Cognitive Architectures for Language Agents

Theodore Sumers* **Shunyu Yao*** **Karthik Narasimhan** **Thomas L. Griffiths**
Princeton University
{sumers, shunyuy, karthikn, tomg}@princeton.edu

Abstract

Recent efforts have incorporated large language models (LLMs) with external resources (e.g., the Internet) or internal control flows (e.g., prompt chaining) for tasks requiring grounding or reasoning. However, these efforts have largely been piecemeal, lacking a systematic framework for constructing a fully-fledged *language agent*. To address this challenge, we draw on the rich history of agent design in symbolic artificial intelligence to develop a blueprint for a new wave of *cognitive* language agents. We first show that LLMs have many of the same properties as *production systems*, and recent efforts to improve their grounding or reasoning mirror the development of *cognitive architectures* built around production systems. We then propose Cognitive Architectures for Language Agents (CoALA), a conceptual framework to systematize diverse methods for LLM-based reasoning, grounding, learning, and decision making as instantiations of language agents in the framework. Finally, we use the CoALA framework to highlight gaps and propose actionable directions toward more capable language agents in the future.

1 Introduction

Despite revolutionizing natural language processing (NLP), large language models (LLMs; Vaswani et al., 2017; Brown et al., 2020; Devlin et al., 2019; Brown et al., 2020; OpenAI, 2023) have limited world knowledge and are not grounded to external environments. To address these shortcomings, recent methods have augmented (Mialon et al., 2023) LLMs with external resources such as memory stores (Guu et al., 2020) or pre-defined sequences of prompts to structure reasoning (Creswell et al., 2023; Wu et al., 2022b). Parallel work has grounded LLMs by placing them in a feedback loop with the environment, leading to an emerging field of *language agents*: interactive systems that use LLMs for sequential decision-making (Yang et al., 2023b; Wang et al., 2023b). While the earliest agents used the LLM to directly select actions (Ahn et al., 2022; Huang et al., 2022b), the latest generation uses a series of LLM calls to reason (Yao et al., 2022b) or read and write from internal memory (Park et al., 2023) to improve decision making. Leveraging complex internal processes, this latest generation of *cognitive* language agents are growing remarkably sophisticated (Wang et al., 2023a), yet we lack a conceptual framework to characterize or design them.

We introduce such a framework, drawing parallels with two ideas from the history of computing and artificial intelligence (AI): *production systems* and *cognitive architectures*. Production systems generate a set of outcomes by iteratively applying rules (Newell and Simon, 1972). They originated as string manipulation systems – an analog of the problem that LLMs solve – and were subsequently adopted by the AI community to define systems capable of complex, hierarchically structured behaviors (Newell et al., 1989). To do so, they were incorporated into cognitive architectures that specified control flow for selecting, applying, and even generating new productions (Laird et al., 1987; Laird, 2022; Kotseruba and Tsotsos, 2020).

We suggest a meaningful analogy between production systems and LLMs: just as productions indicate possible ways to modify strings, LLMs define a distribution over changes or additions to text. This suggests that the kinds of controls used with production systems might be equally applicable to LLMs, addressing aspects including memory, grounding, learning, and decision making, among others.

*Equal contribution, order decided by coin flip. Each person reserves the right to list their name first. A CoALA-based repo of recent work on language agents: `https://github.com/ysymyth/awesome-language-agents`.

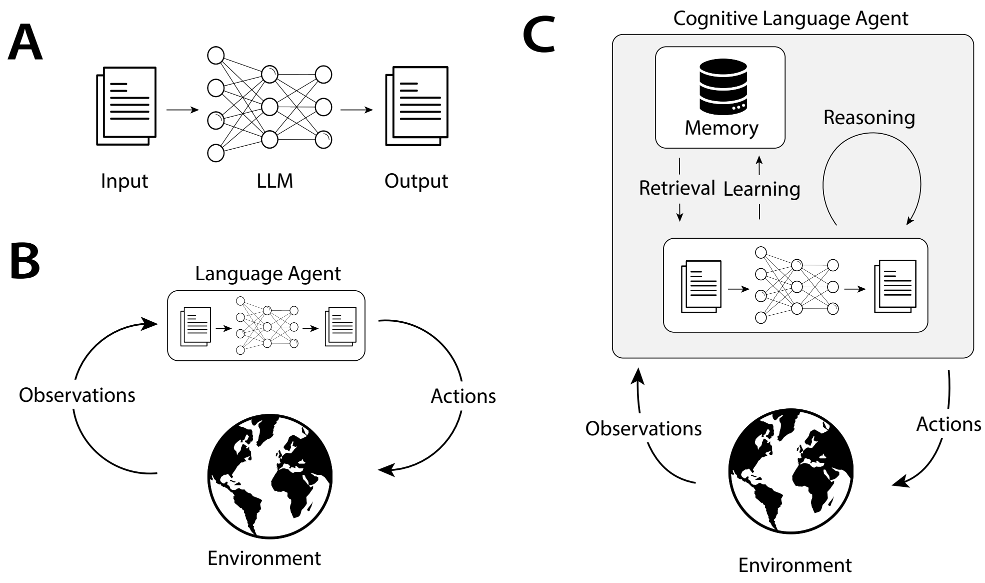

Figure 1: Different uses of large language models (LLMs). **A**: By default, an LLM takes text as input and outputs text. **B**: *Language agents* (Ahn et al., 2022; Huang et al., 2022c) place the LLM in a direct feedback loop with the external environment by transforming observations into text and interpreting outputs as choosing actions. **C**: *Cognitive* language agents (Yao et al., 2022b; Shinn et al., 2023; Wang et al., 2023a) additionally use the LLM to manage the agent's internal state via processes such as learning and reasoning. In this work, we propose a blueprint to structure such agents.

The plan for the rest of the paper is as follows. First, we introduce production systems and cognitive architectures in more detail. We then outline how language models are analogous to production systems, and use this connection to reconstrue prompting strategies as forms of control flow. We then introduce **Co**gnitive **A**rchitecture for **L**anguage **A**gents (CoALA), applying ideas from cognitive architectures to organize existing language agents. We highlight some of the opportunities this presents for designing new cognitive language agents, and close by considering the implications of this approach for future work.

2 Background: From Strings to Symbolic AGI

We first introduce production systems and cognitive architectures, providing a historical perspective on cognitive science and artificial intelligence: beginning with theories of logic and computation (Post, 1943), and ending with attempts to build symbolic artificial general intelligence (Newell et al., 1989).

2.1 Production systems for string manipulation

Production systems provide a basis for symbol manipulation. Intuitively, production systems consist of a set of rules, each specifying a precondition and an action. When the precondition is met, the action can be taken. The idea originates in efforts to characterize the limits of computation. Emil Post (Post, 1943) proposed thinking about arbitrary logical systems in these terms, where formulas are expressed as strings and the conclusions they license are identified by production rules (as one string "produces" another). This formulation was subsequently shown to be equivalent to a simpler string rewriting system. In such as system, we specify rules of the form

$$X\,Y\,Z \rightarrow X\,W\,Z$$

indicating that the string XYZ can be rewritten to the string XWZ. String rewriting plays a significant role in the theory of formal languages, corresponding to Chomsky's phrase structure grammar (Chomsky, 1956).

2.2 Control flow: From strings to algorithms

By itself, a production system simply characterizes the set of strings that can be generated from a starting point. However, they can be used to specify algorithms if we impose *control flow* to determine which productions are executed. For example, Markov algorithms are production systems with a priority ordering (Markov, 1954). The following algorithm implements division-with-remainder by converting a number written as strokes $|$ into the form $Q * R$, where Q is the quotient of division by 5 and R is the remainder:

$$\begin{aligned} *||||| &\quad\to\quad |* \\ * &\quad\overset{\bullet}{\to}\quad * \\ &\quad\to\quad * \end{aligned}$$

where the priority order runs from top to bottom, productions are applied to the first substring matching their preconditions when moving from left to right (including the empty substring, in the last production), and $\overset{\bullet}{\to}$ indicates the algorithm halts after executing the rule. For example, given the input 11, this would yield the sequence of productions $*||||||||||| \to |*|||||| \to ||*| \overset{\bullet}{\to} ||*|$ which is interpreted as 2 remainder 1. Simple productions can result in complex behavior − Markov algorithms can be shown to be Turing complete.

Production systems with control flow were popularized in the AI community by Allen Newell, who was looking for a formalism to capture human problem solving (Newell, 1967; Newell and Simon, 1972). Productions were generalized beyond string rewriting to logical operations: *preconditions* that could be checked against the agent's goals and world state, and *actions* that should be taken if the preconditions were satisfied. In their landmark book *Human Problem Solving* (Newell and Simon, 1972), Allen Newell and Herbert Simon gave the example of a simple production system to describe the operation of a thermostat:

$$\begin{aligned} (\text{temperature} > 70^\circ) \wedge (\text{temperature} < 72^\circ) &\quad\to\quad \text{stop} \\ \text{temperature} < 32^\circ &\quad\to\quad \text{call for repairs; turn on electric heater} \\ (\text{temperature} > 70^\circ) \wedge (\text{furnace off}) &\quad\to\quad \text{turn on furnace} \\ (\text{temperature} > 72^\circ) \wedge (\text{furnace on}) &\quad\to\quad \text{turn off furnace} \end{aligned}$$

Equipped with logical expressions that capture the internal state of the system, such production systems could be used to express classic search algorithms as well as more complex strategies that people use to solve challenging reasoning problems.

