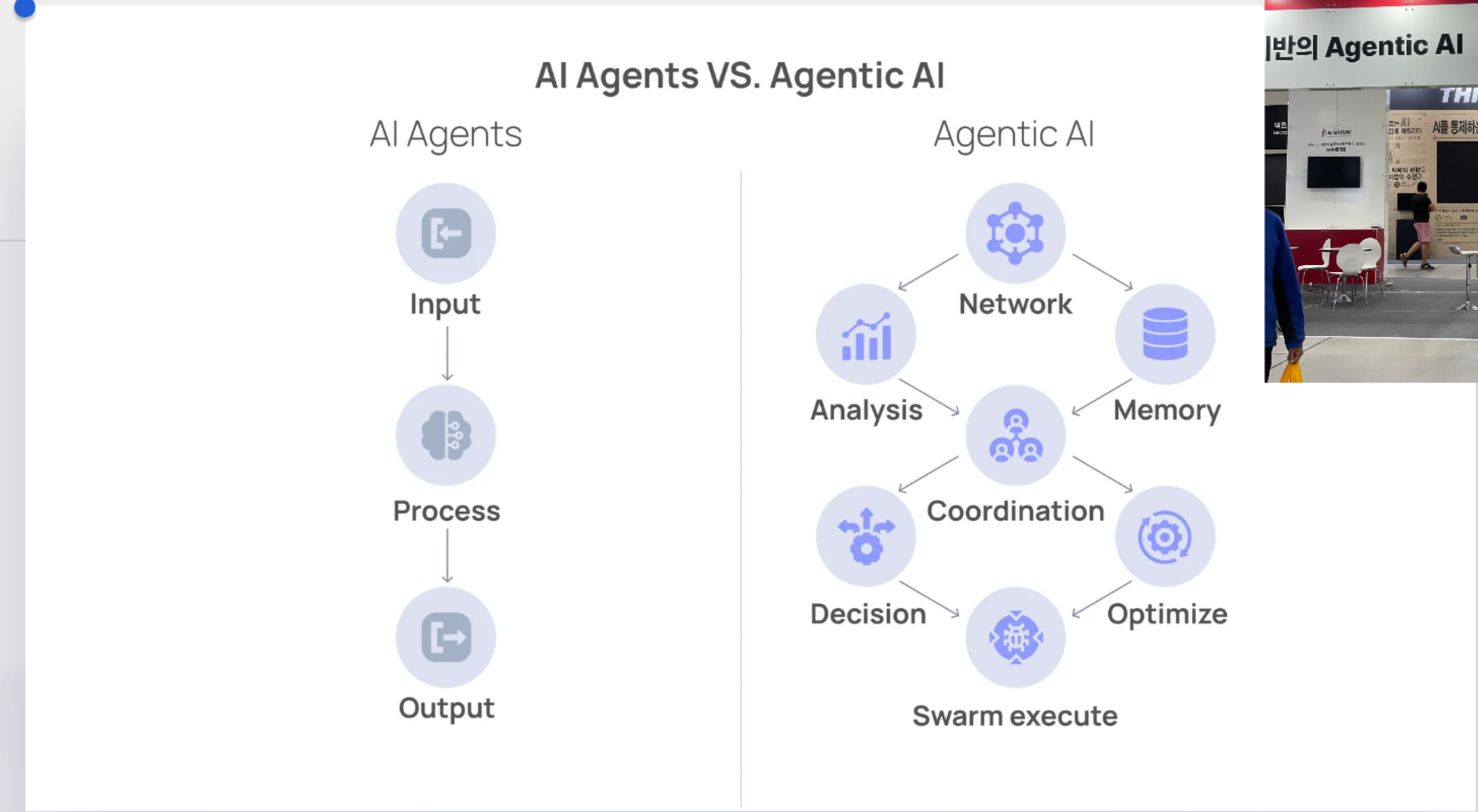
2025 연구실 세미나

HALO: Hierarchical Autonomous Logic-Oriented Orchestration for Multi-Agent LLM Systems

