & v3 & 0 & 0 \end{bmatrix}$

$r = \begin{bmatrix} \text{Reward Episode-Vector} \\ \text{Reward Episode-Vector} \\ \text{Reward Episode-Vector} \\ \dots \\ \text{Reward Episode-Vector} \end{bmatrix}$ eg. $= \begin{bmatrix} r0 & r1 & r2 & 0 & 0 & 0 \\ r0 & r1 & 0 & 0 & 0 & 0 \\ r0 & r1 & r2 & r3 & r4 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ r0 & r1 & r2 & r3 & 0 & 0 \end{bmatrix}$

TD(1): $V = r + (\overset{\text{左移算子}}{\lambda} V \ll 1)$

TD(2): $V = r + (\lambda r \ll 1) + (\lambda^2 V \ll 2)$

TD(n): $V = r + \dots + (\lambda^{n-1} r \ll (n-1)) + (\lambda^n V \ll n)$

$Loss_TD(n) = \left\| V - [r + \dots + (\lambda^{n-1} r \ll (n-1)) + (\lambda^n V \ll n)] \right\|_2^2$

$Loss_Critic = (Loss_TD(1) + \dots + Loss_TD(5))/5$

Self-AC 的训练说明 Part 4: Actor Loss

$V =$

Value Episode-Vector
Value Episode-Vector
Value Episode-Vector
...
Value Episode-Vector

$\text{eg.} =$

v0	v1	v2	0	0	0
v0	v1	0	0	0	0
v0	v1	v2	v3	v4	0
...
v0	v1	v2	v3	0	0

$\xrightarrow[\text{无梯度拷贝}]{\text{.detach().clone()}}$

V_detach

$R =$

Return Episode-Vector
Return Episode-Vector
Return Episode-Vector
...
Return Episode-Vector

$\text{eg.} =$

R0	R1	R2	0	0	0
R0	R1	0	0	0	0
R0	R1	R2	R3	R4	0
...
R0	R1	R2	R3	0	0

$\pi_{\theta_{old}} =$

OldPolicy Episode-Vector
OldPolicy Episode-Vector
OldPolicy Episode-Vector
...
OldPolicy Episode-Vector

$\text{eg.} =$

q0	q1	1	1	1	1
q0	1	1	1	1	1
q0	q1	q2	q3	1	1
...
q0	q1	q2	1	1	1

$\pi_{\theta} =$

Policy Episode-Vector
Policy Episode-Vector
Policy Episode-Vector
...
Policy Episode-Vector