2.3 Cognitive architectures: From algorithms to agents

The success of production systems as cognitive models was paralleled by their increasing use in AI. Large production systems connected to external sensors, actuators, and knowledge bases required correspondingly sophisticated control flow. AI researchers defined "cognitive architectures" that mimicked human cognition – explicitly instantiating processes such as perception, memory, and planning (Adams et al., 2012) to achieve flexible, rational, real-time behaviors (Sun, 2004; Newell, 1980; 1992; Anderson and Lebiere, 2003). This led to applications from psychological modeling to robotics, with hundreds of architectures and thousands of publications (see Kotseruba and Tsotsos (2020) for a recent survey).

A canonical example is the Soar architecture (Fig. 2A). Soar stores productions in long-term memory and executes them based on how well their preconditions match working memory (Fig. 2B). These productions specify actions that modify the contents of working and long-term memory. We next provide a brief overview of Soar and refer readers to Laird (2022; 2019) for deeper introductions.

Memory. Building on psychological theories, Soar uses several types of memory to track the agent's state (Atkinson and Shiffrin, 1968). *Working memory* (Baddeley and Hitch, 1974) reflects the agent's current circumstances: it stores the agent's recent perceptual input, goals, and results from intermediate, internal reasoning. *Long term memory* is divided into three distinct types. *Procedural* memory stores the production system itself: the set of rules that can be applied to working memory to determine the agent's behavior. *Semantic* memory stores facts about the world (Lindes and Laird, 2016), while *episodic* memory stores sequences of the agent's past behaviors (Nuxoll and Laird, 2007).

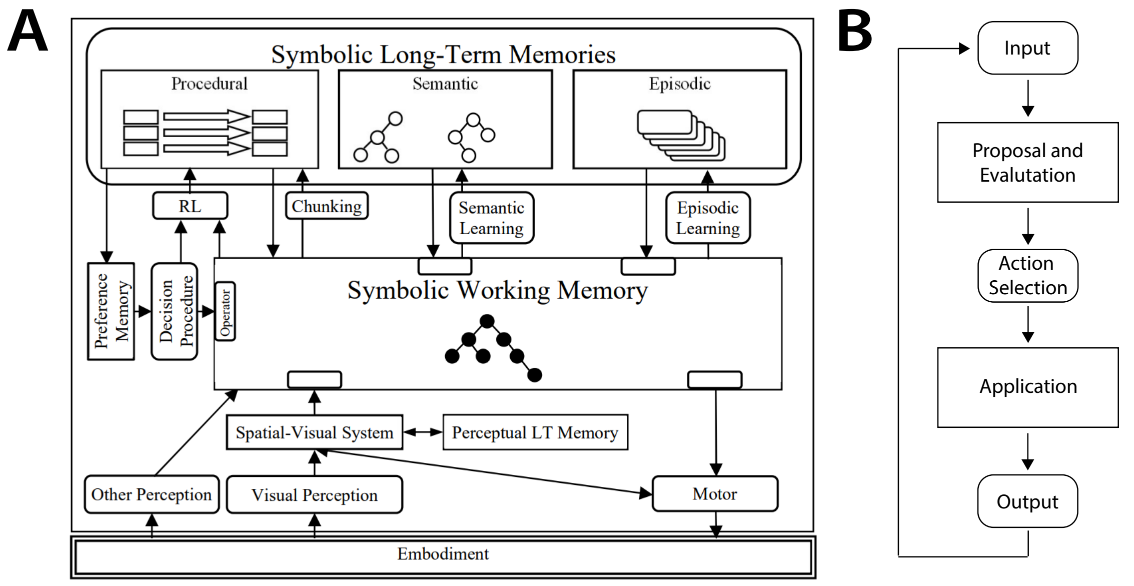

Figure 2: Cognitive architectures augment a production system with sensory groundings, long-term memory, and a decision procedure for selecting actions. **A**: The Soar architecture, reproduced from Laird (2022). **B**: Soar's decision procedure uses productions to select and implement actions. These actions may be *internal* (such as modifying the agent's memory) or *external* (such as a motor command).

Grounding. Soar can be instantiated in simulations (Tambe et al., 1995; Jones et al., 1999) or real-world robotic systems (Laird et al., 2012). In embodied contexts, a variety of sensors stream perceptual input into working memory, where it is available for decision making. Soar agents can also be equipped with actuators, allowing for physical actions and interactive learning via language (Mohan et al., 2012; Mohan and Laird, 2014; Kirk and Laird, 2014).

Decision making. Soar implements a decision loop that evaluates productions and applies the one that matches best (Fig. 2B). Productions are stored in long-term procedural memory. During each decision cycle, their preconditions are checked against the agent's working memory. In the *proposal and evaluation* phase, a set of productions is used to generate and rank a candidate set of possible actions.[1] The best action is then chosen.[2] Another set of productions is then used to implement the action – for example, modifying the contents of working memory or issuing a motor command.

Learning. Soar supports multiple modes of learning. First, new information can be stored directly in long-term memory: facts can be written to semantic memory, while experiences can be written to episodic memory (Derbinsky et al., 2012). This information can later be retrieved back into working memory when needed for decision-making. Second, behaviors can be modified. Reinforcement learning (Sutton and Barto, 2018) can be used to up-weight productions that have yielded good outcomes, allowing the agent to learn from experience (Nason and Laird, 2005). Most remarkably, Soar is also capable of writing new productions into its procedural memory (Laird et al., 1986) – effectively updating its source code.

Cognitive architectures were used broadly across psychology and computer science, with applications including robotics (Laird et al., 2012), military simulations (Jones et al., 1999; Tambe et al., 1995), and intelligent tutoring (Koedinger et al., 1997). Yet they have become less popular in the AI community over the last few decades. This decrease in popularity reflects two of the challenges involved in such systems: they are limited to domains that can be described by logical predicates and require many pre-specified rules to function.

[1]In more detail, Soar divides productions into two types: "operators," which we refer to as actions, and "rules" which are used to propose, evaluate, and execute operators. Differentiating these is conceptually important for Soar but not language agents, and so we elide the distinction.

[2]If no actions are valid, or multiple actions tie, then an *impasse* occurs. Soar creates a subgoal to resolve the impasse, resulting in hierarchical task decomposition. We refer the reader to Laird (2022) for a more detailed discussion.

Intriguingly, LLMs appear well-posed to meet these challenges. First, they operate over arbitrary text, making them more flexible than logic-based systems. Second, rather than requiring the user to specify deterministic productions, they learn a distribution over productions via pre-training on a vast internet corpus. Recognizing this, researchers have begun to use LLMs within cognitive architectures – most typically leveraging their implicit world knowledge (Wray et al., 2021) to augment traditional symbolic approaches (Kirk et al., 2023; Romero et al., 2023). Here, instead, we import principles from cognitive architecture to guide the design of LLM-based agents.

2.4 Language models and agents

Language modeling is a decades-old endeavor in the NLP and AI communities to develop systems that can generate text given some context (Jurafsky, 2000). Formally, language models learn a distribution $P(w_i|w_{<i})$, where each w is an individual token (word). This model can then generate text by sampling from the distribution, one token at a time. At its core, a language model is a probabilistic input-output system, since there are inherently several ways to continue a text (e.g., "I went to the" $\rightarrow$ "market" | "beach" | ...). While earlier attempts at modeling language (e.g., n-grams) faced challenges in generalization and scaling, there has been a recent resurgence of the area due to the rise of Transformer-based (Vaswani et al., 2017) LLMs with a large number (billions) of parameters (e.g., GPT-4 (OpenAI, 2023)) and smart tokenization schemes. Modern LLMs are trained to predict the next token on enormous amounts of data, which helps them accumulate knowledge from a large number of input-output combinations and successfully generate human-like text.

Unexpectedly, training these models on internet-scale text also made them useful for many tasks beyond generating text, such as writing code (Li et al., 2022; Rozière et al., 2023; Li et al., 2023), modeling proteins (Meier et al., 2021), and acting in interactive environments (Yao et al., 2022b; Nakano et al., 2021). The latter has led to the rise of "language agents" – systems that use LLMs as a core computation unit to reason, plan, and act – with applications in areas such as robotics (Ahn et al., 2022), web manipulation (Yao et al., 2022a; Deng et al., 2023), puzzle solving (Yao et al., 2023; Hao et al., 2023) and interactive code generation (Yang et al., 2023a). The combination of language understanding and decision making capabilities is an exciting and emerging direction that promises to bring these agents closer to human-like intelligence.

3 Connections between Language Models and Production Systems

Based on their common origins in processing strings, there is a natural analogy between production systems and language models. We first develop this analogy. We then review prompting methods, showing that these efforts recapitulate the algorithms and agents based on production systems – and suggesting that cognitive architectures like those developed for production systems may be usefully applied to LLMs.

