Eliciting Problem Specifications via Large Language Models

Center for Integrated Cognition

Robert Wray, James Kirk, John Laird

29 May 2024

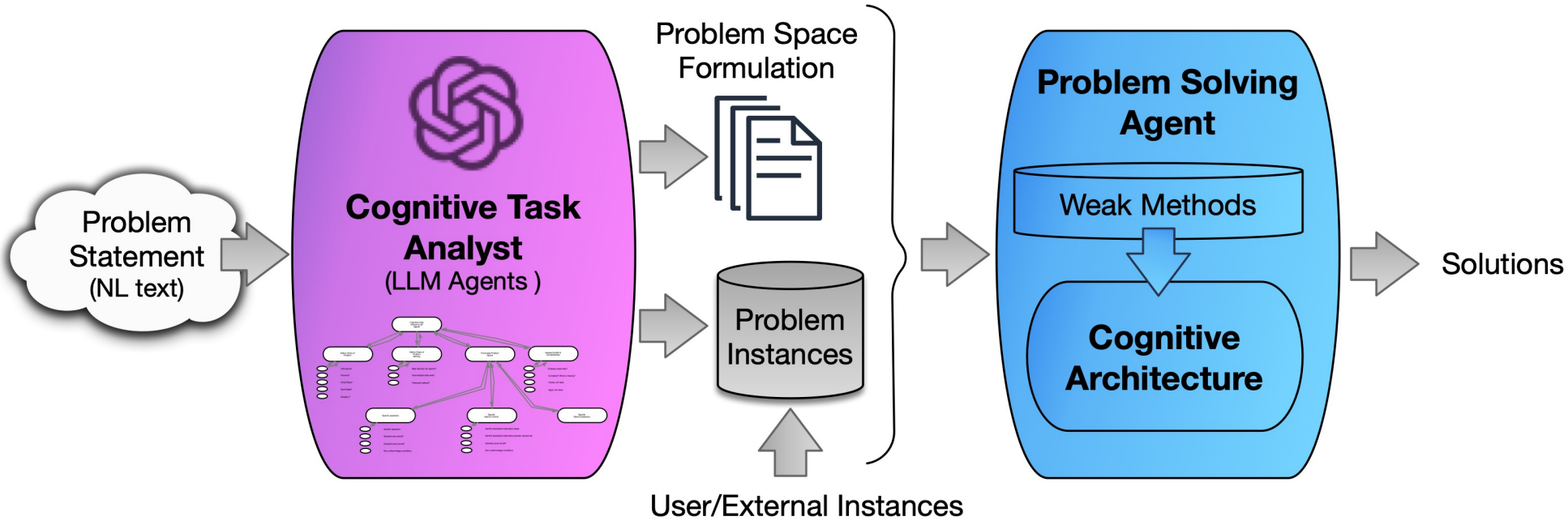
Accepted for presentation at ACS2024.
Preprint: https://arxiv.org/abs/2405.12147

A Longstanding Problem…

> [R]*epresentation of problems also raises a question of the locus of power.... we talk about the representation of a problem ... presumably a translation from its representation in some other form, such as natural language. These changes of the basic representational system are clearly of great importance to problem solving.... A suspicion arises that changes of the representation at this level ... might constitute a substantial part of problem solving.*
>
> Newell (1969)

- Implications
  - Limited autonomy
  - Potential limitations on claims/results
  - Labor/time-intensive agent development

LLMs as a Solution/Mitigation?



Problem Space Formulation

Problem Space:	A problem space consists a set of symbolic structures (the states of the space) and a set of operators over the space. Each operator takes a state as input and produces a state as output (although there may be other inputs and outputs as well). The operators may be partial (i.e., not defined for all states). Sequences of operators define paths that thread their way through sequences of states.
Problem:	A problem in a problem space consists of a set of initial states, a set of goal states, and a set of path constraints. The problem is to find a path through the space that starts at the initial state, passes only along paths that satisfy the path constraints, and ends at any goal state.