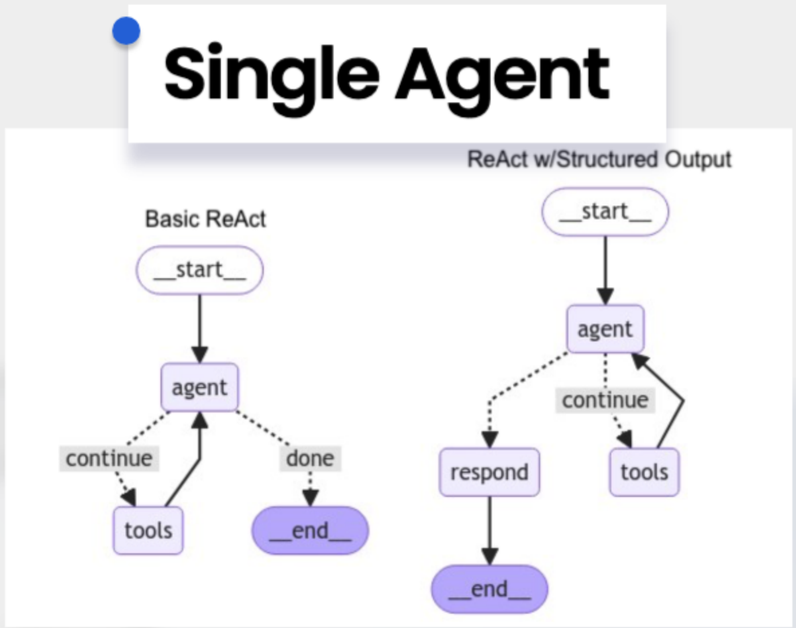
논문 세미나 발표

2025.10.15 강지윤

AI AGENT VS AGENTIC AI

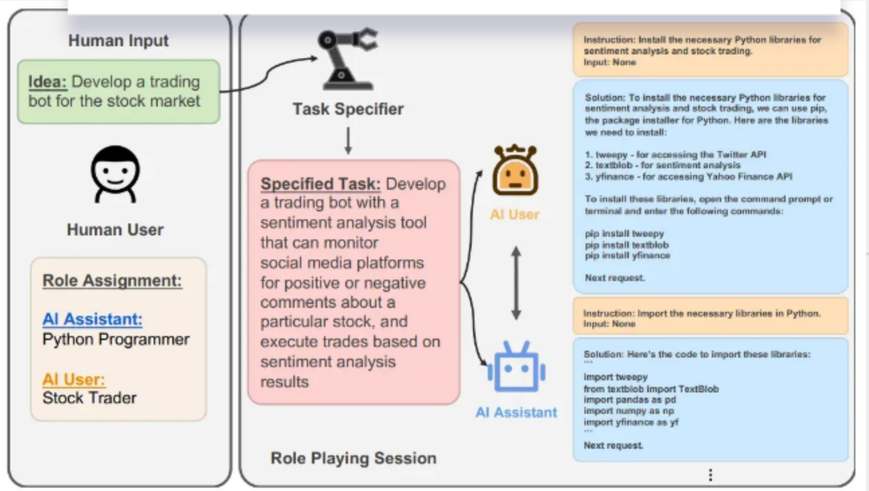


HISTORY OF AGENTIC AI

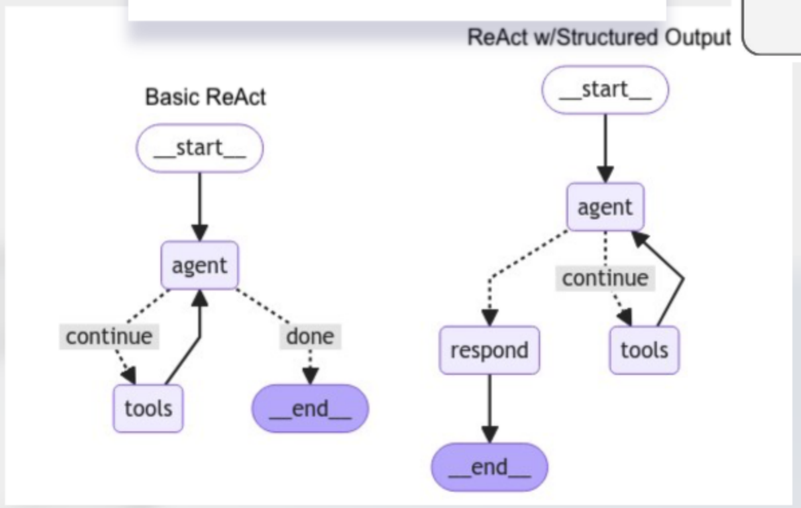


HISTORY OF AGENTIC AI

Static Multi Agents



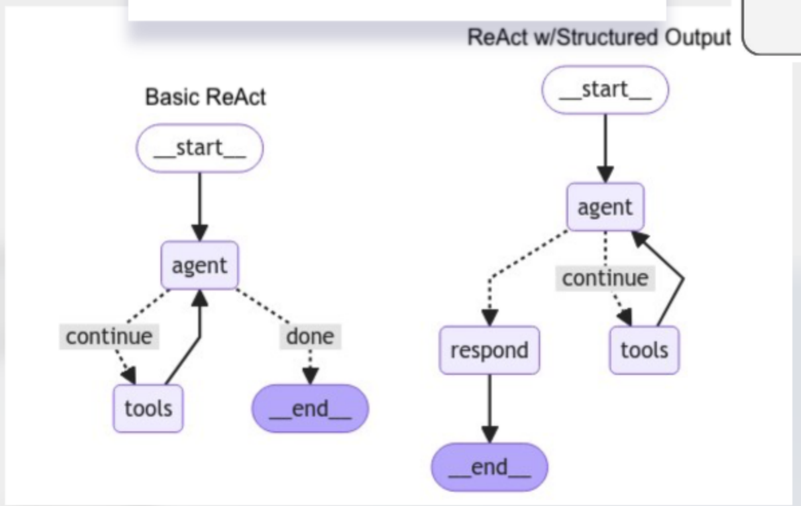
Single Agent



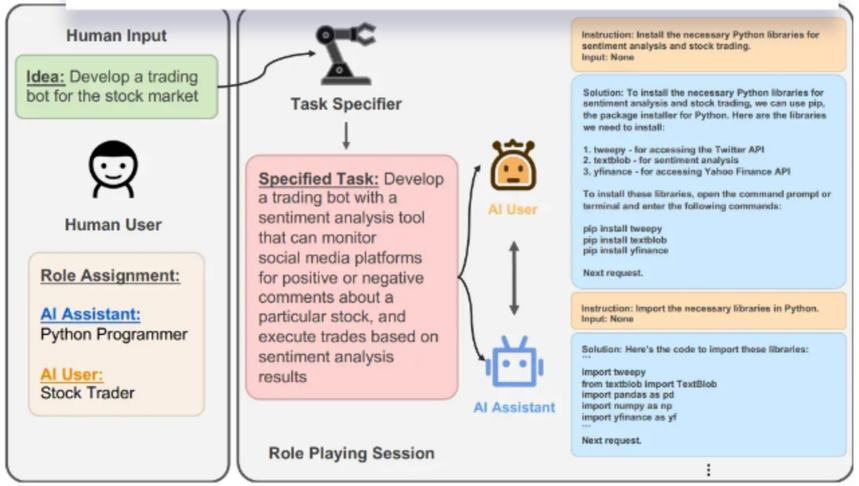
정해진 Role에 따라
Agent가 역할 놀이(Role playing)을 하는 방식

HISTORY OF AGENTIC AI

Single Agent

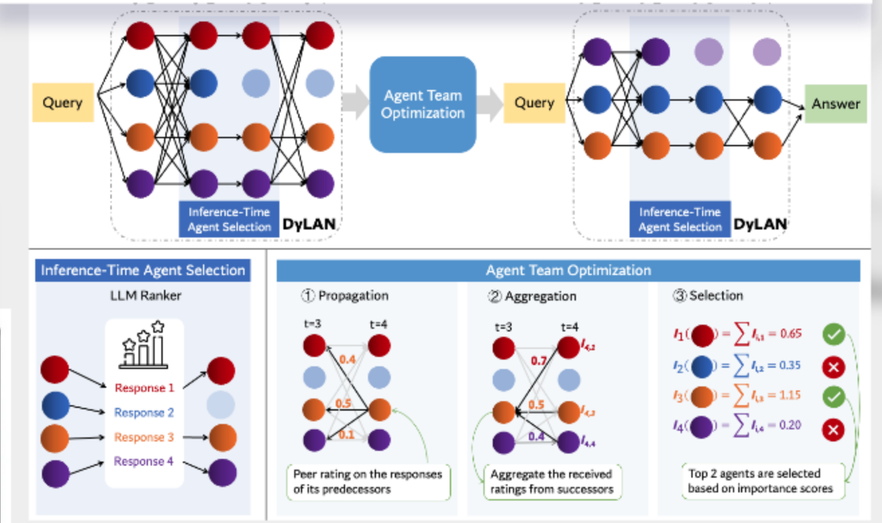


Static Multi Agents



여러 에이전트(agent)가
역할(role)을 배정받아 역할 놀이(role-playing) 방식으로
상호작용하면서 과제를 해결하도록 설계

Dynamic Multi Agents



LLM 기반 에이전트를 동적으로 선택하고,
그들 사이 통신 구조를 동적으로 조정하면서 협업하도록
설계된 프레임워크

ROLE DESIGN IN LLM-BASED ARCHITECTURE

MetaGPT

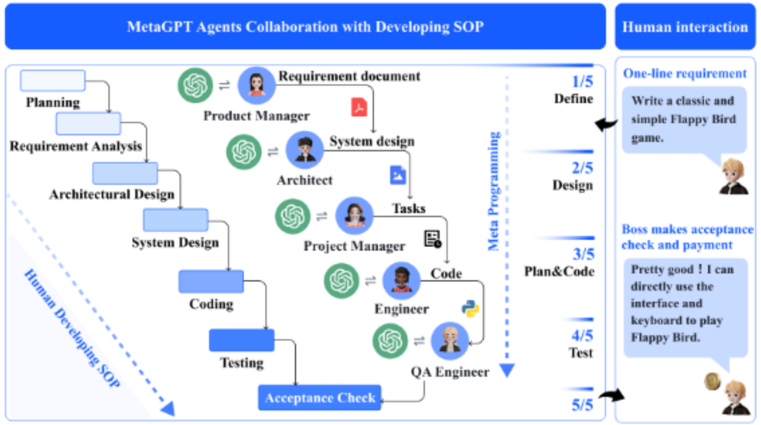
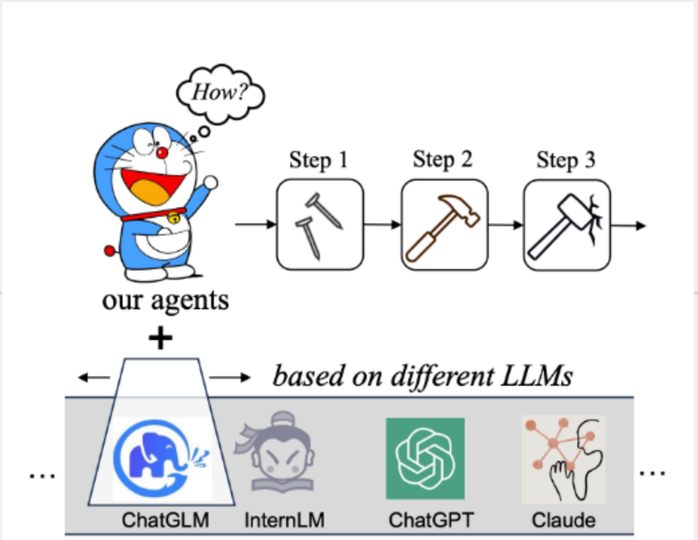


Figure 1: **The software development SOPs between MetaGPT and real-world human teams.** In software engineering, SOPs promote collaboration among various roles. MetaGPT showcases its ability to decompose complex tasks into specific actionable procedures assigned to various roles (e.g., Product Manager, Architect, Engineer, etc.).