3.1 Language models as probabilistic production systems

In their original instantiation, production systems specified the set of strings that could be generated from a starting point, breaking this process down into a series of string rewriting operations. Language models also define a possible set of expansions or modifications of a string – the prompt provided to the model.[3]

For example, we can formulate the problem of completing a piece of text as a production. If X is the prompt and Y the continuation, then we can write this as the production $X \rightarrow X\ Y$.[4] We might want to allow multiple possible continuations, in which case we have $X \rightarrow X\ Y_i$ for some set of Y_i. LLMs assign a *probability* to each of these completions. Viewed from this perspective, the LLM defines a probability distribution over *which productions to select* when presented with input X, yielding a distribution $P(Y_i|X)$ over possible completions (Dohan et al., 2022). LLMs can thus be viewed as probabilistic production systems that sample a possible completion each time they are called, e.g., $X \rightsquigarrow X\,Y$.

[3]In this work, we focus on autoregressive LLMs which are typically used to construct language agents. However, bidirectional LLMs trained on the cloze task, such as BERT (Devlin et al., 2019), can be seen in a similar light: they define a distribution over *in-filling* productions.

[4]Alternatively, we can treat the prompt as input and take the output of the LLM as the next state, represented by the production $X \rightarrow Y$ – a more literal form of rewriting.

Prompting Method	Production Sequence
Zero-shot	$Q \overset{\text{LLM}}{\rightsquigarrow} Q\ A$
Few-shot (Brown et al., 2020)	$Q \longrightarrow Q_1\ A_1\ Q_2\ A_2\ Q \overset{\text{LLM}}{\rightsquigarrow} Q_1\ A_1\ Q_2\ A_2\ Q\ A$
Zero-shot Chain-of-Thought (Kojima et al., 2022)	$Q \longrightarrow Q_{\text{Step-by-step}} \overset{\text{LLM}}{\rightsquigarrow} Q_{\text{Step-by-step}} A$
Retrieval Augmented Generation (Lewis et al., 2020)	$Q \xrightarrow{\text{Wiki}} Q\,O \overset{\text{LLM}}{\rightsquigarrow} Q\,O\,A$
Socratic Models (Zeng et al., 2022)	$Q \overset{\text{VLM}}{\rightsquigarrow} Q\,O \overset{\text{LLM}}{\rightsquigarrow} Q\,O\,A$
Self-Critique (Saunders et al., 2022)	$Q \overset{\text{LLM}}{\rightsquigarrow} Q\,A \overset{\text{LLM}}{\rightsquigarrow} Q\,A\,C \overset{\text{LLM}}{\rightsquigarrow} Q\,A\,C\,A$

Table 1: Conceptual diagram illustrating how prompting methods manipulate the input string before generating completions. Q = question, A = answer, O = observation, C = critique, and $\rightsquigarrow$ denotes sampling from a stochastic production. These pre-processing manipulations – which can employ other models such as vision-language models (VLMs), or even the LLM itself – can be seen as productions. Prompting methods thus define a *sequence* of productions.

This probabilistic form offers both advantages and disadvantages compared to traditional production systems. The primary disadvantage of LLMs is their inherent opaqueness: while production systems are defined by discrete and human-legible rules, LLMs consist of billions of uninterpretable parameters. This opaqueness – coupled with inherent randomness from their probabilistic formulation – makes it challenging to analyze or systematically control their behaviors (Romero et al., 2023). Nonetheless, their scale and pre-training provide massive advantages over traditional production systems. LLMs pre-trained on large-scale internet data learn a remarkably effective prior over string completions, allowing them to solve a wide range of tasks out of the box (Huang et al., 2022b).

3.2 Prompt engineering as control flow

The weights of an LLM define a prioritization over output strings (completions), conditioned by the input string (the prompt). The resulting distribution can be interpreted as a task-specific prioritization of productions – in other words, a simple control flow. Tasks such as question answering can be formulated directly as an input string (the question), yielding conditional distributions over completions (possible answers).

Early work on few-shot learning (Brown et al., 2020) and prompt engineering (Wei et al., 2022b; Kojima et al., 2022; Xu et al., 2023c) found that the LLM could be further biased towards high-quality productions by pre-processing the input string. These simple manipulations – typically concatenating additional text to the input – can themselves be seen as productions, meaning that these methods define a sequence of productions (Table 1). Later work extended these approaches to dynamic, context-sensitive prompts: for example, selecting few-shot examples that are maximally relevant to the input (Liu et al., 2021) or populating a template with external observations from video (Zeng et al., 2022) or databases (Lewis et al., 2020). For a survey of such prompting techniques, see Liu et al. (2023b).

Subsequent work used the LLM itself as a pre-processing step, eliciting targeted reasoning to foreground a particular aspect of the problem (Bai et al., 2022; Jin et al., 2022; Ganguli et al., 2023; Madaan et al., 2023; Saunders et al., 2022; Kim et al., 2023; Kirk et al., 2023) or generate intermediate reasoning steps (Tafjord et al., 2021; Creswell et al., 2023; Yao et al., 2023) before returning an answer. *Chaining* multiple calls to an LLM (Wu et al., 2022a;b; Dohan et al., 2022) allows for increasingly complicated algorithms (Fig. 3).

3.3 Towards cognitive language agents

Language agents move beyond pre-defined prompt chains and instead place the LLM in a feedback loop with the external environment (Fig. 1B). These approaches first transform multimodal input into text and pass it

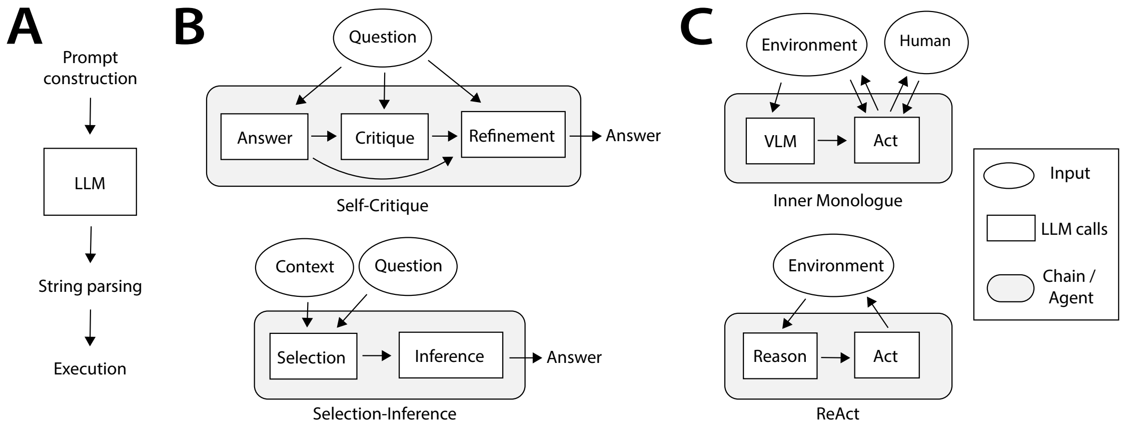

Figure 3: From language models to language agents. **A**: Basic structure of an LLM call. Prompt construction selects a template and populates it with variables from working memory. After calling the LLM, the string output is parsed into an action space and executed. An LLM call may result in one or more actions – for example, returning an answer, calling a function, or issuing motor commands. **B**: *Prompt chaining* techniques such as Self-Critique (Wang et al., 2022b) or Selection-Inference (Creswell et al., 2023) use a pre-defined sequence of LLM calls to generate an output. **C**: *Language agents* such as Inner Monologue (Huang et al., 2022c) and ReAct (Yao et al., 2022b) instead use an interactive feedback loop with the external environment. Vision-language models (VLMs) can be used to translate perceptual data into text for the LLM to process.

to the LLM. The LLM's output is then parsed and used to determine an external action (Fig. 3C). Early agents interfaced the LLM directly with the external environment, using it to produce high-level instructions based on the agent's state (Ahn et al., 2022; Huang et al., 2022c; Dasgupta et al., 2022). Later work developed more sophisticated language agents that use the LLM to perform intermediate reasoning before selecting an action (Yao et al., 2022b). The most recent agents incorporate sophisticated learning strategies such as reflecting on episodic memory to generate new semantic inferences (Shinn et al., 2023) or modifying their program code to generate procedural knowledge (Wang et al., 2023a), using their previous experience to adapt their future behaviors.

These *cognitive* language agents employ nontrivial LLM-based reasoning and learning (Fig. 1C). Just as cognitive architectures were used to structure production systems' interactions with agents' internal state and external environments, we suggest that they can help design LLM-based cognitive agents. In the remainder of the paper, we use this perspective to organize existing approaches and highlight promising extensions.

4 Cognitive Architectures for Language Agents (CoALA): A Conceptual Framework

We present Cognitive Architectures for Language Agents (CoALA) as a framework to organize existing language agents and guide the development of new ones. CoALA positions the LLM as the core component of a larger cognitive architecture (Figure 4). The agent's internal **memory** is organized into discrete modules (Section 4.1), and its **action space** is divided into external and internal actions (Figure 5):

- **External actions** interact with external environments (e.g., control a robot, communicate with a human, navigate a website) through **grounding** (Section 4.2).

- **Internal actions** interact with internal memories. Depending on which memory gets accessed and whether the access is read or write, internal actions can be further decomposed into three kinds: **reasoning** (update the short-term working memory with LLM; Section 4.4), **retrieval** (read from long-term memory; Section 4.3), and **learning** (write to long-term memory; Section 4.5).