Newell (1980)

Example Formulation: Water Jugs

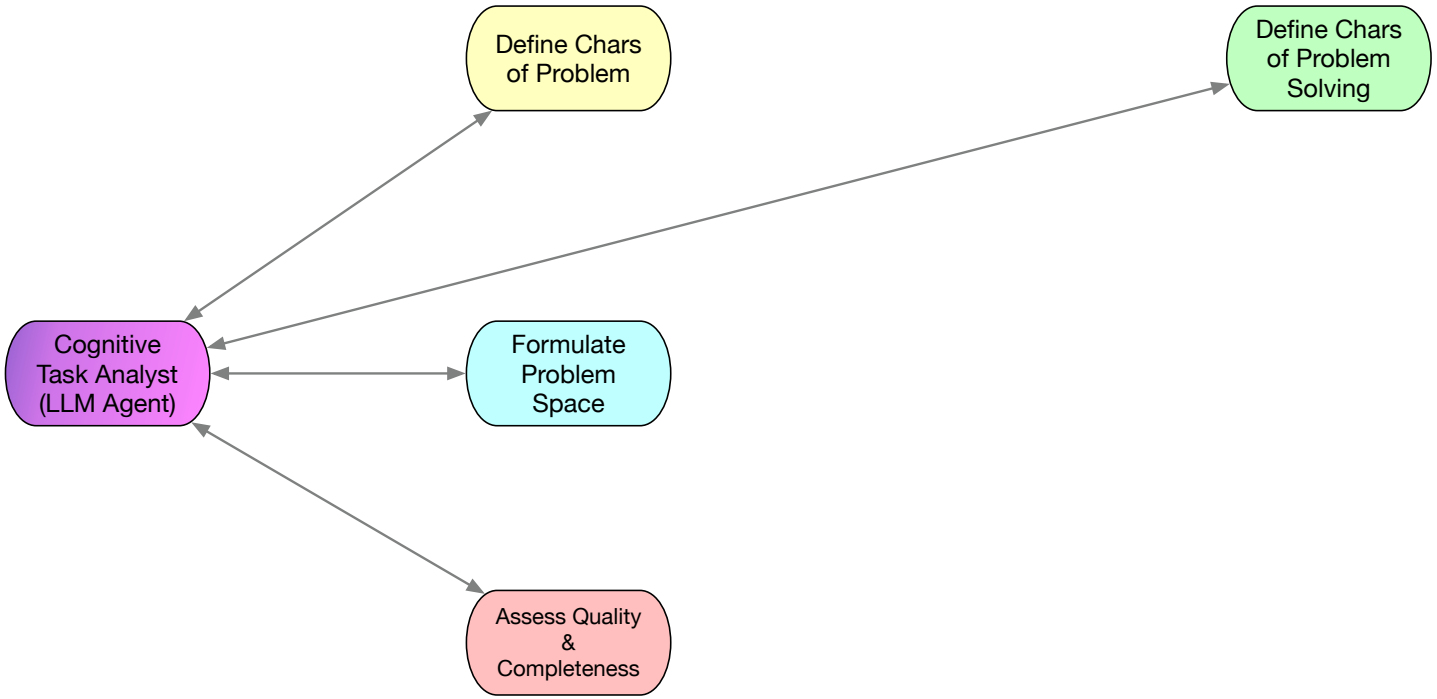
State:	Current volume of each jug.
Operators:	• **Fill** a given jug. • **Empty** a given jug. • **Pour** the contents of jug into the other until the source jug is empty, or the receiving jug is full.
Path Constraints:	• If the target jug has the goal amount, done. • Do not undo the previous action. • Do not take an action that will produce a state already on the path (avoid loops)

Laird & Newell (1983)