인간이 사용하는 표준 운영 절차 (SOP: Standard Operating Procedure)를 에이전트 워크플로우에 "프롬프트 시퀀스" 형태로 내재화

FIXED-ROLE

TPTU

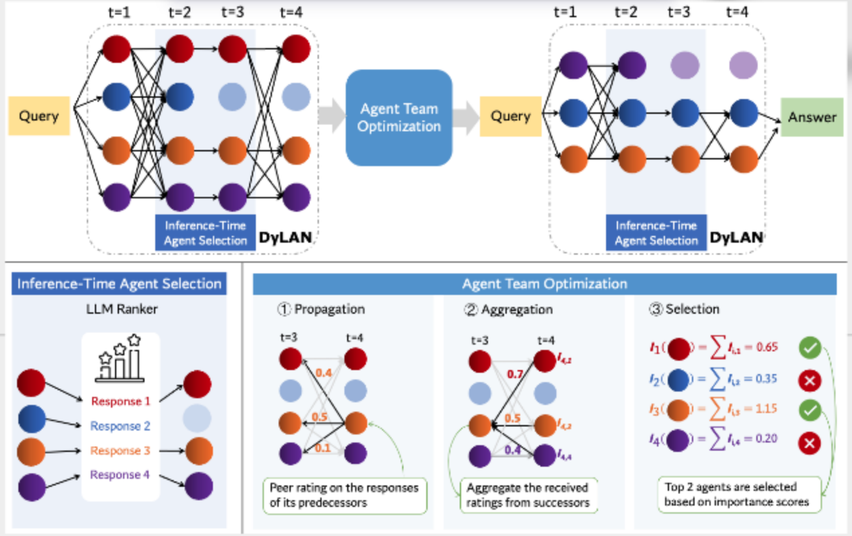


"텍스트를 생성하는 것"에서 벗어나, "임무를 분해하고, 어떤 도구를 쓸지 결정하고, 도구를 호출하고, 중간 결과를 관리하면서 완성해 나가는 에이전트"

TOOL-PLANNING

TOOL USAGE

DyLAN



- 2단계 방식 (Two-stage paradigm)
  - 팀 최적화 (Team Optimization) 단계: 후보 에이전트 집합 중에서 주어진 작업에 가장 기여할 가능성이 높은 에이전트를 선정
  - 작업 해결 (Task Solving) 단계: 선정된 에이전트들이 실제로 협업하면서 최종 답을 도출

- Agent Importance Score

- 이 점수를 기반으로 에이전트를 선택하거나 제외하는 방식→ 필요없는 에이전트 비활성화 (중간에 LLM 기반의 랭커(ranker) 역할을 두어, 어느 에이전트가 계속 참여할지 판단하게 함)

AGENT-PLANNING

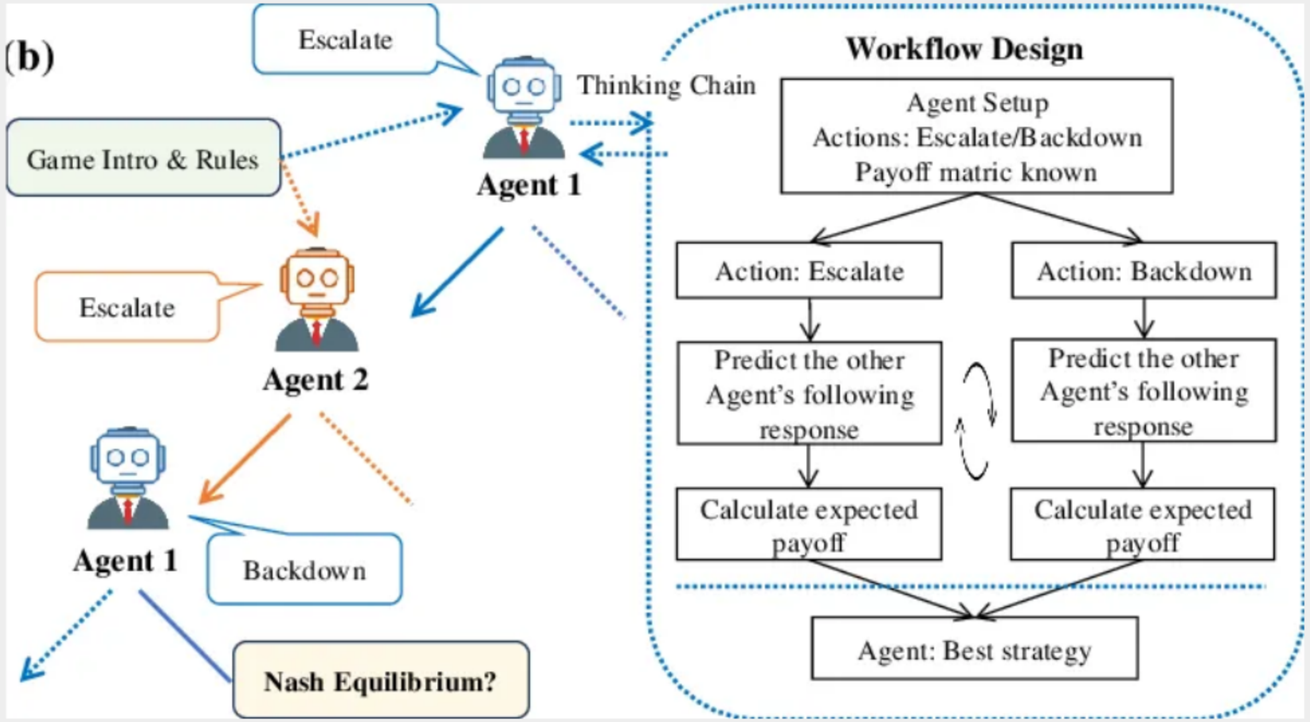
COOPERATION OPTIMIZATION STRATEGIES

중앙집권체제
Centralized framework

ex. AgentLaboratory, ScoreFlow,
WORKFLOW-LLM
Controller agent

지방분권체제
Decentralized approaches

Game-theoretic Workflow : LLM에게 단순한 "무엇을 해라" 지시를 넘어서,
"이런 사고 과정을 거쳐야 한다"라는 구조적 지침을 제공
(규칙 설명 → 전략 탐색 → 균형 분석 →...



CHALLENGES

PROBLEM 1

heavily depend on expert insight and manually-designed policies (rigid and predefined role space)

SOLUTION 1

Extensible Role Instantiation Mechanism

역할을 유연하게 확장 및 생성할 수 있는 구조

→ 다양한 환경에서도 self-organization & coordination 가능

PROBLEM 2

users' lack expertise in prompt engineering

SOLUTION 2

Prompt Engineering Module

사용자의 질의(query)를 자동으로 refinement하여 multi-agent collaboration의 효율성과 효과성 향상

HALO'S OBJECTIVE

HALO

워크플로우 탐색(workflow search)을 통해

다중 에이전트 시스템의 최적 reasoning 경로를 찾는 문제

워크플로우는 하위 과업 (sub tasks)로 구성 $\{T_1, T_2, \ldots, T_K\}$

각 하위 과업은 역할별 특화 LLM 기반 에이전트 집합 $\mathcal{A}_k$ $\{a_k^{(1)}, a_k^{(2)}, \ldots\}$

에 의해 수행된다.

OBJECTIVE

S(워크플로우 공간)는 "가능한 모든 과업–에이전트 조합의 집합"이며,

HALO는 주어진 Query를 바탕으로 이 중 가장 효과적인 reasoning workflow를

찾아내는 것을 목표

$$\mathcal{W}^* = \arg\max_{\mathcal{W} \in \mathcal{S}} \ \mathrm{Value}(\mathcal{Q}, \mathcal{W})$$