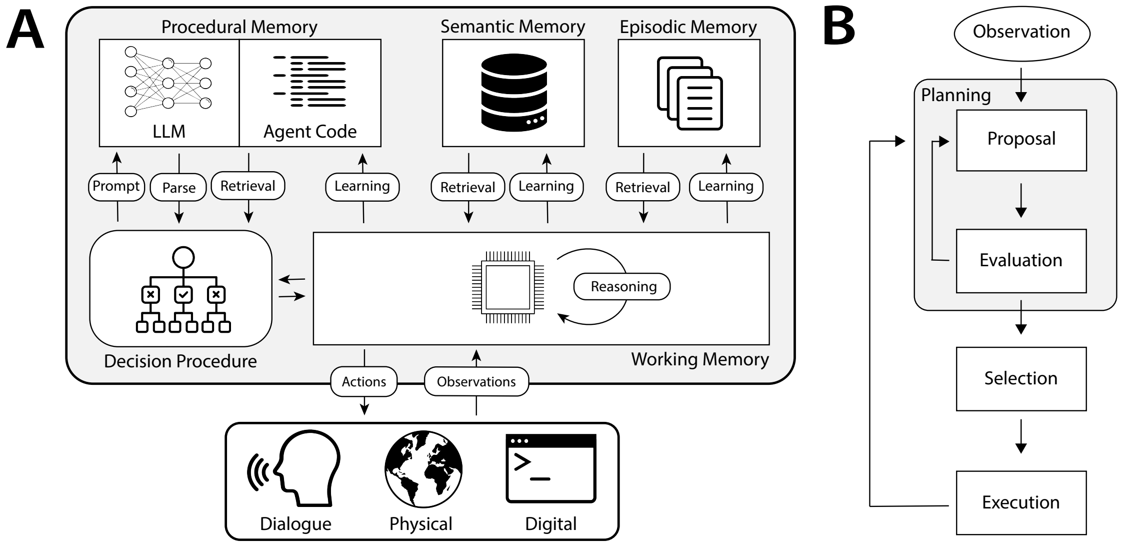

Figure 4: Cognitive architectures for language agents (CoALA). **A**: CoALA defines a set of interacting modules and processes. The **decision procedure** executes the agent's source code. This source code consists of procedures to interact with the LLM (prompt templates and parsers), internal memories (retrieval and learning), and the external environment (grounding). **B**: Temporally, the agent's decision procedure executes a **decision cycle** in a loop with the external environment. During each cycle, the agent uses **retrieval** and **reasoning** to plan by proposing and evaluating candidate **learning** or **grounding** actions. The best action is then selected and executed. An observation may be made, and the cycle begins again.

Language agents choose actions via **decision making**, which follows a repeated cycle (Section 4.6, Figure 4B). In each cycle, the agent can use reasoning and retrieval actions to plan. This planning subprocess selects a grounding or learning action, which is executed to affect the outside world or the agent's long-term memory.

CoALA (Figure 4) is inspired by the decades of research in cognitive architectures (Section 2.3), leveraging key concepts such as memory, grounding, learning, and decision making. Yet the incorporation of an LLM leads to the addition of "reasoning" actions, which can flexibly produce new knowledge and heuristics for various purposes – replacing hand-written rules in traditional cognitive architectures. It also makes the text the *de facto* internal representation, streamlining agents' memory modules. Finally, recent advances in vision-language models (VLMs; Alayrac et al., 2022) can simplify grounding by providing a straightforward translation of perceptual data into text (Zeng et al., 2022).

The rest of this section details key concepts in CoALA: memory, actions (grounding, reasoning, retrieval, and learning), and decision making. For each concept, we use existing language agents (or relevant NLP/RL methods) as examples – or note gaps in the literature to highlight possibilities that remain unexplored.

4.1 Memory

Language models are stateless: they do not persist information across calls. In contrast, language agents may store and maintain information internally for multi-step interaction with the world. Under the CoALA framework, language agents explicitly organize information into multiple memory modules, each containing a different form of information. These include short-term working memory and several long-term memories: episodic, semantic, and procedural.

Working memory. Working memory maintains active and readily available information as symbolic variables for the current decision cycle (Section 4.6). This includes perceptual inputs from grounding, knowledge generated by reasoning, knowledge retrieved from long-term memory, and other core information carried over from the previous decision cycle (e.g., the task instruction). Importantly, the working memory (a data structure) should not be confused with the LLM context (a string). Rather, the LLM input is synthesized

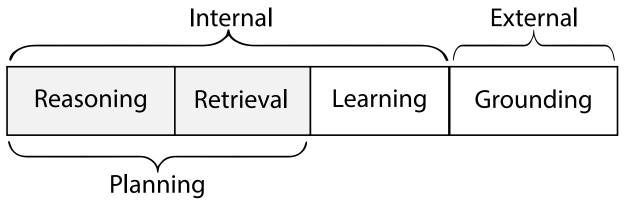

Figure 5: Agents' action spaces can be divided into **internal** memory accesses and **external** interactions with the world. **Reasoning** and **retrieval** actions are used to support planning.

from a subset of working memory (e.g., a prompt template and relevant variables). The LLM output is then parsed back into other variables (e.g., an action name and arguments) which are stored back in working memory and used to execute the corresponding action (Figure 3A). Besides the LLM, the working memory also interacts with long-term memories and grounding interfaces. It thus serves as the central hub connecting different components of a language agent.

Episodic memory. Episodic memory stores experience from earlier decision cycles. This can consist of training input-output pairs (Rubin et al., 2021), history event flows (Weston et al., 2014; Park et al., 2023), game trajectories from previous episodes (Yao et al., 2020; Tuyls et al., 2022), or other representations of the agent's experiences. During the planning stage of a decision cycle, these episodes may be retrieved into working memory to support reasoning. An agent can also write new experiences from working to episodic memory as a form of learning (Section 4.5).

Semantic memory. Semantic memory stores an agent's knowledge about the world and itself. Traditional NLP or RL approaches that leverage retrieval for reasoning or decision making initialize semantic memory from an external database for knowledge support. For example, retrieval-augmented methods in NLP (Lewis et al., 2020; Borgeaud et al., 2022; Chen et al., 2017) can be viewed as retrieving from a semantic memory of unstructured text (e.g., Wikipedia). In RL, "reading to learn" approaches (Branavan et al., 2012; Narasimhan et al., 2018; Hanjie et al., 2021; Zhong et al., 2021) leverage game manuals and facts as a semantic memory to affect the policy. While these examples essentially employ a fixed, read-only semantic memory, language agents may also write new knowledge obtained from LLM reasoning into semantic memory as a form of learning (Section 4.5) to incrementally build up world knowledge from experience.

Procedural memory. Language agents contain two forms of procedural memory: *implicit* knowledge stored in the LLM weights, and *explicit* knowledge written in the agent's code. The agent's code can be further divided into two types: procedures that implement actions (reasoning, retrieval, grounding, and learning procedures), and procedures that implement decision making itself (Section 4.6). During a decision cycle, the LLM can be accessed via reasoning actions, and various code-based procedures can be retrieved and executed. Unlike episodic or semantic memory that may be initially empty or even absent, procedural memory must be initialized by the designer with proper code to bootstrap the agent. Finally, while learning new actions by writing to procedural memory is possible (Section 4.5), it is significantly riskier than writing to episodic or semantic memory, as it can easily introduce bugs or allow an agent to subvert its designers' intentions.

4.2 Grounding actions

Grounding procedures execute external actions and process environmental feedback into working memory as text. This effectively simplifies the agent's interaction with the outside world as a "text game" with textual observations and actions. We categorize three kinds of external environments:

Physical environments. Physical embodiment is the oldest instantiation envisioned for AI agents (Nilsson, 1984). It involves processing perceptual inputs (visual, audio, tactile) into textual observations (e.g., via pre-trained captioning models), and affecting the physical environments via robotic planners that take language-based commands. Recent advances in LLMs have led to numerous robotic projects (Ahn et al., 2022; Liang et al., 2023a; Singh et al., 2023; Palo et al., 2023; Ren et al., 2023) that leverage LLMs as a "brain" for robots to generate actions or plans in the physical world. For perceptual input, vision-language models are typically used to convert images to text (Alayrac et al., 2022; Sumers et al., 2023) providing additional context for the LLM (Driess et al., 2023; Huang et al., 2023; Brohan et al., 2022; 2023).

Dialogue with humans or other agents. Classic linguistic interactions allow the agent to accept instructions (Winograd, 1972; Tellex et al., 2011) or learn from people (Nguyen et al., 2021; Sumers et al., 2022; 2021). Agents capable of *generating* language ask for help (Ren et al., 2023; Nguyen et al., 2022b) or clarification (Biyik and Palan, 2019; Sadigh et al., 2017) – or entertain or emotionally help people (Zhang et al., 2020; Zhou et al., 2018; Pataranutaporn et al., 2021; Hasan et al., 2023; Ma et al., 2023). Recent work also investigates interaction among multiple language agents for social simulation (Park et al., 2023; Jinxin et al., 2023; Gao et al., 2023), debate (Chan et al., 2023; Liang et al., 2023b; Du et al., 2023), improved safety (Irving et al., 2018), or collabrative task solving (Qian et al., 2023; Wu et al., 2023; Hong et al., 2023).