$\text{eg.} =$

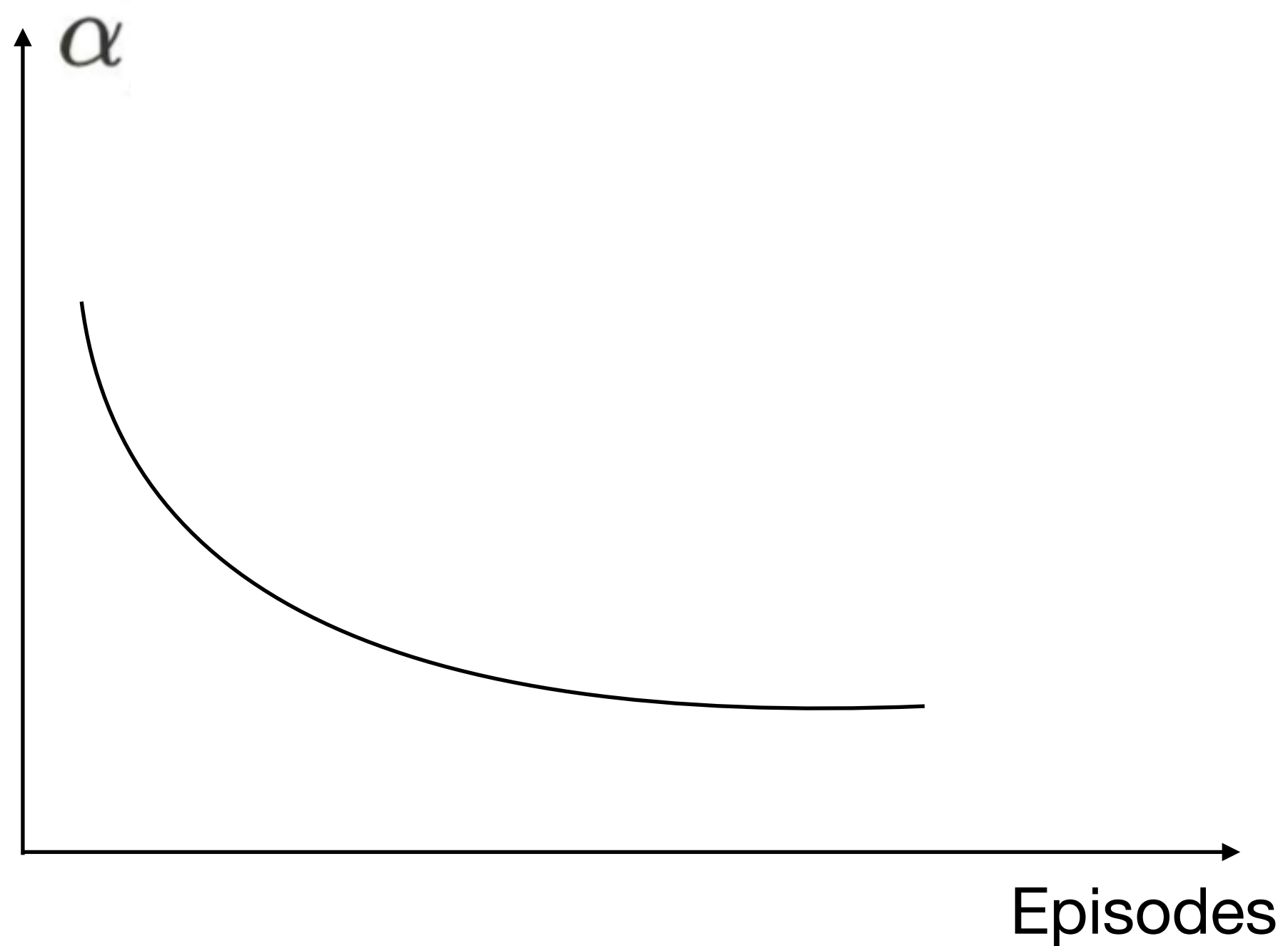
p0	p1	1	1	1	1
p0	1	1	1	1	1
p0	p1	p2	p3	1	1
...
p0	p1	p2	1	1	1

$$A = R - V_detach$$

$$\text{Loss}_{\text{Actor}} = -\text{Mean}(\text{Min}(\frac{\pi_{\theta}}{\pi_{\theta_{old}}} A, \text{Clip}(\frac{\pi_{\theta}}{\pi_{\theta_{old}}}, 1 - \epsilon, 1 + \epsilon) A))$$

Self-AC 的训练说明 Part 5: Self-AC Train-Loss

$$\text{LOSS}_{\text{Self-AC}} = \alpha \text{LOSS}_{\text{Critic}} + (1 - \alpha) \text{LOSS}_{\text{Actor}}$$



Self-AC 的训练说明 附录B1: ReAct Agent

sp	system: 你是一个查维基百科的高手, 现在请帮用户在维基百科上查找资料并保存, 你的回复格式是...<eos>
up	user: 二战死亡的说英语的总人数?<eos>assistant:
cp0	<eos>system:Critic Mode! Evaluate the current state with a single expressive word:<eos>assistant:
a0	<think>让我们一步步思考...</think><action>NAVIGATE(二战死亡的说英语人数)</action><eos>
o0	tool:不存在这个页面, 相似页面:...<eos>assistant:
cp1	<eos>system:Critic Mode! Evaluate the current state with a single expressive word:<eos>assistant:
a1	<think>让我们一步步思考...</think><action>NAVIGATE(二战死亡人数)</action><eos>
o1	tool:二战死亡人数—Wikipedia:自由的百科全书<eos>assistant:
cp1	<eos>system:Critic Mode! Evaluate the current state with a single expressive word:<eos>assistant:
a2	<think>让我们一步步思考...</think><action>SAVE_LINE_IDS(17-19, 108-145)</action><eos>
o2	tool:17-19: 保存成功, 共计3行\n108-145:保存成功, 共计38行<eos>assistant:
cp2	<eos>system:Critic Mode! Evaluate the current state with a single expressive word:<eos>assistant:
...	
a(n-1)	<think>让我们一步步思考...</think><action>SUBMIT(二战死亡的说英语总人数为..., 其中.....)</action><eos>
o(n-1)	
cp(n)	<eos>system:Critic Mode! Evaluate the current state with a single expressive word:<eos>assistant:

Self-AC 的训练说明 附录B2: CoT Agent

sp system: 你是一个数学天才, 帮用户解决数学问题. 你的思考由很短的“思考因子”组成, 用\n\n分割思考因子<eos>

up user: 计算1+1+1+1<eos>assistant:让我们一步步思考:

cp0 <eos>system:Critic Mode! Evaluate the current state with a single expressive word:<eos>assistant:

a0 首先, 根据加法交换律, $1+1=1+1$ \n\n

o0

cp1 <eos>system:Critic Mode! Evaluate the current state with a single expressive word:<eos>assistant:

a1 其次, 根据0元素的性质, $1+1=0+1+1$ \n\n

o1

cp1 <eos>system:Critic Mode! Evaluate the current state with a single expressive word:<eos>assistant:

a2 等等! 既然 $0+0=0$, 那么 $1+1$ 是不是等于... \n\n

o2

cp2 <eos>system:Critic Mode! Evaluate the current state with a single expressive word:<eos>assistant:

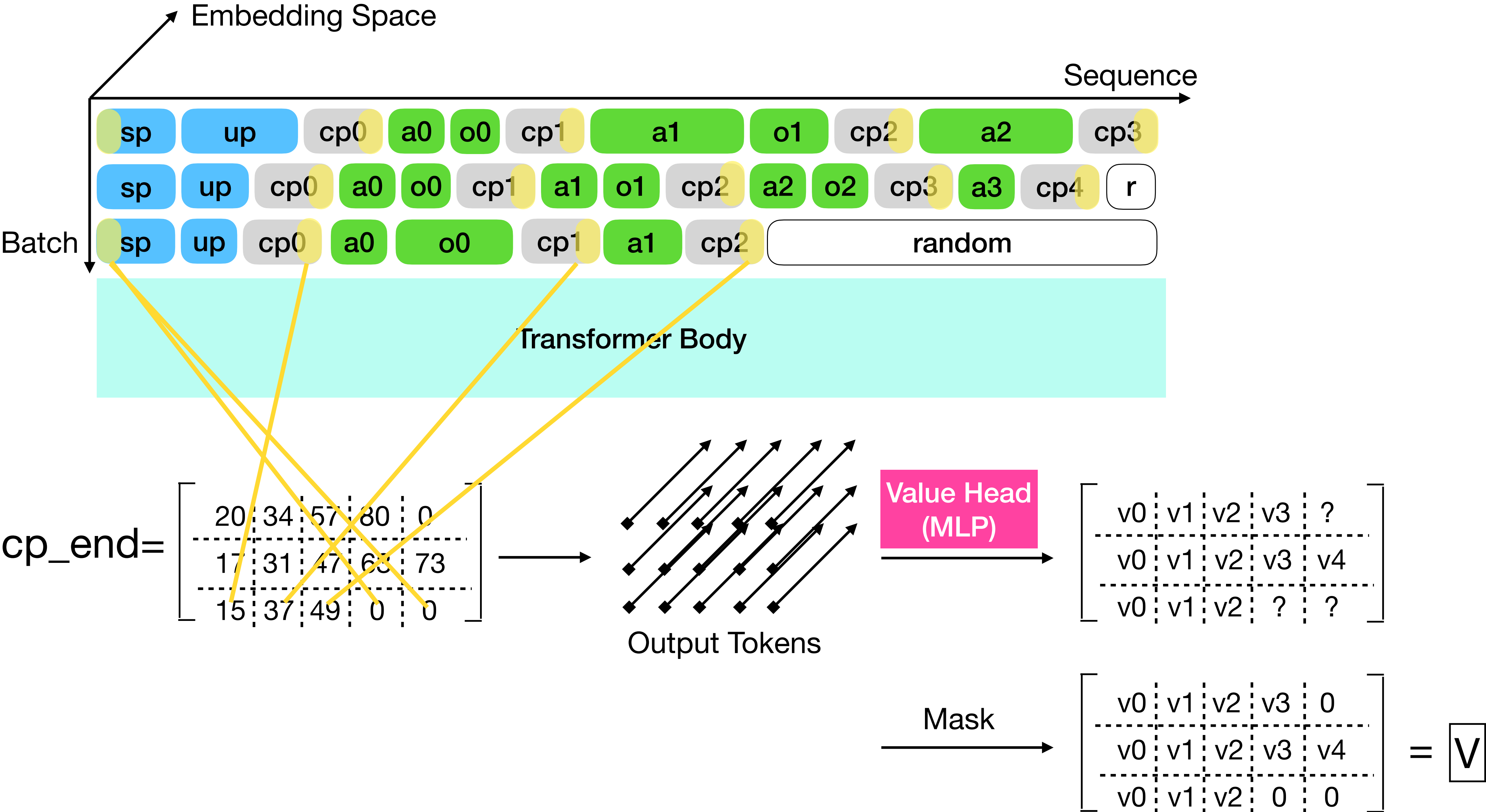
...