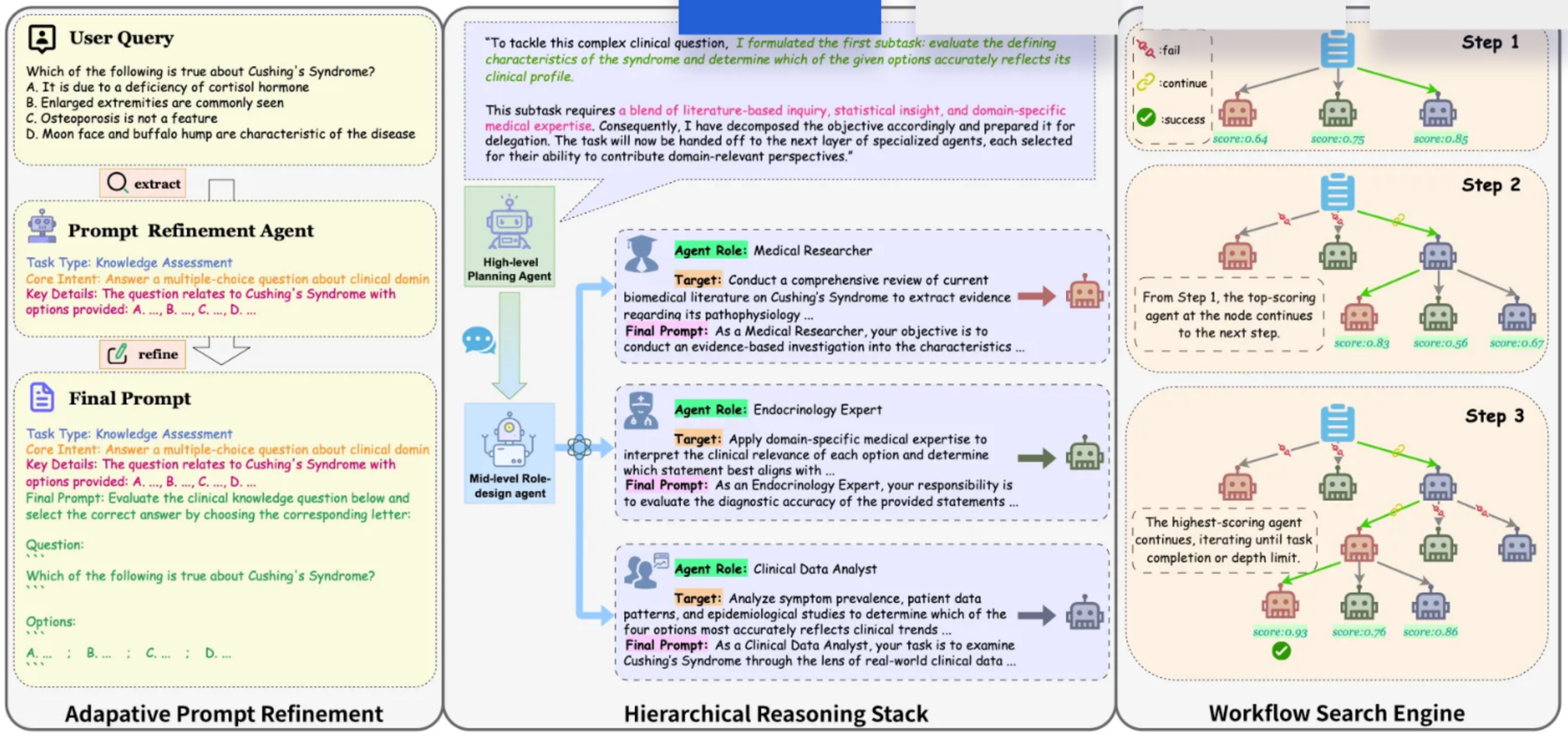
HALO Framework

Adaptive Prompt Refinement module

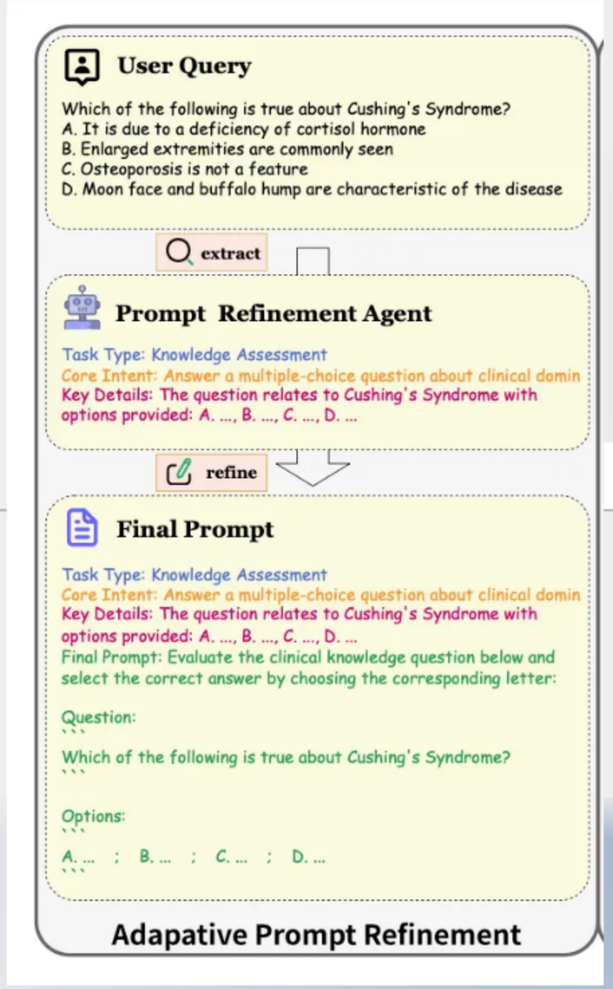
HIGH — PLANNING AGENT

MID — ROLE-DESIGN AGENTS

LOW — INFERENCE AGENT



ADAPTIVE PROMPT REFINEMENT



AGENT P1

Task Parser Agent

$\mathcal{F} = \mathcal{P}_1(\mathcal{Q}) = (\mathcal{T}, \mathcal{I}, \mathcal{D})$

F → a Global semantic context throughout the entire reasoning flow

1) task type T
2) core intent I
3) key details D

AGENT P2

Prompt Template Agent

$\mathcal{Q}_0 = \mathcal{P}_2(\mathcal{Q}, \mathcal{F})$

Qo : task description 재구성, clear reasoning objectives, bounded input conditions. explicit output format

AGENT P3

Prompt Optimization Agent

slow thinking prompting strategies, tool calling instructions, optimized prompt

AGENT P4

Prompt Generator Agent

Sythesizes the optimized structure

HIERARCHICAL REASONING STACK

HIGH

PLANNING AGENT

$$T_k = \mathcal{A}_{\text{plan}}(\mathcal{Q}^*, \mathcal{F}, H_{k-1})$$

앞선 refined Query와 전역 구조화된 과업 표현F을 입력 받으면, 과업들을 세분화

이때, 전체 과업을 한번에 세분화하지 않고, 단계적(step-wise) 전략을 사용하여 한번에 하나의 하위 과업만 생성하고, 이전 하위 과업들의 수행 이력 Hk-1 을 바탕으로 decomposition policy로 점진적 갱신

MID

ROLE-DESIGN AGENTS

하위 에이전트 생성

$$\mathcal{A}_k = \{a_k^{(1)}, a_k^{(2)}, \dots\}$$

역할별 시스템 프롬프트 생성

$$a_k^{(i)} = \mathcal{A}_{\text{role}}(T_k, \mathcal{Q}^*, \mathcal{F}) \Rightarrow \rho_k^{(i)}$$

LOW

INFERENCE AGENT

협력적 추론 수행

$$\mathcal{Y}_k = \{y_k^{(1)}, y_k^{(2)}, \dots\} = \mathcal{A}_k(T_k, \mathcal{Q}^*, \mathcal{F})$$

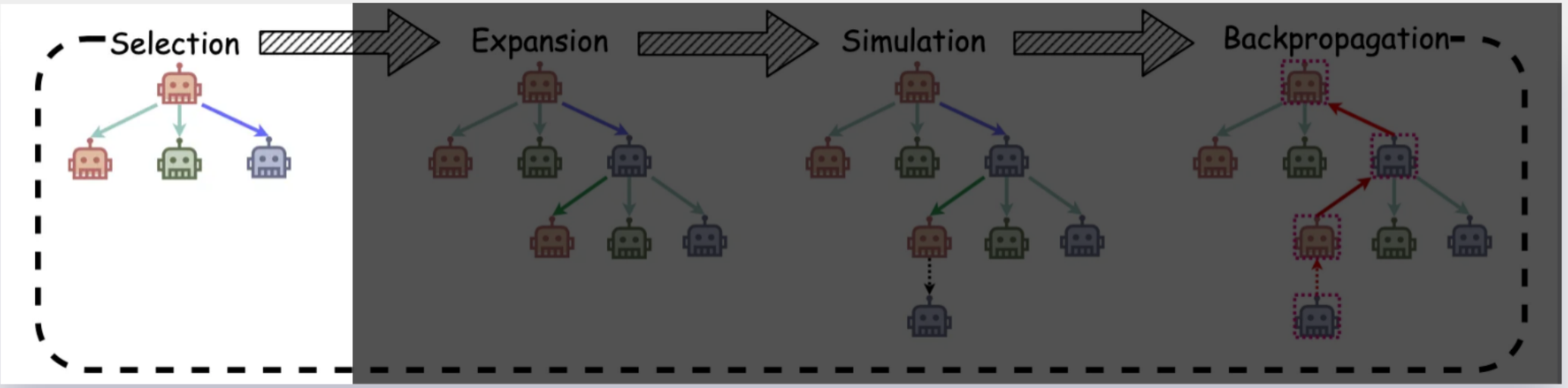
Tk,Q*, F를 받은 Inference agent는 협력적 추론을 수행하고 이에 맞는 집합 Yk라는 중간 산출물을 생성