Digital environments. This includes interacting with games (Hausknecht et al., 2020; Côté et al., 2019; Shridhar et al., 2020; Wang et al., 2022a; Liu et al., 2022), APIs (Schick et al., 2023; Yao et al., 2022b; Parisi et al., 2022; Tang et al., 2023b), and websites (Shi et al., 2017; Nakano et al., 2021; Yao et al., 2022a; Zhou et al., 2023; Gur et al., 2023; Deng et al., 2023) as well as general code execution (Yang et al., 2023a; Le et al., 2022; Ni et al., 2023). Such digital grounding is cheaper and faster than physical or human interaction. It is thus a convenient testbed for language agents and has been studied with increasing intensity in recent years. In particular, for NLP tasks that require augmentation of external knowledge or computation, stateless digital APIs (e.g., search, calculator, translator) are often packaged as "**tools**" (Parisi et al., 2022; Schick et al., 2023; Xu et al., 2023a; Tang et al., 2023b; Qin et al., 2023), which can be viewed as special "single-use" digital environments.

4.3 Retrieval actions

In CoALA, a retrieval procedure reads information from long-term memories into working memory. Depending on the information and memory type, it could be implemented in various ways, e.g., rule-based, sparse, or dense retrieval. For example, Voyager (Wang et al., 2023a) loads code-based skills from a skill library via dense retrieval to interact with the Minecraft world – effectively retrieving grounding procedures from a procedural memory. Generative Agents (Park et al., 2023) retrieves relevant events from episodic memory via a combination of recency (rule-based), importance (reasoning-based), and relevance (embedding-based) scores. DocPrompting (Zhou et al., 2022a) proposes to leverage library documents to assist code generation, which can be seen as retrieving knowledge from semantic memory.

4.4 Reasoning actions

Reasoning allows language agents to process the contents of working memory to generate new information. Unlike retrieval (which reads from long-term memory into working memory), reasoning reads from *and* writes to working memory. This allows the agent to summarize and distill insights about the most recent observation (Yao et al., 2022b), the most recent trajectory (Shinn et al., 2023), or information retrieved from long-term memory (Park et al., 2023). Reasoning can be used to support learning (by writing the results into long-term memory) or decision-making (by using the results as additional context for subsequent LLM calls).

4.5 Learning actions

Learning occurs by writing information to long-term memory, which includes a spectrum of diverse procedures.

Updating episodic memory with experience. It is common practice for RL agents to store episodic trajectories to update a parametric policy (Blundell et al., 2016; Pritzel et al., 2017) or establish a non-parametric policy (Ecoffet et al., 2019; Tuyls et al., 2022). For language agents, added experiences in episodic memory may be retrieved later as examples and bases for reasoning or decision making (Weston et al., 2014; Rubin et al., 2021; Park et al., 2023).

Updating semantic memory with knowledge. Recent work (Shinn et al., 2023; Park et al., 2023) has applied LLMs to reason about raw experiences and store the resulting inferences in semantic memory. For example, Reflexion (Shinn et al., 2023) uses an LLM to reflect on failed episodes and stores the results (e.g., "there is no dishwasher in kitchen") as semantic knowledge to be attached to LLM context for solving later episodes. Finally, work in robotics (Chen et al., 2023a) uses vision-language models to build a semantic map of the environment, which can later be queried to execute instructions.

Updating LLM parameters (procedural memory). The LLM weights represent implicit procedural knowledge. These can be adjusted to an agent's domain by fine-tuning during the agent's lifetime. Such fine-tuning can be accomplished via supervised or imitation learning (Hussein et al., 2017), reinforcement learning (RL) from environment feedback (Sutton and Barto, 2018), human feedback (RLHF) (Christiano et al., 2017; Ouyang et al., 2022; Nakano et al., 2021), or AI feedback (Bai et al., 2022). For example, XTX (Tuyls et al., 2022) periodically finetunes a small language model on high-scoring trajectories stored in episodic memory, which serves as a robust "exploitation" policy to reach exploration frontiers in the face of stochasity. Recent work (Huang et al., 2022a; Zelikman et al., 2022) has also shown the potential of finetuned small language models distilling then surpassing larger ones. Fine-tuning the agent's LLM is a costly form of learning; thus, present studies specify learning schedules. However, as training becomes more efficient – or if agents utilize smaller subtask-specific LLMs – it may be possible to allow language agents to autonomously determine when and how to fine-tune their LLMs.

Updating agent code (procedural memory). CoALA allows agents to update their source code, thus modifying the implementation of various procedures. These can be broken down as follows:

- **Updating reasoning** (e.g., prompt templates; Gao et al., 2020; Zhou et al., 2022b). For example, APE (Zhou et al., 2022b) infers prompt instructions from input-output examples, then uses these instructions as part of the LLM prompt to assist task solving. Such a prompt update can be seen as a form of learning to reason.
- **Updating grounding** (e.g., code-based skills; Liang et al., 2023a; Ellis et al., 2021; Wang et al., 2023a). For example, Voyager (Wang et al., 2023a) maintains a curriculum library. Notably, current methods are limited to creating new code skills to interact with external environments.
- **Updating retrieval**. To our knowledge, these learning options are not studied in recent language agents. Retrieval is usually considered a basic action designed with some fixed implementation (e.g., BM25 or dense retrieval), but research in query/document expansion (Nogueira et al., 2019; Wang et al., 2023c; Tang et al., 2023a) or retrieval distillion (Izacard et al., 2021) may be helpful for language agents to learn better retrieval procedures.
- **Updating learning or decision-making**. Finally, it is theoretically possible for CoALA agents to learn new procedures for learning or decision making, thus providing significant adaptability. In general, however, updates to these procedures are risky both for the agent's functionality and alignment. At present, we are not aware of any language agents that implement this form of learning; we discuss such possibilities more in Section 6.

While RL agents usually fix one way of learning (e.g., Q-learning, PPO, or A3C) and learn by updating model parameters, language agents can select from a diversity of learning procedures. This allows them to learn rapidly by storing task-relevant language (cheaper and quicker than parameter updates), and leverage multiple forms of learning to compound their self-improvement (e.g., Generative Agents discussed in Section 5).

Finally, while our discussion has mostly focused on **adding** to memory, **modifying** and **deleting** (a case of "unlearning") are understudied in recent language agents. We address these areas more in Section 6.

4.6 Decision making

With various actions (grounding, learning, reasoning, retrieval) in the action space, how should a language agent choose which action to apply? This is handled by the decision making procedure, which structures the agent's actions into decision making cycles (or decision cycles for short). In each cycle, language agents use a systematic code-based procedure with the assistance of reasoning and retrieval actions to decide a learning or grounding action (**planning stage**), then execute the action (**execution stage**) – then the cycle loops again.

Planning stage. During planning, reasoning, and retrieval can be flexibly applied to propose, evaluate, and select actions, and these sub-stages could interleave or iterate to build up multi-step simulation before taking an external action (Yao et al., 2023; Hao et al., 2023). It also enables agents to iteratively improve candidate solutions – for example, by using the LLM to simulate them, identifying defects, and proposing modifications that address those defects (Kirk et al., 2023; Shinn et al., 2023).

- **Proposal**. The proposal sub-stage generates one or more action candidates. The usual approach is to use **reasoning** (and optionally retrieval) to sample one (Huang et al., 2022c) or more (Chen et al., 2021; Wang et al., 2022b) actions from the LLM. For simple domains with limited actions, the proposal stage might simply include all actions (e.g., SayCan in Section 5). It is also possible to leverage if-else or while-if code structures (Wang et al., 2023a; Park et al., 2023), or more advanced planner code packages such as Planning Domain Definition Language (PDDL; Haslum et al., 2019) for more informed proposal (Liu et al., 2023a; Dagan et al., 2023) in well-defined domains.

- **Evaluation**. If multiple actions are proposed, the evaluation sub-stage assigns a scalar value to each. This may use heuristic rules, LLM (perplexity) values (Ahn et al., 2022), learned values (Yao et al., 2020), LLM reasoning (Yao et al., 2023; Hao et al., 2023), or some combination. Particularly, LLM reasoning can help evaluate actions by internally simulating their grounding feedback from the external world (Hao et al., 2023; Yang et al., 2023a).

- **Selection**. Given a set of actions and their values, the selection procedure selects an action by argmax, or softmax, or decides to re-iterate to some proposal or evaluate sub-stage.

Execution stage. The selected action is applied by executing the relevant procedures from the agent's source code. An observation feedback can be made from the external environment, providing feedback from the agent's action, and the cycle loops again.

Empirically, many early language agents simply use LLMs to propose an action (Schick et al., 2023), a sequence of actions (Huang et al., 2022b), or evaluate a fixed set of actions (Ahn et al., 2022) without intermediate reasoning or retrieval. Followup work (Yao et al., 2022b; Shinn et al., 2023; Xu et al., 2023b; Lin et al., 2023; Wang et al., 2023a; Park et al., 2023) has exploited intermediate reasoning and retrieval to analyze the situation, make and maintain action plans, refine the previous action given the environmental feedback, and leveraged a more complex procedure to propose a single action (still without evaluation and selection). Most recently, research has started to investigate more complex decision making by proposing more than one action, and with iterative proposal and evaluation. These procedures are beginning to resemble classical planning algorithms: for example, Tree of Thoughts (Yao et al., 2023) leverages LLM to iteratively propose and evaluate "thoughts" in a systematic tree search (DFS/BFS) to return a final solution to a problem. RAP (Hao et al., 2023) similarly generates partial thoughts and maintains them in a tree structure, using Monto Carlo Tree Search (MCTS; Browne et al., 2012) and LLM-based simulation to evaluate thoughts.