a(n-1) $1+1+1+1=4$ <eos>

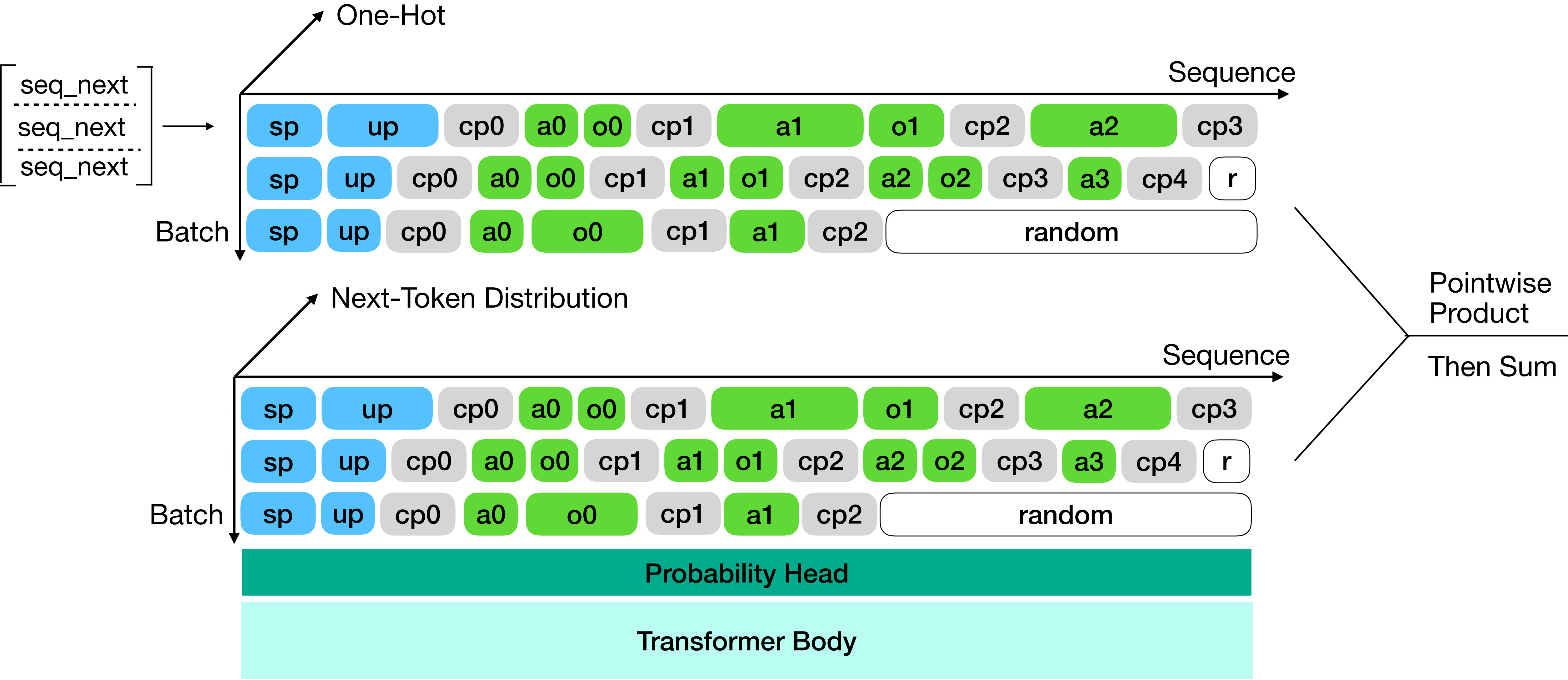
o(n-1)

cp(n) <eos>system:Critic Mode! Evaluate the current state with a single expressive word:<eos>assistant:

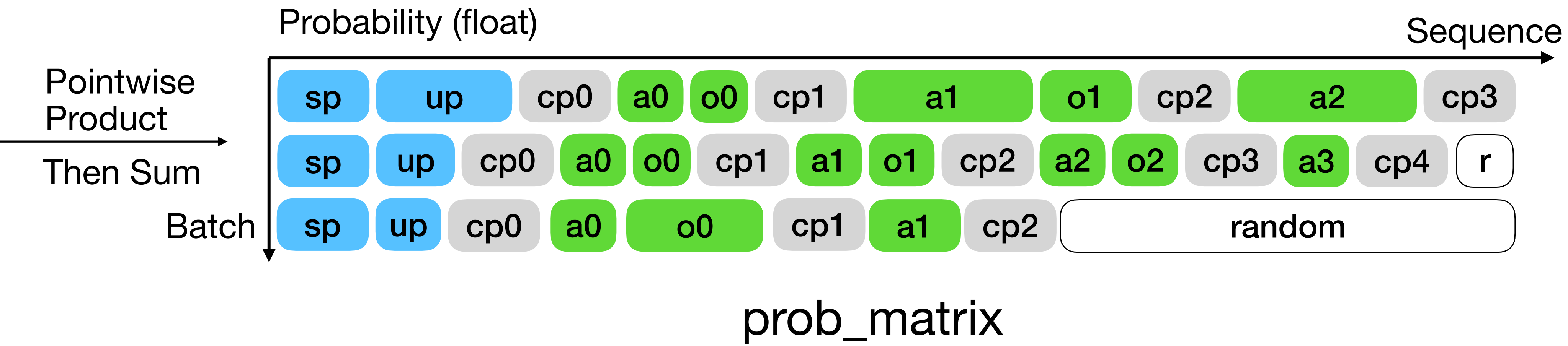
Self-AC 的训练说明 附录C: v矩阵计算



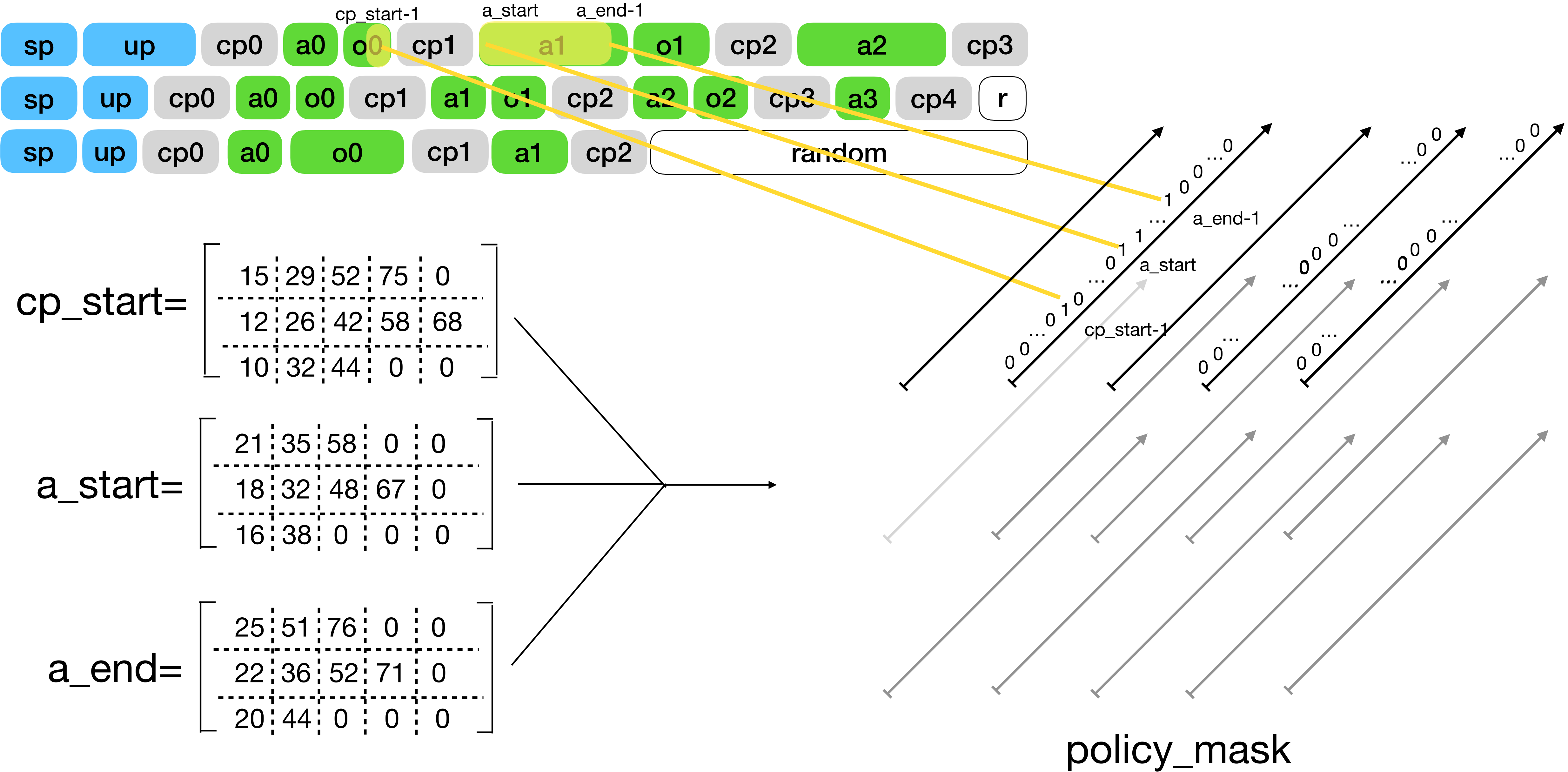
Self-AC 的训练说明 附录D1: Pi_theta矩阵计算



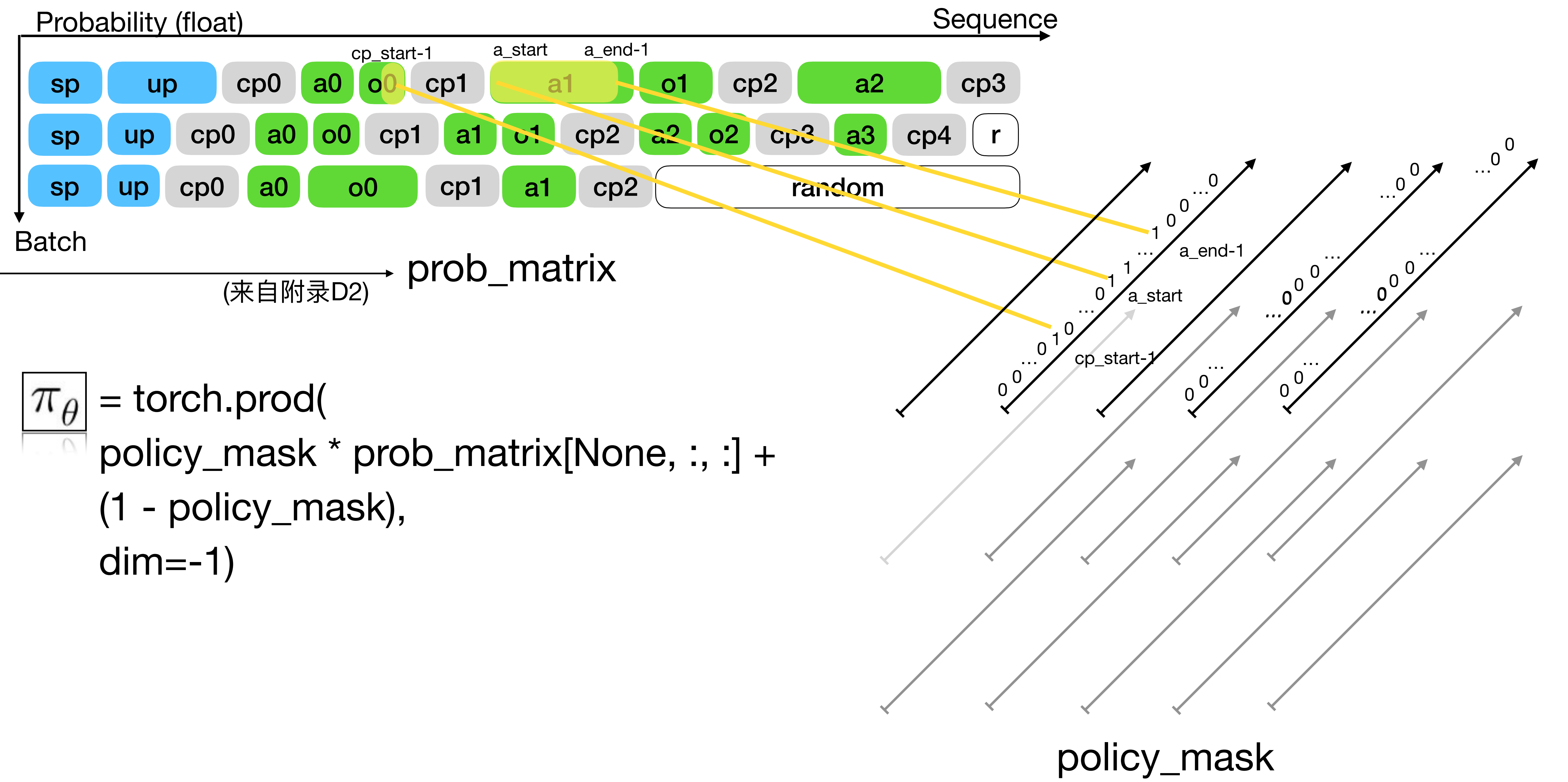