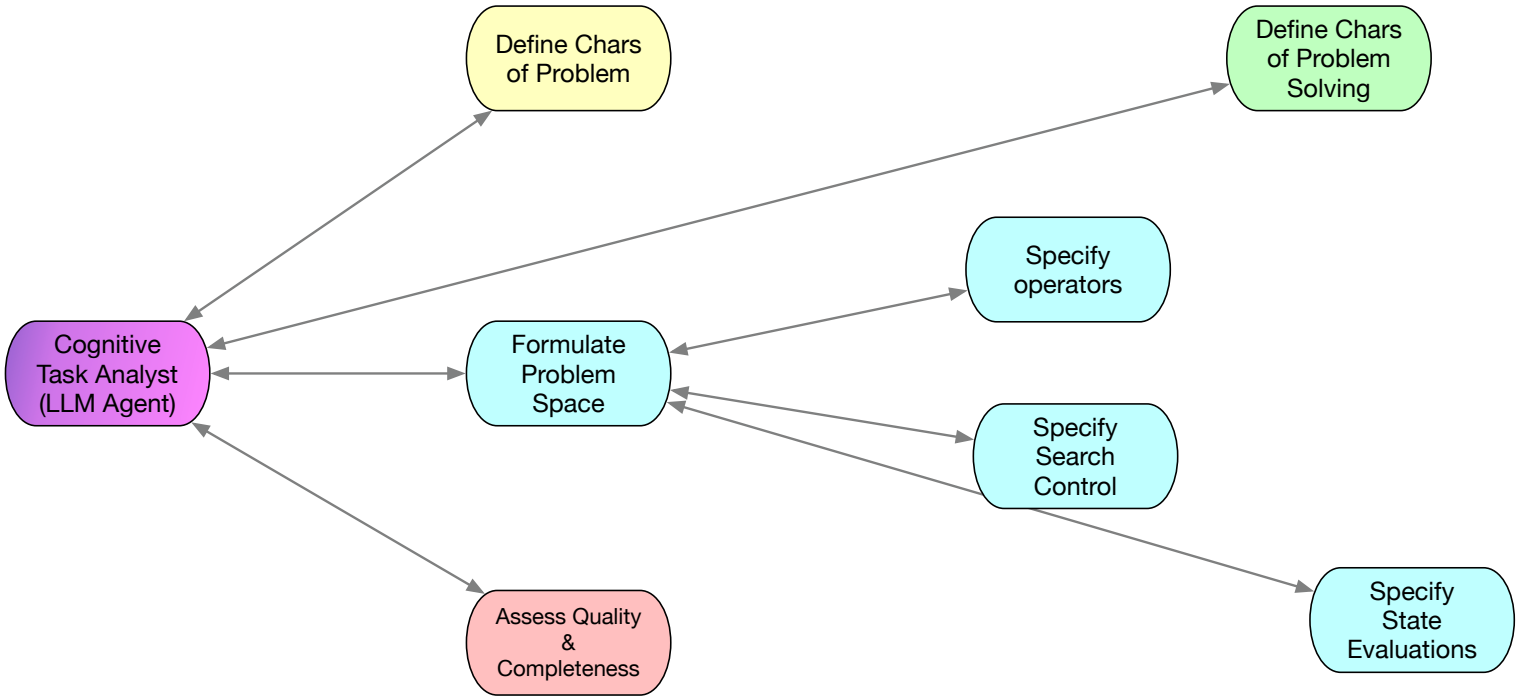
Alternatives

- LLM as code generator
  - Examples: Python generation, PDDL generation (e.g., Valmeekam, 2023)
  - LLMs excel, but humans are (largely) doing the problem space formulation
- LLM as problem solver
  - Examples: Chain of Thought, Tree of Thoughts, Graph of Thoughts, etc.
  - Limited evidence that LLM can scale problem solving (easy problems sometimes but generally still not ones with large search spaces)
- ITL with LLM integration
  - Requires human knowledge/understanding of the problem
  - (Ideally, this approach does not depend on online human input)