5 Case Studies

With variations and ablations of the memory modules, action space, and decision making procedures, CoALA can express a wide spectrum of language agents. Table 2 lists some popular recent methods across diverse domains — from Minecraft to robotics, from pure reasoning to social simulacra. CoALA helps characterize their internal mechanisms and reveal their similarities and differences in a simple and structured way.

SayCan (Ahn et al., 2022) grounds a language model to robotic interactions in a kitchen to satisfy user commands (e.g., "I just worked out, can you bring me a drink and a snack to recover?"). Its long-term memory is procedural only (an LLM and a learned value function). The action space is external only – a fixed set of 551 grounding skills (e.g., "find the apple", "go to the table"), with no internal actions of reasoning, retrieval, or learning. During decision making, SayCan evaluates each action using a combination of LLM and learned values, which balance a skill's usefulness and groundedness. SayCan therefore employs the LLM primarily as a stateless single-step planner.

	Long-term Memory[5]	External Grounding	Internal Actions	Decision Making
SayCan (Ahn et al., 2022)	-	physical	-	evaluate
ReAct (Yao et al., 2022b)	-	digital	reason	propose
Voyager (Wang et al., 2023a)	procedural	digital	reason/retrieve/learn	propose
Generative Agents (Park et al., 2023)	episodic/semantic	digital/agent	reason/retrieve/learn	propose
Tree of Thoughts (Yao et al., 2023)	-	digital[6]	reason	propose, evaluate, select

Table 2: Some recent language agents cast into the CoALA framework.

ReAct (Yao et al., 2022b) is a language agent grounded to various digital environments (e.g., Wikipedia API, text game, website). Like SayCan, it lacks semantic or episodic memory and therefore has no retrieval or learning actions. Its action space consists of (internal) reasoning and (external) grounding. Its decision cycle is fixed to use a single reasoning action to analyze the situation and (re)make action plans, then generates a grounding action without evaluation or selection stages. ReAct can be considered the simplest language agent that leverages both internal and external actions, and is the initial work that demonstrates their synergizing effects: reasoning helps guide acting, while acting provides environmental feedback to support reasoning.

Voyager (Wang et al., 2023a) is a language agent grounded to the Minicraft API. Unlike SayCan, which grounds to perception via the learned value function, Voyager's grounding is text-only. It has a long-term procedural memory that stores a library of code-based grounding procedures a.k.a. skills (e.g., "combatZombie", "craftStoneSword"). This library is hierarchical: complex skills can use simpler skills as sub-procedures (e.g., "combatZombie" may call "craftStoneSword" if no sword is in inventory). Most impressively, its action space has all four kinds of actions: grounding, reasoning, retrieval, and learning (by adding new grounding procedures). During a decision cycle, Voyager first reasons to propose a new task objective if it is missing in the working memory, then reasons to propose a code-based grounding procedure to solve the task. In the next decision cycle, if the environmental feedback is reasoned to suggest task completion, a learning action is selected to add the grounding procedure to procedural memory; otherwise, reasoning is used to reflect bugs and refine code, and a grounding action is selected to attempt task solving again. The importance of long-term memory and procedural learning is empirically verified by comparing to baselines like ReAct and AutoGPT and ablations without the procedural memory, where Voyager is shown to better explore areas, master tech tree, and zero-shot generalize to unseen tasks.

Generative Agents (Park et al., 2023) is a language agent grounded to a sandbox game in which it interacts with the environment and other agents. Each agent has a long-term episodic memory that stores everyday events in a list and a long-term semantic memory that stores their periodic reflections over the episodic memory (e.g., "I like to ski now."). Its action space also has all four kinds of actions: grounding, reasoning, retrieval, and learning. Notably, it learns by adding new events to episodic memory, and new reflections to semantic memory. During decision making, it retrieves relevant reflections from semantic memory and reasons to make a high-level plan of the day, or with grounding observations, reasons to maintain or adjust the plan.

Tree of Thoughts (ToT) (Yao et al., 2023) can be seen as a special kind of language agent with only one external action: submitting a final solution to a reasoning problem (game of 24, creative writing, crosswords puzzle). It has no long-term memory, and only reasoning in its internal action space, but differs from all previous agents in its deliberate decision making. During planning, ToT iteratively proposes, evaluates, and selects "thoughts" (reasoning actions) based on LLM reasoning, and systematically maintains them via a tree search algorithm to enable global exploration as well as local backtrack and foresight.

6 Actionable Insights

While there have been several recent surveys related to language agents (Mialon et al., 2023; Weng, 2023; Wang et al., 2023b), CoALA offers a structured way of thinking about these agents grounded in the well-established research of cognitive architectures. Not every application will require all of CoALA's components, but using concepts from CoALA while designing and describing the system will aid in modularity and clarity. For

[5]All agents contain some procedural memory (agent code and LLM weights), so here we only list writable procedural memory.
[6]Special digital grounding with the only external action being submitting a final answer.

example, it may be beneficial to consider whether an application requires semantic or episodic memory; whether it should be capable of modifying its semantic memory; and so on. Importantly, such a systematic way of thinking readily leads to several future directions for language agents.

Working memory and reasoning: thinking beyond LLM prompt engineering. If one only thinks of maintaining the LLM context with string manipulations and parsings, model design is likely to degenerate into low-level trivialities and "prompt engineering" tricks. Instead, the community should think about a structured working memory and systematic "reasoning" actions that update working memory variables. Recent steps in this direction include more systematic prompting frameworks such as LangChain[7] and LlamaIndex[8], as well as more structural output parsing solutions such as Guidance[9] and OpenAI function calling[10]. Defining and building good working memory modules will be an important direction of future research. These modules could be especially important for industry solutions where LLM reasoning needs to seamlessly integrate and conform with large-scale code infrastructure.

Long-term memory: thinking beyond retrieval augmentation. Compared to retrieval-augmented language models (Guu et al., 2020; Lewis et al., 2020; Borgeaud et al., 2022) that only read from human-written corpora, memory-augmented language agents can both read and write self-generated content autonomously. By organically combining existing human knowledge with self-discovered and self-maintained experience, knowledge, and skills in long-term memory, future language agents may more efficiently learn and solve tasks. For example, a future coding agent could maintain human-provided programming knowledge (semantic) such as manuals, textbooks, problems, and examples, as well as its problem solutions and test records (episodic), reflections, and summaries on top of these experiences (semantic), and a gradually expanding code library that stores useful methods, e.g., QuickSort, GCD, LCA (procedural). Similar development is also possible for solving interactive text games, book-level QA, personalized chat, or any task where agents could exploit existing human experiences and explore new ones.

Learning: thinking beyond in-context learning or finetuning. CoALA's definition of "learning" as long-term memory updates is simple yet versatile. As detailed in Section 4.5, it covers updates to episodic experience, semantic knowledge, LLM weights (finetuning), and various procedural updates that are currently understudied. For example, learning better retrieval procedures could enable agents to better connect memorized information to new scenarios, and recent expansion-based techniques (Nogueira et al., 2019; Wang et al., 2023c; Tang et al., 2023a) may be readily applied (e.g., reason about "in what situations would this knowledge be useful?", and append the reasoning results to the knowledge to help later connect the knowledge to new situations). Learning new learning and decision procedures can be seen as ways of "meta-learning" and "meta-decision making" for language agents, and are currently understudied due to their difficulty and high risk (see safety discussion in the next paragraph). However, these may be necessary to empower agents to go beyond the limitations of human-provided code. Also, future directions could explore learning smaller models for specific reasoning needs (Zelikman et al., 2022; Huang et al., 2022a; Ahn et al., 2022), deleting unneeded memory items for "unlearning" (Nguyen et al., 2022c), and various ways to combine multiple forms of learning (Tuyls et al., 2022; Park et al., 2023; Xie et al., 2023).

Action space: thinking beyond external tools or actions. Although "action space" is a standard term in reinforcement learning, it is seldom used for language agents.[11] CoALA argues for defining a clear and task-suitable action space with both internal (reasoning, retrieval, learning) and external (grounding) actions, which will help systematize and inform the agent design. For example, as can be seen in Section 5, the size of the action space yields a tradeoff — a larger action space (e.g., Voyager, Generative Agents) means more agent capabilities, but also a harder decision making problem that requires more crafted decision code. Another important future direction is agent *safety* through the lens of its action space: from the CoALA categorization, it is clear that "learning" actions (especially procedural deletion and modification) could cause internal harm, while "grounding" actions (e.g., "rm" in bash terminal, harmful speech in human dialog, holding a knife in physical environments) could cause external harm. Preliminary safety measures have been

[7]`https://www.langchain.com`.

[8]`https://www.llamaindex.ai`.

[9]`https://github.com/guidance-ai/guidance`.

[10]`https://openai.com/blog/function-calling-and-other-api-updates`.

[11]Perhaps the only exception is ReAct (Yao et al., 2022b), which explicitly argues to "augment the action space with reasoning".

limited to task-specific heuristics (e.g., remove “os” operations in Python (Chen et al., 2021), filter keywords in dialog (Chowdhery et al., 2022; Driess et al., 2023), limit physical environments to controlled setups (Ahn et al., 2022)), but as agents are grounded to more complex and multi-facet environments (e.g., navigate both the Internet and city streets) with richer internal mechanisms, it will be necessary to clearly specify and ablate the agent’s action space for worst-case scenario prediction and prevention (Yao and Narasimhan, 2023).