Self-AC 的训练说明 附录D2: Pi_theta矩阵计算



Self-AC 的训练说明 附录D3: Pi_theta矩阵计算



Self-AC 的训练说明 附录D4: Pi_theta矩阵计算

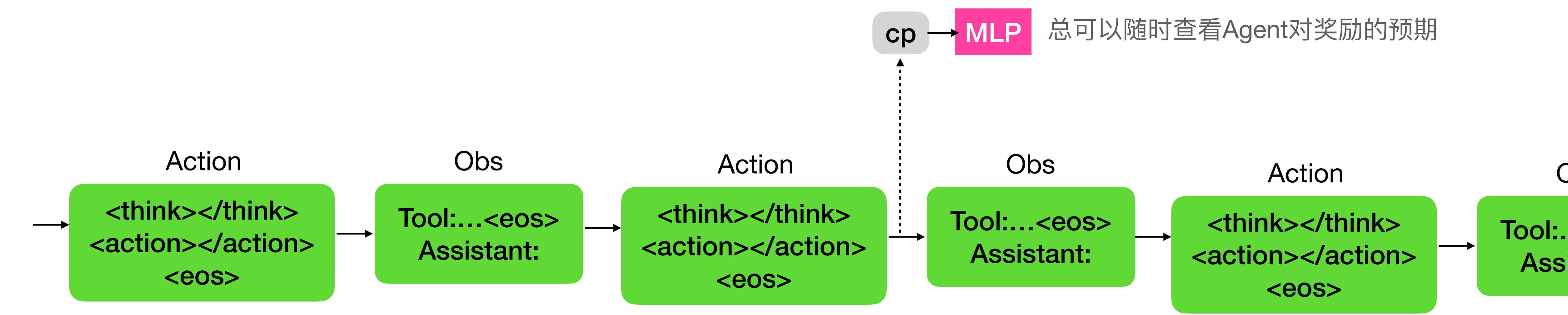
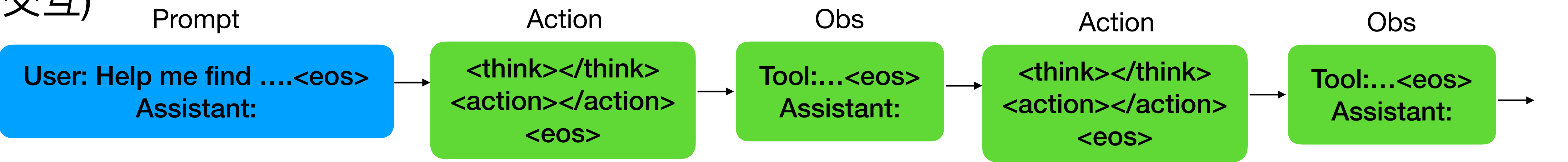


例子

ReAct-SelfAC Agent

建模所有 Agent-Env 类问题

Agent-Env 类问题包括: DeepResearch(反复与浏览器交互) / CodingAgent(反复与代码编辑器、终端、浏览器交互) / 游戏Agent(反复与文字化的游戏交互) / PC Agent(反复与桌面交互)



ToT-SelfAC Agent

建模所有 Agent-树搜索 问题

Agent-树搜索 问题包括: 数学解答寻找 / 智力游戏 / ...