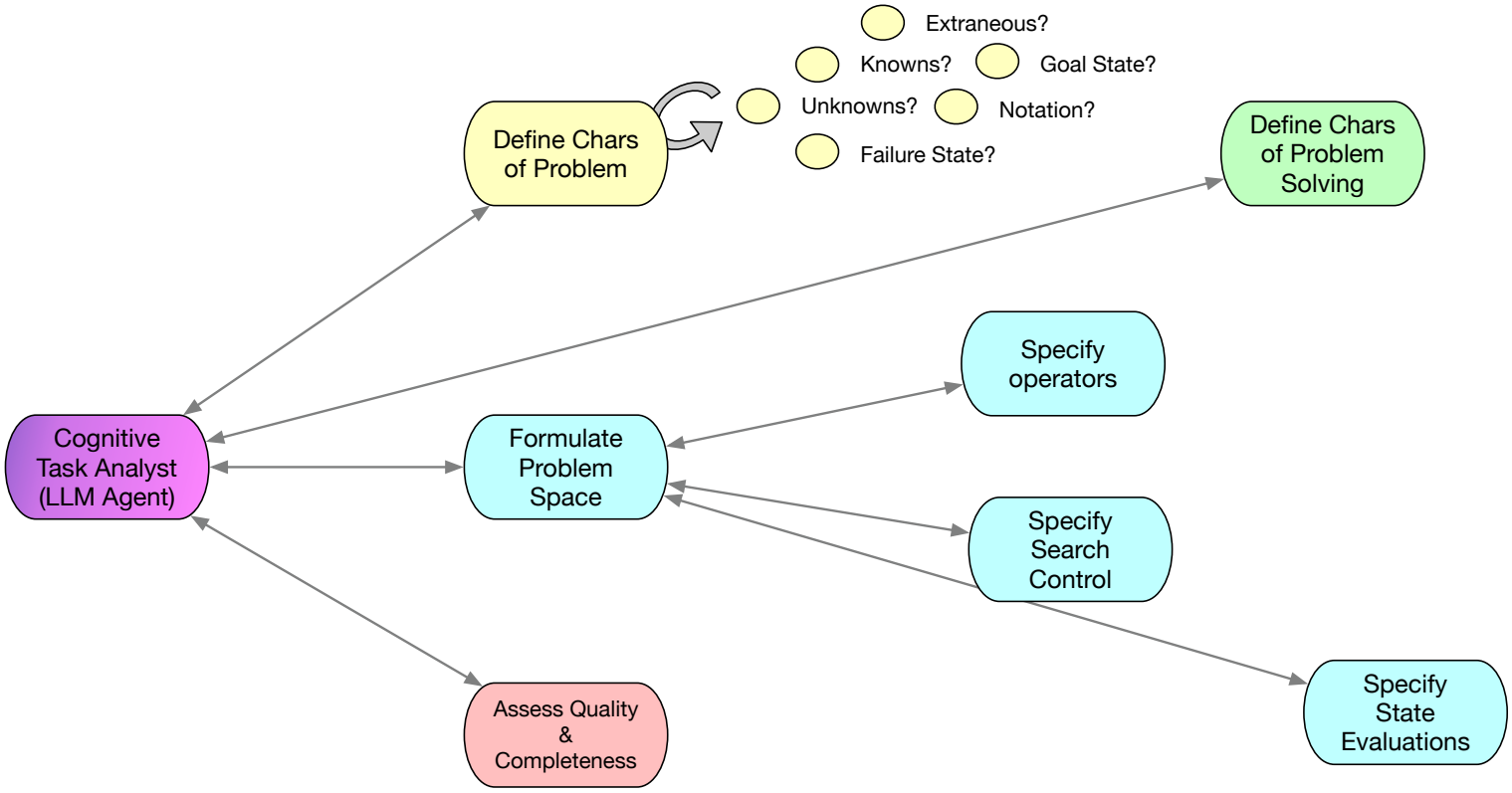
Cognitive Task Analysis Agent



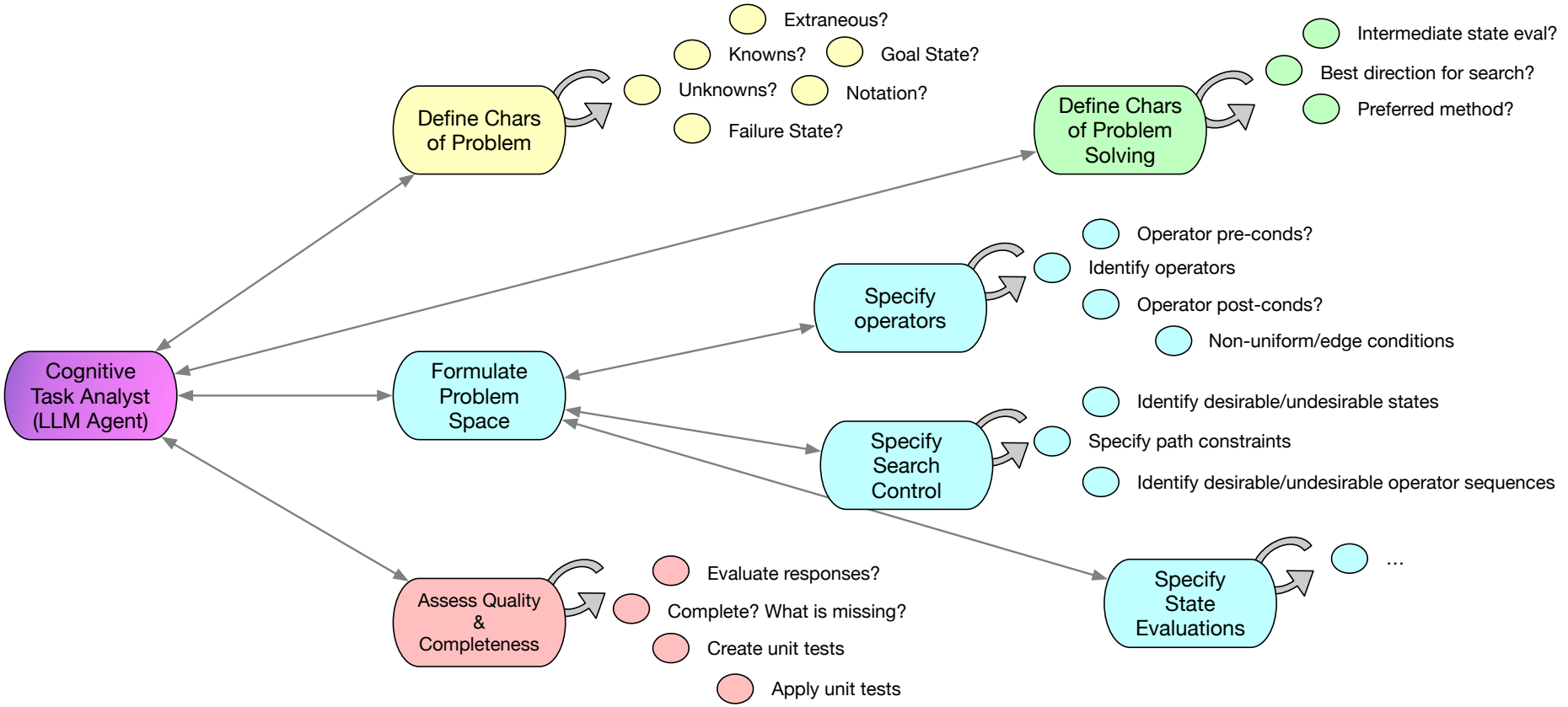
Cognitive Task Analysis Agent



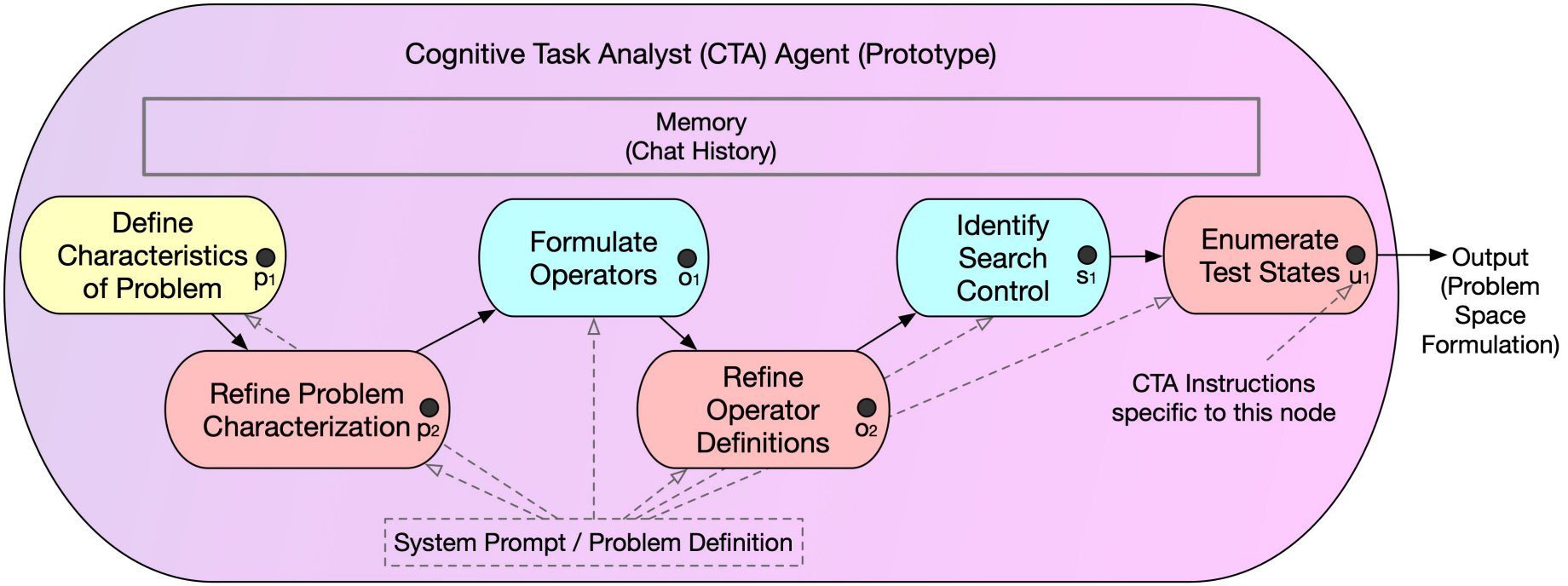
Cognitive Task Analysis Agent



Cognitive Task Analysis Agent



Feasibility Prototype & Exploration



- Implemented in LangGraph as a collection of LLM agents
- Only the problem definition prompt mentions a problem (no few shot)

Example Prompts

System Prompt:	You are an expert in cognitive task analysis [and] are helping design a reasoner that can solve many different instances of a class of problem. Problems are defined in 1-2 paragraphs. Focus each response on the most recent, specific question asked of you. Questions are designed to break the problem-space formulation into a set of discrete steps.