Decision making: thinking beyond action generation. We believe one of the most exciting future directions for language agents is decision making: as detailed in Section 4.6, most works are still confined to proposing (or directly generating) a single action. Present agents have just scratched the surface of more deliberate, propose-evaluate-select decision making procedures analogous to classical planning and tree search (Yao et al., 2023; Hao et al., 2023; Liu et al., 2023a; Dagan et al., 2023), using toy tasks such as game of 24 or block building. It would be powerful to extend such schemes to more complicated tasks with grounding (Qin et al., 2023) and long-term memory; enable learning of decision making procedures; and reduce the cost of extensive reasoning to facilitate more deliberate planning (potentially via smaller task-specific models for proposal or evaluation, as discussed above). On the other hand, LLM development might be influenced or even shaped by the increased usage of reasoning toward complex decision making, and better support decision making by solving known issues such as over-confidence and miscalibration (Jiang et al., 2021; Braverman et al., 2020; Chen et al., 2022), misalignment with human values or bias (Liang et al., 2021; Feng et al., 2023), hallucinations in self-evaluation (Shinn et al., 2023), and lack of human-in-the-loop mechanisms in face of uncertainties (Nguyen et al., 2022a; Ren et al., 2023). In summary, more powerful and efficient LLMs will enable more deliberate language agents, and the agent-building experiences could in turn inform the improvement of base LLMs in the future.

7 Discussion

In this section, we discuss some conceptual questions raised by our proposal.

Internal vs. external actions: what is the boundary between an agent and its environment? While humans or robots are clearly separated from their embodied environment, the boundary of language agents has become vaguer as a result of their digital grounding. For example, is a Wikipedia database an internal semantic memory or an external digital environment (Yao et al., 2022b)? If an agent runs some code and improves it based on the execution feedback before submitting the final code (Shinn et al., 2023; Yang et al., 2023a), is the code execution a form of external grounding or internal simulation? If a method consists of proposal and evaluation prompts (Yao et al., 2023), should it be considered a single agent with deliberate decision making, or two simpler agents (proposer and evaluator) collaborating for task solving?

We suggest the boundary question can be answered in terms of **controllability** and **indispensability**. For example, Wikipedia is an external environment if constantly modified by other users, but an offline version that only the agent may write to can be considered an internal memory. If an agent explicitly stores a compiler and all needed packages to run code, it could be considered an internal reasoning action. Otherwise, running code in some external machine (that someone may hack) should be considered external grounding. Lastly, if the proposal and evaluation prompts are specifically designed for each other in a particular task, it seems more proper to package them as an indispensable whole; but if both parts are fairly capable of interacting with other agents or environments without always coupling, a multi-agent view is more fit. While these dilemmas are purely conceptual and subjective with respect to existing implementations, such conceptual understanding may be important for systematically designing more capable and complex agents in the future.

Planning vs. execution: how much should agents plan? Making a call to an LLM is both slow and computationally intensive. Relying on LLMs for decision-making thus requires balancing the cost of planning against the utility of the resulting improved plan. This has analogs to the “value of computation” studied in human metareasoning (Lieder and Griffiths, 2020; Callaway et al., 2022; Gershman et al., 2015). The present work fixes a search budget by specifying a depth of reasoning (Yao et al., 2023), but studies of humans suggest that they allocate computation adaptively (Russek et al., 2022). Future work should develop mechanisms to estimate the utility of planning and modify the decision procedure accordingly, i.e., learning update decision making procedures in the CoALA framework.

Learning vs. acting: how should agents continuously and autonomously learn? In the CoALA framework, learning is a result action of a decision making cycle just like grounding: the agent deliberately chooses to commit information to long-term memory. This is in contrast to most agents, which simply fix a learning schedule and only use decison making for external actions. Biological agents, however, do not have this luxury: they must balance learning against external actions in their lifetime, choosing when and what to learn (Mattar and Daw, 2018). More flexible language agents (Wang et al., 2023a; Park et al., 2023) would follow a similar design and treat learning on par with external actions. Learning could be proposed as a possible action during regular decision-making, allowing the agent to "defer" it until the appropriate time.

LLMs vs. code: where should agents rely on each? CoALA agents possess two forms of procedural memory: agent code (which specifies deterministic procedures) and LLM parameters (a large, stochastic production system). Agent code is interpretable and extensible, but inherently brittle and only capable of addressing situations the designer anticipates. In contrast, the LLM's parameters are much less interpretable, but it possesses significant flexibility and is capable of generating zero-shot solutions in completely new contexts (Huang et al., 2022b). CoALA thus suggests that good design uses agent code primarily to implement classic, generic planning algorithms – and relies heavily on the LLM for action proposal and evaluation.

It is also important to bear a developmental perspective for LLMs, for new capabilities emerge with scaling (Wei et al., 2022a). For example, earlier language models such as GPT-2 (Radford et al., 2019) would not enable flexible reasoning as in the CoALA framework and might need to work hand in hand with some learned value function for action proposal and evaluation respectively (Yao et al., 2020). While GPT-3 (Brown et al., 2020) unlocked flexible few-shot and zero-shot reasoning for the first time, it was only until GPT-4 (OpenAI, 2023) that language agents may more reliably self-evaluate (Saunders et al., 2022; Shinn et al., 2023; Yao et al., 2023) or self-refine (Madaan et al., 2023; Chen et al., 2023b) their reasoning. Would the future development of LLMs further reduce the need for coded rules and extra-learned models, and indicate changes to the CoALA framework? As a thought experiment, imagine GPT-N that could "simulate" memory, grounding, learning, and decision making in context: list all the possible actions, deliberately evaluate each one (even by simulating how the world will respond), while maintaining long-term memory explicitly in a very long context. Or even more boldly: can GPT-N just generate the next action by simulating these implicitly in neurons, without any intermediate reasoning in context? While these extreme cases seem unlikely for now, a longer context might downplay the importance of long-term memory and more powerful reasoning for internal evaluation and simulation might bolster planning and weaken the need for grounding feedback. LLMs may keep scaling beyond the fundamental limitations on biological beings like humans (Griffiths, 2020). CoALA, and cognitive science more generally, may still help systematically organize tasks where LLMs excel or show deficits – using humans as a gauge and target of alignment – to determine appropriate code-based procedures to complement a given LLM on a given task. Even in the most extreme case where GPT-N could accomplish all of the CoALA mechanisms in neurons, it may be helpful to leverage CoALA as a conceptual guide to discover and interpret GPT-N's implicit circuits.

8 Conclusion

We proposed Cognitive Architectures for Language Agents (CoALA), a conceptual framework to systematically understand and build language agents. Our framework draws inspiration from the rich history of symbolic artificial intelligence and cognitive science and connects decades-old insights to frontier research on large language models. We believe this approach provides a path towards developing more general and human-like artificial intelligence.

Acknowledgements

SY and KN acknowledge support from an Oracle Collaborative Research award and the National Science Foundation under Grant No. 2239363. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. SY is also supported by the Harold W. Dodds Fellowship from Princeton. TS is supported by the National Defense Science and Engineering (NDSEG) Graduate Fellowship Program.

References

S. Adams, I. Arel, J. Bach, R. Coop, R. Furlan, B. Goertzel, J. S. Hall, A. Samsonovich, M. Scheutz, M. Schlesinger, et al. Mapping the landscape of human-level artificial general intelligence. *AI magazine*, 33 (1):25–42, 2012.

M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, et al. Do as I can, not as I say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in Neural Information Processing Systems*, 35:23716–23736, 2022.

J. R. Anderson and C. Lebiere. The Newell test for a theory of cognition. *Behavioral and Brain Sciences*, 26 (5):587–601, 2003.

R. C. Atkinson and R. M. Shiffrin. Human memory: A proposed system and its control processes. In *Psychology of Learning and Motivation*, volume 2, pages 89–195. Elsevier, 1968.

A. D. Baddeley and G. Hitch. Working memory. In *Psychology of Learning and Motivation*, volume 8, pages 47–89. Elsevier, 1974.

Y. Bai, S. Kadavath, S. Kundu, A. Askell, J. Kernion, A. Jones, A. Chen, A. Goldie, A. Mirhoseini, C. McKinnon, et al. Constitutional AI: Harmlessness from AI feedback. *arXiv preprint arXiv:2212.08073*, 2022.

E. Biyik and M. Palan. Asking easy questions: A user-friendly approach to active reward learning. In *Proceedings of the 3rd Conference on Robot Learning*, 2019.

C. Blundell, B. Uria, A. Pritzel, Y. Li, A. Ruderman, J. Z. Leibo, J. Rae, D. Wierstra, and D. Hassabis. Model-free episodic control. *arXiv preprint arXiv:1606.04460*, 2016.

S. Borgeaud, A. Mensch, J. Hoffmann, T. Cai, E. Rutherford, K. Millican, G. B. Van Den Driessche, J.-B. Lespiau, B. Damoc, A. Clark, et al. Improving language models by retrieving from trillions of tokens. In *International Conference on Machine Learning*, pages 2206–2240, 2022.