조기 종료(early-stopping)

HALO는 수행 이력 Hk에 포함된 하위 과업 중 66% 이상이 동일한 일관된 답변 Y^에 도달했을 경우, 추론 과정을 조기에 종료

INFERENCE -WORKFLOW SEARCH ENGINE

MCTS Paradigm



SELECTION

UCT는 "성과가 좋거나, 아직 많이 시도하지 않은 노드"를 우선적으로 선택하는 알고리즘 !

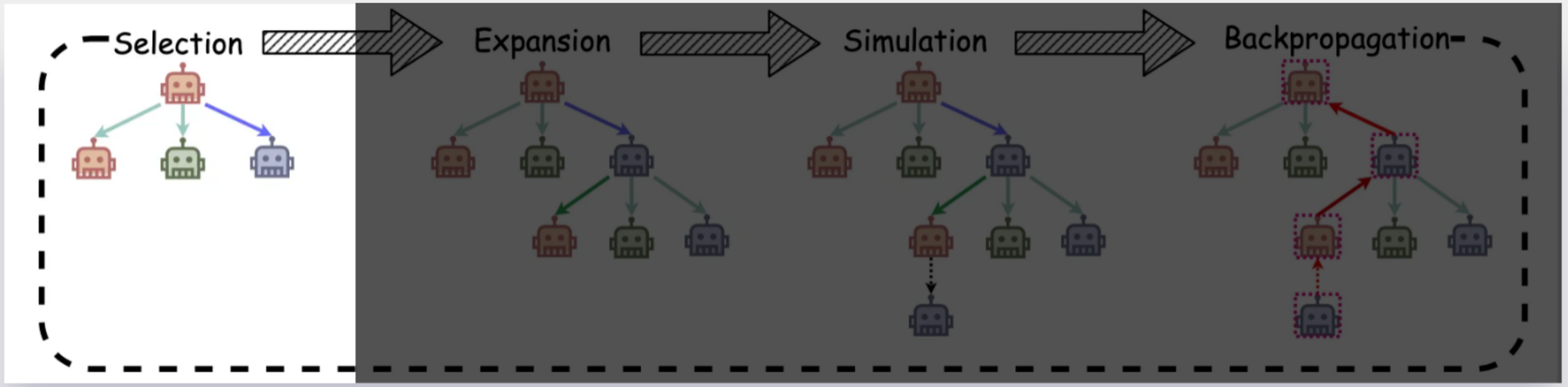
$$\mathrm{UCT}(a_k^{(i)}) = \frac{v_k^{(i)}}{n_k^{(i)}} + \alpha\sqrt{\frac{\log N}{n_k^{(i)}}}$$

평균 보상값(Vk : the score value of agent)

;; 지금까지의 보상이 큰 경우 -> 활용(exploitation)

INFERENCE –WORKFLOW SEARCH ENGINE

MCTS Paradigm



SELECTION

UCT는 "성과가 좋거나, 아직 많이 시도하지 않은 노드"를 우선적으로 선택하는 알고리즘 !

$$\mathrm{UCT}(a_k^{(i)}) = \frac{v_k^{(i)}}{n_k^{(i)}} + \alpha\sqrt{\frac{\log N}{n_k^{(i)}}}$$

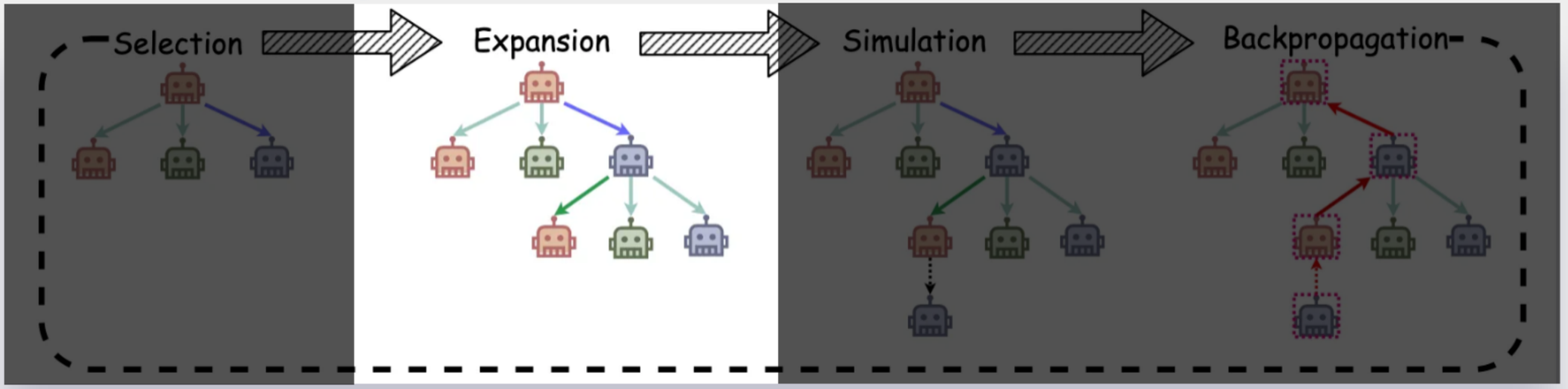
전체 탐색 횟수

현재 노드 K의 방문 횟수

현재 노드의 방문 횟수가 적을수록 값이 커짐 → 탐험(exploration)