Problem Characterization (p_1):	For this response, focus on characterizing the problem itself. Use notation if/as possible to characterize the problem. What is the initial state? What is the final state? Are there illegal / impossible states that are not allowed?
Refinement (p_2):	You specialize in reviewing the work of other analysts. Improve the prior problem characterization. Are elements incorrect? Are elements missing? Focus on problem characterization (specifications of states) rather than what actions to take. Are elements poorly formed or ambiguous? For example, replace qualitative terms such as “less” or “more” with specific expressions that reflect quantitative values. [Produce] a revised characterization that reflects the analysis.

Feasibility Exploration: Water Jugs

- Three conditions:
  - Chain of Thought solver
  - One-shot, CoT problem space formulation
  - CTA Agent Prototype + Soar (default rules, iterative deepening search)
- Two LLMs
  - GPT 3.5
  - GPT 4
- Water-jug variations:
  - Familiar cases (6,8,20-step solutions)
  - Variations (units, number of jugs)
  - Analogue (“Space flux”)

Notation:

Fam / **V**ar / **A**na: (low,…,high) → Goal

F(3,5)→4 (*Die Hard* problem)

Overall Summary

- Models: GPT3.5 not good enough, needed GPT4
- Familiar cases: Often, CoT solver was sufficient (even 20-step)
- Mixed-units /Extra jugs: CTA Agent produced formulations solvable by Soar.
- Space flux/analog: Solved by CoT Solver (not sufficiently “disguised”?)
- CoT problem space formulation vs. CTA Agent:
  - CoT: Relatively poor outcomes
  - CTA Agent: Sufficient problem space formulation for all cases
    - Significant benefit from reflection/reformulation nodes
    - Ok but imperfect at eliciting search control knowledge
    - Not able to generate meaningful test cases