INFERENCE –WORKFLOW SEARCH ENGINE

MCTS Paradigm



EXPANSION

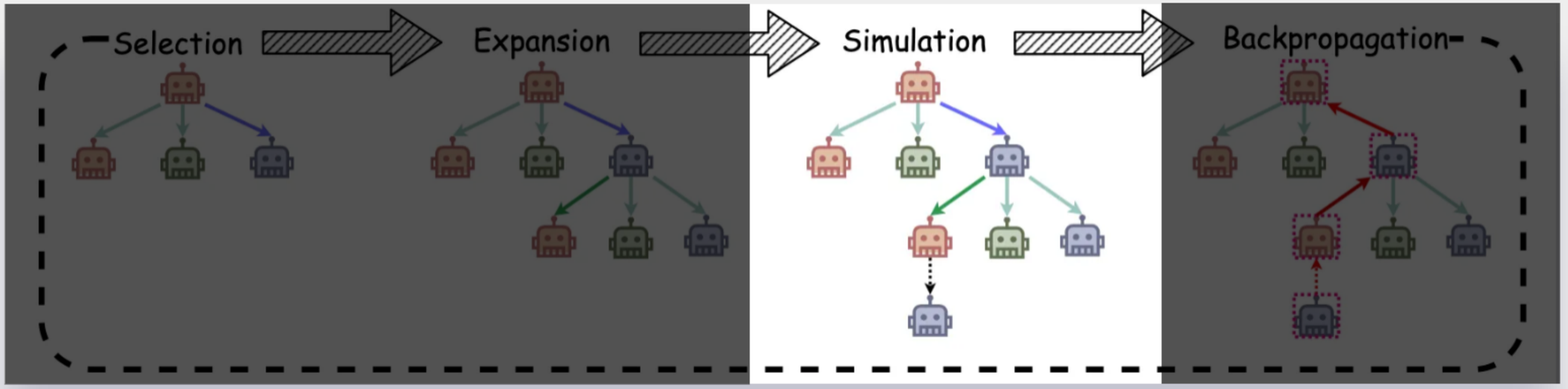
$a_k^{(i)}$ selected agent

$a_k^{(i+1)}$ child agent

role-specific agent

INFERENCE -WORKFLOW SEARCH ENGINE

MCTS Paradigm

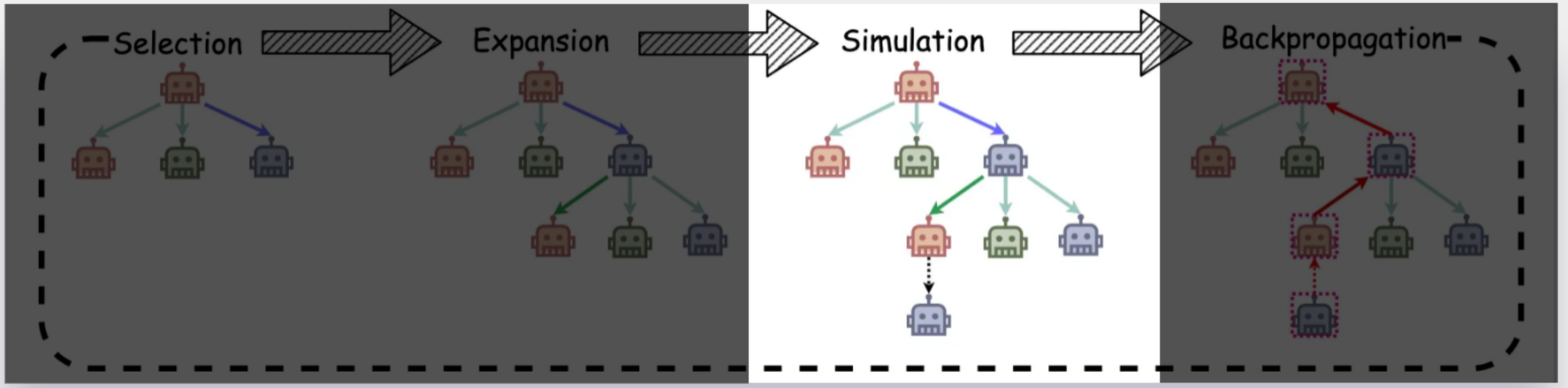


SIMULATION

시뮬레이션을 위해 expansion한 agent 각각의 실제가 아닌 가상 agent를 만들어서
중간 산출물(intermediate output) yk를 만든 다음,
두개의 보조 에이전트에 의해 평가됨 (judging agent & scoring agent)

INFERENCE –WORKFLOW SEARCH ENGINE

MCTS Paradigm



SIMULATION

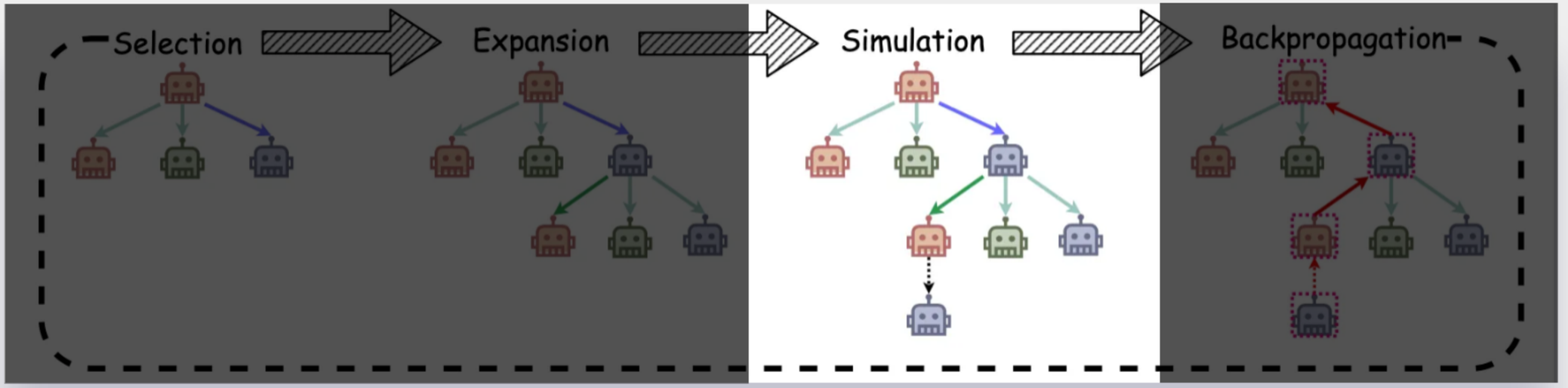
1. 가상 에이전트 생성

"~(틸드)"가 붙은 에이전트들은 실제가 아닌 가상(simulated) 에이전트이며,
HALO가 "만약 이 방향으로 진행한다면 어떤 결과가 나올까?"를 탐색하기 위해 사용하는 것

$$\tilde{a}_k^{(i+2)}, \tilde{a}_k^{(i+3)}, \cdots$$

INFERENCE -WORKFLOW SEARCH ENGINE

MCTS Paradigm



SIMULATION

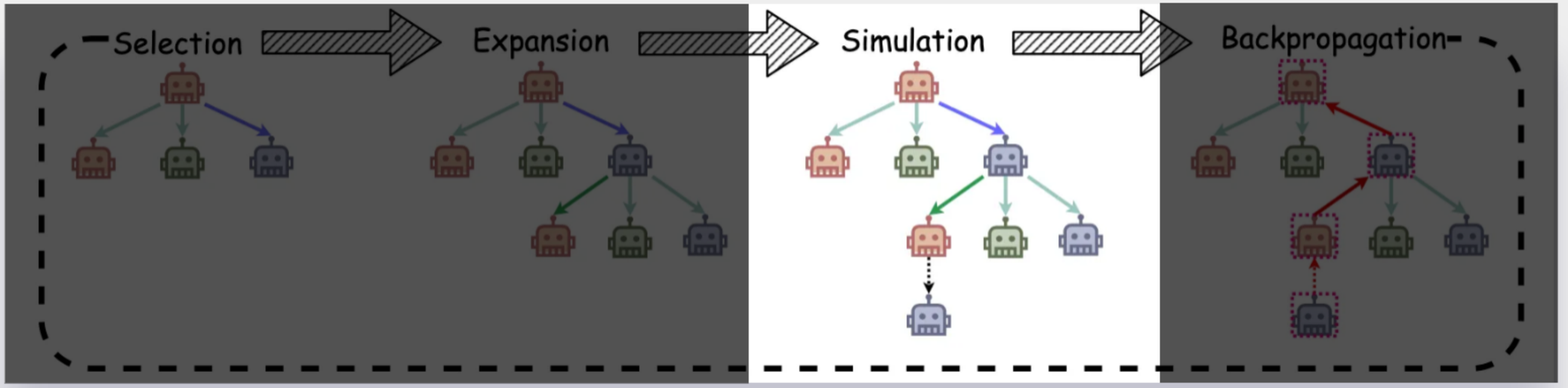
2. 중간 산출물 생성

각 시뮬레이션 에이전트 a~k(j)는 하위 과업 Tk, 정제된 프롬프트 Q*, 전역 구조화된 과업 표현 F을 입력으로 받아 중간 산출물(intermediate output) 를 생성