Reformulation Example

Initial Response:	**Transfer** (FHDx, FHDy): Transfers flucotone from FHDx to FHDy until FHDx is empty or FHDy is full. • Preconditions: FHDx is not empty, and FHDy is not full. • Postconditions: FHDx decreases by the transferred amount, and FHDy increases by the same amount, without exceeding FHDy's capacity.
Revised Response:	**Transfer**(FHDx, FHDy) • Preconditions: (FHDx > 0) AND (FHDy < $Capacity_{FHDy}$) • Postconditions: (FHDx' = max(0, FHDx - ($Capacity_{FHDy}$ - FHDy))), (FHDy' = min($Capacity_{FHDy}$, FHDy + FHDx)) Rationale: The original characterization did not explicitly account for the exact amount transferred. The revised postconditions now clearly define the new states of FHDx' and FHDy' after the transfer, ensuring that the transfer does not exceed FHDy's capacity or result in FHDx having a negative amount of flucotone.

Flucotone = "Space flux" (analogue to water); FHD = Flucotone holding device (analogue to a jug)

Impact of Search Control (Decisions)

Test Case	Min. Soln	with Failure Detection			without Failure Detection		
		No Learning	*During Learning*	*After Learning*	*No Learning*	*During Learning*	*After Learning*
$F(4,9) \rightarrow 6$	8	20,657	1357	9	80,434	2126	9
$F(3,5) \rightarrow 4$	6	3355	548	7	8072	1087	7
$F(9,17) \rightarrow 5$	20	86.9M	8201	21	600M+ (see note)	7147	21
$F(4q,9g) \rightarrow 6g$	6	3422	526	7	8071	801	7
$F(2,3,5) \rightarrow 4$	4	1599	908	5	1890	860	5

Failure detection for iterative deepening stops exploration of search branch when all jugs full OR all jugs empty

```
Soar 9.6.2 on warspite-mbp-21.lan at Fri Apr 26 15:42:23 2024

141 productions (76 default, 65 user, 0 chunks)
   + 64 justifications
                                        |  Computed
Phases:     Input   Propose   Decide   Apply    Output |    Totals
============================================================|==========
Kernel:     1.227   40.987 2120.402 21223.258  224.062  |  23609.937
============================================================|==========
Input fn:   0.573                             |     0.573
============================================================|==========
Outpt fn:                                0.000 |     0.000
============================================================|==========
Callbcks:   0.000    0.000    0.000    0.000 20790.667  |  20790.667
============================================================|==========
Computed-------------------------------------------------+----------
Totals:     1.800   40.987 2120.402 21223.258 21014.729  |  44401.177

Values from single timers:
 Kernel CPU Time:   23016.882 sec.
 Total  CPU Time:   44662.113 sec.

201000000 decisions (0.115 msec/decision)
968250902 elaboration cycles (4.817 ec's per dc, 0.024 msec/ec)
1045971566 inner elaboration cycles
228998128 p-elaboration cycles (1.139 pe's per dc, 0.101 msec/pe)
2960366332 production firings (3.057 pf's per ec, 0.008 msec/pf)
9532545581 wme changes (4766273938 additions, 4766271643 removals)
WM size: 2295 current, 2129.408 mean, 2751 maximum
```

Conclusions

Nuggets

- Demonstrated potential feasibility of automated problem space formulation
- Potentially big impact on cognitive systems research
  - Fewer “variables” in agent dev
  - Faster agent dev
  - More autonomous agent systems

Coal

- Single, simple, familiar domain
- Complete code generation is likely feasible but still a lot of work
- How might an autonomous agent leverage something like the CTA Agent?

Acknowledgments

This work was supported by the Office of Naval Research, contract N00014-21-1-2369. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Department of Defense or Office of Naval Research. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.