$$\tilde{y}_k^{(j)} = \tilde{a}_k^{(j)}(T_k, \mathcal{Q}^*, \mathcal{F})$$

INFERENCE –WORKFLOW SEARCH ENGINE

MCTS Paradigm



SIMULATION

3. 결과 평가: 두 보조 에이전트

가상 산출물y~k(j)은 **두 개의 보조 에이전트(auxiliary agents)**에 의해 평가

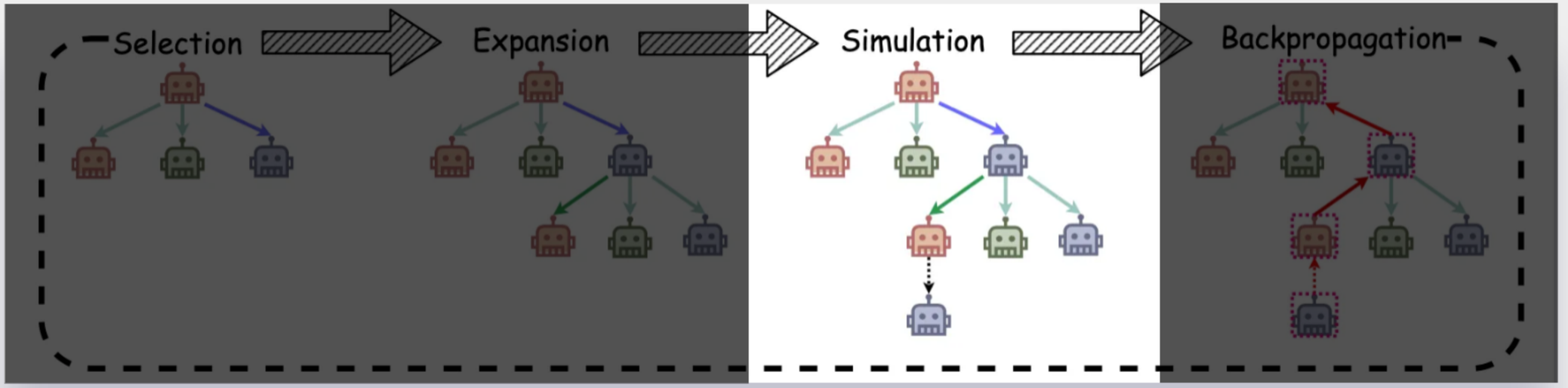
1) Judge agent $\tilde{\ell}_k^{(j)} \in \{success, fail, continue\}$

현재 하위 과업이 완료되었는지를 나타내기 위해 상태 레이블(status label)을 부여

이 값은 성공(success), 실패(fail), 계속 진행(continue) 중 하나

INFERENCE –WORKFLOW SEARCH ENGINE

MCTS Paradigm



SIMULATION

3. 결과 평가: 두 보조 에이전트

가상 산출물y~k(j)은 **두 개의 보조 에이전트(auxiliary agents)**에 의해 평가

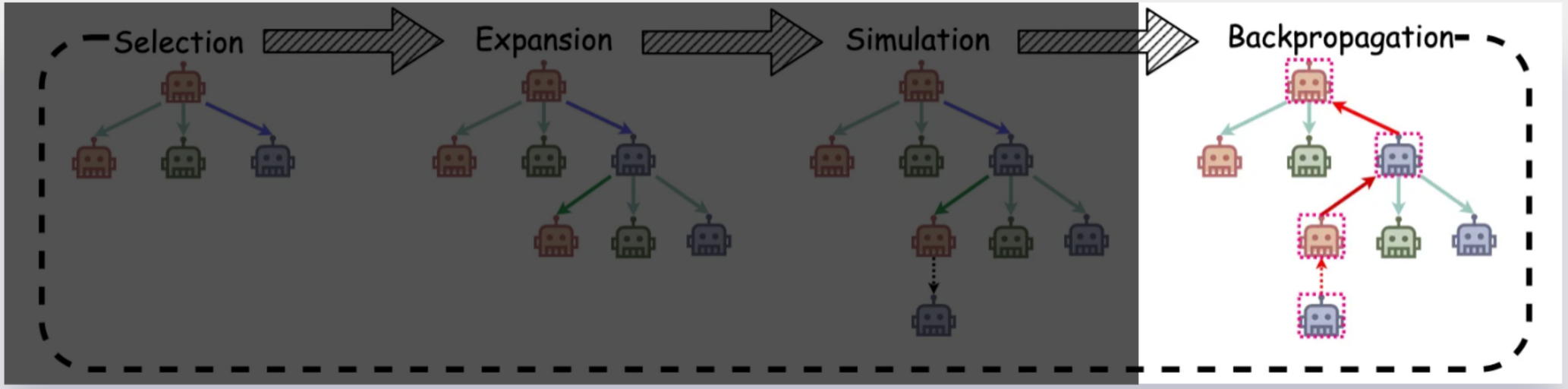
2) scoring agent $\tilde{v}_k^{(j)} \in [0, 1]$

생성된 산출물의 효과성(effectiveness)을 수치적으로 평가하기 위해

품질 점수(quality score) 을 계산

INFERENCE –WORKFLOW SEARCH ENGINE

MCTS Paradigm



BACKPROPAGATION

reward signal adjustment mechanism based on the judgement outcome

상태 레이블에 따른 보상/패널티 값

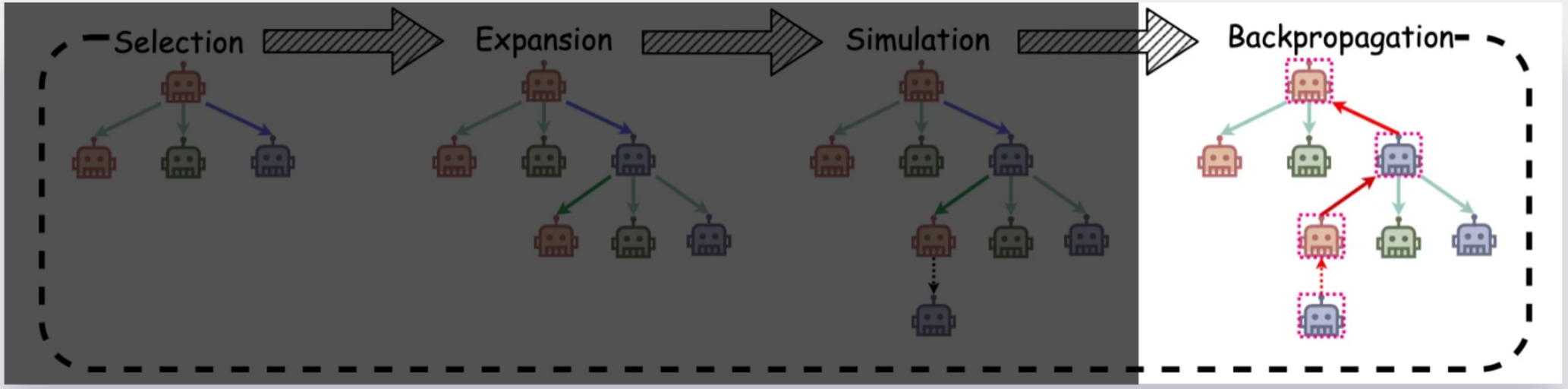
$$v_k^{(i)*} = \frac{v_k^{(i)} \cdot n_k^{(i)} + \lambda(\tilde{\ell}_k^{(j^\dagger)}) + \sum_{\tilde{a}_k^{(j)} \subset \mathrm{Child}(a_k^{(i)})} \tilde{v}_k^{(j)}}{n_k^{(i)} + |\mathrm{Child}(a_k^{(i)})|}$$

$\tilde{\ell}_k^{(j)} \in \{succcss, fail, continuc\}$

보상 계수 λ: judge agent에 의한 레이블에 영향 계수로 시뮬레이션 결과에 대한 보상 또는 패널티 반영

INFERENCE –WORKFLOW SEARCH ENGINE

MCTS Paradigm



BACKPROPAGATION

reward signal adjustment mechanism based on the judgement outcome

상태 레이블에 따른 보상/패널티 값

$$v_k^{(i)*} = \frac{v_k^{(i)} \cdot n_k^{(i)} + \lambda(\tilde{\ell}_k^{(j^\dagger)}) + \sum_{\tilde{a}_k^{(j)} \subset \mathrm{Child}(a_k^{(i)})} \tilde{v}_k^{(j)}}{n_k^{(i)} + |\mathrm{Child}(a_k^{(i)})|}$$

현재 노드의 기존 점수 × 노드 방문횟수

시뮬레이션 노드의 품질 점수의 합

자식 수 + 기존 방문 수

: 가중 평균(weighted average)

$$\tilde{v}_k^{(j)} \in [0, 1]$$

EXPERIMENTS

EXPERIMENTAL SETUP

1. Code generation : HumanEval dataset
(164 Python programming problems and units test),
평가지표- pass@1 metric (code accuracy)

2. General reasoning : MMLU dataset
(57 subjects with 15,908 questions in multiple-choice format with four options),
평가지표 - accuracy, 13% sampling

3. Arithmetic reasoning: : MATH dataset (12,500 math problems- 7 subareas with 5 diffculty levels)
500 math problems sampling, 평가 - accuracy

HALO SETUP

1. LLM: GPT -4o
2. random seed : 10
3. temperature : 0.8
4. max tokens : 2048
5. same number of few-shot examples
6. code interpreters as tools in the code generation task

MAIN RESULT

인지적 과부하(cognitive overload) 해결

ReAct는 하나의 에이전트가 계획 수립, 추론, 반성(reflection)을 동시에 수행해야 하지만,
HALO는 이 과정을 과업 분해(task decomposition), 역할 할당(role instantiation),
하위 과업 추론(subtask inference)으로 분리하여 에이전트 간 역할을 분담

HALO는 ReAct 대비 다음과 같은 개선

- HumanEval pass@1: 69.1% →95.2% (+26.1%)
- MMLU 정확도: 57.6% → 81.6% (+24.0%)
- MATH 정확도: 29.2% → 58.9% (+29.7%)

Table 2: Performance comparison of HALO against competitive baselines across three benchmarks. Metrics include $pass@1$ (%) for HumanEval, $accuracy$ (%) for MMLU as well as MATH, and $Avg.$ (%) for the mean performance over three runs. All methods are executed with GPT-4o.

Baseline Type	Method	Structure	Benchmarks			Avg.
			HumanEval	MMLU	MATH	
Single-agent	ReAct [11]	Monolithic Sequential Reasoning Flow	69.1	57.6	29.2	52.0
Static MAS	CAMEL [12]	Feedback Triplet	72.4	64.3	31.9	56.2
	LLM-Debate [43]	Fully Connected Bipartite	73.7	66.3	32.6	57.5
Dynamic MAS	DyLAN [21]	DAG + Feedback Loop	81.7	70.1	35.2	62.3
	AgentVerse [33]	Hierarchical Tree	75.2	67.5	34.4	59.0
	ADAS [44]	Search-Based Dynamic Graph Structure	82.4	72.8	36.9	64.0
	HALO (Ours)	**Hierarchical Structure + MCTS**	**95.2**	**81.6**	**58.9**	**78.6**

MAIN RESULT

적응형 에이전트 생성과 탐색 기반 워크플로우

동적 MAS(예: DyLAN, AgentVerse, ADAS)는
세밀한 역할–과업 정렬(fine-grained alignment)이 부족
반면 HALO는 MCTS 기반 탐색(Monte Carlo Tree Search)을 통해
실시간 피드백에 따라 적절한 역할을 동적으로 생성하고, 실행 경로를 반복적으로 개선

HALO는 ADAS 대비 다음과 같은 개선
- 평균 성능: HALO 78.6% vs. ADAS 64.0% (+14.6%)
- HumanEval: 82.4% → 95.2% (+12.8%)
- MMLU: 72.8% → 81.6% (+8.8%)
- MATH: 36.9% → 58.9% (+22.0%)

Table 2: Performance comparison of HALO against competitive baselines across three benchmarks. Metrics include $pass@1$ (%) for HumanEval, $accuracy$ (%) for MMLU as well as MATH, and $Avg.$ (%) for the mean performance over three runs. All methods are executed with GPT-4o.

Baseline Type	Method	Structure	Benchmarks			Avg.
			HumanEval	MMLU	MATH	
Single-agent	ReAct [11]	Monolithic Sequential Reasoning Flow	69.1	57.6	29.2	52.0
Static MAS	CAMEL [12]	Feedback Triplet	72.4	64.3	31.9	56.2
	LLM-Debate [43]	Fully Connected Bipartite	73.7	66.3	32.6	57.5
Dynamic MAS	DyLAN [21]	DAG + Feedback Loop	81.7	70.1	35.2	62.3
	AgentVerse [33]	Hierarchical Tree	75.2	67.5	34.4	59.0
	ADAS [44]	Search-Based Dynamic Graph Structure	82.4	72.8	36.9	64.0
	HALO (Ours)	**Hierarchical Structure + MCTS**	**95.2**	**81.6**	**58.9**	**78.6**

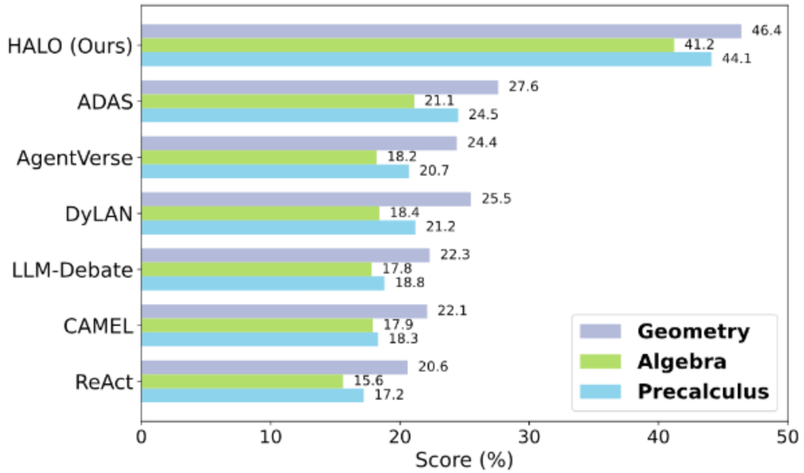
MAIN RESULT

MMLU의 abstract subjects과 MATH의 고난도 세부 영역(subareas)을 비교한 결과:

- MMLU(추상 과목): HALO 70.8% vs. ADAS 56.4% → +14.4% 향상
- MATH(고난도 영역): HALO 43.9% vs. ADAS 24.4% → +19.5% 향상

Table 3: Performance comparison on five abstract subjects selected from the MMLU dataset. Metrics are reported as *accuracy* (%) averaged over three runs.

Baseline Type	Method	Subjects				
		Abstract Algebra	College Physics	Formal Logic	High School Mathematics	Moral Scenarios
Single-agent	ReAct [11]	44.2	46.5	40.3	44.7	37.6
Static MAS	CAMEL [12]	50.4	54.7	49.8	51.9	44.2
	LLM-Debate [43]	51.1	54.6	48.3	52.8	45.1
Dynamic MAS	DyLAN [21]	52.9	60.4	51.2	58.8	49.6
	AgentVerse [33]	53.6	59.4	50.3	57.1	48.7
	ADAS [44]	55.7	62.4	52.6	60.0	51.3
	HALO (Ours)	**69.7**	**77.3**	**66.8**	**75.5**	**64.6**



ABLATION STUDY

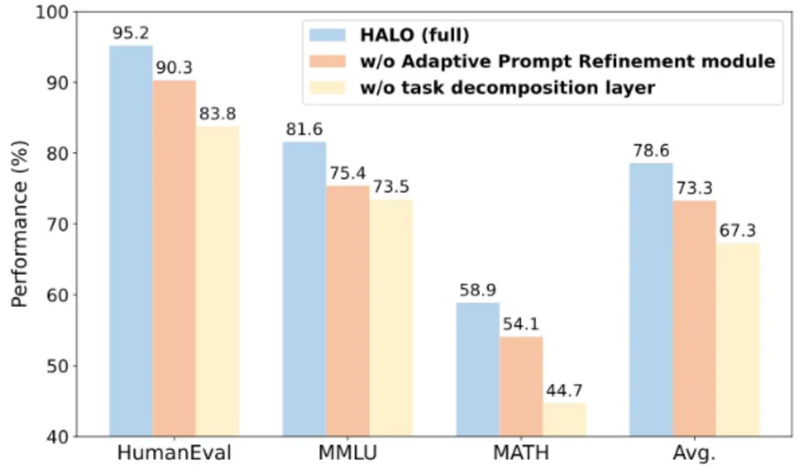


Figure 4: Ablation study of removing the Adaptive Prompt Refinement module and the high-level planning agent on GPT-4o across three benchmarks.

(1) Adaptive Prompt Refinement

performance drops by 5.3% on average, with MMLU suffering the most (81.6% →75.4%).

→ This result highlights the importance of structured prompt construction in enhancing task understanding and aligning reasoning trajectories with user intent.

(2) High-level planning agent

- this leads to an average performance drop of 11.3% across all benchmarks.
- Notable drop on HumanEval (95.2% →83.8%) and MATH (58.9% →44.7%) demonstrate that removing task

2025 연구실 세미나

THANK YOU

논문 세미나 발표

2025.10.15 강지윤