S. Branavan, D. Silver, and R. Barzilay. Learning to win by reading manuals in a Monte-Carlo framework. *Journal of Artificial Intelligence Research*, 43:661–704, 2012.

M. Braverman, X. Chen, S. Kakade, K. Narasimhan, C. Zhang, and Y. Zhang. Calibration, entropy rates, and memory in language models. In *International Conference on Machine Learning*, pages 1089–1099, 2020.

A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, et al. RT-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.

C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

F. Callaway, B. van Opheusden, S. Gul, P. Das, P. M. Krueger, T. L. Griffiths, and F. Lieder. Rational use of cognitive resources in human planning. *Nature Human Behaviour*, 6(8):1112–1125, 2022.

C.-M. Chan, W. Chen, Y. Su, J. Yu, W. Xue, S. Zhang, J. Fu, and Z. Liu. Chateval: Towards better llm-based evaluators through multi-agent debate. *arXiv preprint arXiv:2308.07201*, 2023.

B. Chen, F. Xia, B. Ichter, K. Rao, K. Gopalakrishnan, M. S. Ryoo, A. Stone, and D. Kappler. Open-vocabulary queryable scene representations for real world planning. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11509–11522, 2023a.

D. Chen, A. Fisch, J. Weston, and A. Bordes. Reading Wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*, 2017.

M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

X. Chen, M. Lin, N. Schärli, and D. Zhou. Teaching large language models to self-debug. *arXiv preprint arXiv:2304.05128*, 2023b.

Y. Chen, L. Yuan, G. Cui, Z. Liu, and H. Ji. A close look into the calibration of pre-trained language models. *arXiv preprint arXiv:2211.00151*, 2022.

N. Chomsky. Three models for the description of language. *IRE Transactions on information theory*, 2(3): 113–124, 1956.

A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

M.-A. Côté, A. Kádár, X. Yuan, B. Kybartas, T. Barnes, E. Fine, J. Moore, M. Hausknecht, L. El Asri, M. Adada, et al. Textworld: A learning environment for text-based games. In *Computer Games: 7th Workshop, CGW 2018*, pages 41–75. Springer, 2019.

A. Creswell, M. Shanahan, and I. Higgins. Selection-inference: Exploiting large language models for interpretable logical reasoning. In *The Eleventh International Conference on Learning Representations*, 2023.

G. Dagan, F. Keller, and A. Lascarides. Dynamic planning with a llm. *arXiv preprint arXiv:2308.06391*, 2023.

I. Dasgupta, C. Kaeser-Chen, K. Marino, A. Ahuja, S. Babayan, F. Hill, and R. Fergus. Collaborating with language models for embodied reasoning. In *Second Workshop on Language and Reinforcement Learning*, 2022.

X. Deng, Y. Gu, B. Zheng, S. Chen, S. Stevens, B. Wang, H. Sun, and Y. Su. Mind2Web: Towards a generalist agent for the web. *arXiv preprint arXiv:2306.06070*, 2023.

N. Derbinsky, J. Li, and J. Laird. A multi-domain evaluation of scaling in a general episodic memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26, pages 193–199, 2012.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT (1)*, 2019.

D. Dohan, W. Xu, A. Lewkowycz, J. Austin, D. Bieber, R. G. Lopes, Y. Wu, H. Michalewski, R. A. Saurous, J. Sohl-Dickstein, et al. Language model cascades. *arXiv preprint arXiv:2207.10342*, 2022.

D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, et al. PALM-E: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

Y. Du, S. Li, A. Torralba, J. B. Tenenbaum, and I. Mordatch. Improving factuality and reasoning in language models through multiagent debate. *arXiv preprint arXiv:2305.14325*, 2023.

A. Ecoffet, J. Huizinga, J. Lehman, K. O. Stanley, and J. Clune. Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995*, 2019.

K. Ellis, C. Wong, M. Nye, M. Sablé-Meyer, L. Morales, L. Hewitt, L. Cary, A. Solar-Lezama, and J. B. Tenenbaum. Dreamcoder: Bootstrapping inductive program synthesis with wake-sleep library learning. In *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, pages 835–850, 2021.

S. Feng, C. Y. Park, Y. Liu, and Y. Tsvetkov. From pretraining data to language models to downstream tasks: Tracking the trails of political biases leading to unfair nlp models. *arXiv preprint arXiv:2305.08283*, 2023.

D. Ganguli, A. Askell, N. Schiefer, T. Liao, K. Lukošiūtė, A. Chen, A. Goldie, A. Mirhoseini, C. Olsson, D. Hernandez, et al. The capacity for moral self-correction in large language models. *arXiv preprint arXiv:2302.07459*, 2023.

C. Gao, X. Lan, Z. Lu, J. Mao, J. Piao, H. Wang, D. Jin, and Y. Li. S3: Social-network simulation system with large language model-empowered agents. *arXiv preprint arXiv:2307.14984*, 2023.

T. Gao, A. Fisch, and D. Chen. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*, 2020.

S. J. Gershman, E. J. Horvitz, and J. B. Tenenbaum. Computational rationality: A converging paradigm for intelligence in brains, minds, and machines. *Science*, 349(6245):273–278, 2015.

T. L. Griffiths. Understanding human intelligence through human limitations. *Trends in Cognitive Sciences*, 24(11):873–883, 2020.

I. Gur, H. Furuta, A. Huang, M. Safdari, Y. Matsuo, D. Eck, and A. Faust. A real-world webagent with planning, long context understanding, and program synthesis. *arXiv preprint arXiv:2307.12856*, 2023.

K. Guu, K. Lee, Z. Tung, P. Pasupat, and M. Chang. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938, 2020.

A. W. Hanjie, V. Zhong, and K. Narasimhan. Grounding language to entities and dynamics for generalization in reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2021.

S. Hao, Y. Gu, H. Ma, J. J. Hong, Z. Wang, D. Z. Wang, and Z. Hu. Reasoning with language model is planning with world model. *arXiv preprint arXiv:2305.14992*, 2023.

M. Hasan, C. Ozel, S. Potter, and E. Hoque. Sapien: Affective virtual agents powered by large language models. *arXiv preprint arXiv:2308.03022*, 2023.

P. Haslum, N. Lipovetzky, D. Magazzeni, C. Muise, R. Brachman, F. Rossi, and P. Stone. *An introduction to the planning domain definition language*, volume 13. Springer, 2019.

M. Hausknecht, P. Ammanabrolu, M.-A. Côté, and X. Yuan. Interactive fiction games: A colossal adventure. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7903–7910, 2020.

S. Hong, X. Zheng, J. Chen, Y. Cheng, C. Zhang, Z. Wang, S. K. S. Yau, Z. Lin, L. Zhou, C. Ran, et al. Metagpt: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*, 2023.

J. Huang, S. S. Gu, L. Hou, Y. Wu, X. Wang, H. Yu, and J. Han. Large language models can self-improve. *arXiv preprint arXiv:2210.11610*, 2022a.

S. Huang, Z. Jiang, H. Dong, Y. Qiao, P. Gao, and H. Li. Instruct2Act: Mapping Multi-modality Instructions to Robotic Actions with Large Language Model. *arXiv preprint arXiv:2305.11176*, 2023.

W. Huang, P. Abbeel, D. Pathak, and I. Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning*, pages 9118–9147, 2022b.

W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022c.

A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35, 2017.

G. Irving, P. Christiano, and D. Amodei. AI safety via debate. *arXiv preprint arXiv:1805.00899*, 2018.

G. Izacard, M. Caron, L. Hosseini, S. Riedel, P. Bojanowski, A. Joulin, and E. Grave. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*, 2021.

Z. Jiang, J. Araki, H. Ding, and G. Neubig. How can we know when language models know? on the calibration of language models for question answering. *Transactions of the Association for Computational Linguistics*, 9:962–977, 2021.

Z. Jin, S. Levine, F. G. Adauto, O. Kamal, M. Sap, M. Sachan, R. Mihalcea, J. B. Tenenbaum, and B. Schölkopf. When to make exceptions: Exploring language models as accounts of human moral judgment. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

S. Jinxin, Z. Jiabao, W. Yilei, W. Xingjiao, L. Jiawen, and H. Liang. Cgmi: Configurable general multi-agent interaction framework. *arXiv preprint arXiv:2308.12503*, 2023.

R. M. Jones, J. E. Laird, P. E. Nielsen, K. J. Coulter, P. Kenny, and F. V. Koss. Automated intelligent pilots for combat flight simulation. *AI magazine*, 20(1):27–27, 1999.

D. Jurafsky. *Speech & language processing*. Pearson Education India, 2000.

G. Kim, P. Baldi, and S. McAleer. Language models can solve computer tasks. *arXiv preprint arXiv:2303.17491*, 2023.

J. R. Kirk and J. E. Laird. Interactive task learning for simple games. *Advances in Cognitive Systems*, 3 (13-30):5, 2014.

J. R. Kirk, W. Robert, P. Lindes, and J. E. Laird. Improving Knowledge Extraction from LLMs for Robotic Task Learning through Agent Analysis. *arXiv preprint arXiv:2306.06770*, 2023.

K. R. Koedinger, J. R. Anderson, W. H. Hadley, M. A. Mark, et al. Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education*, 8(1):30–43, 1997.

T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa. Large language models are zero-shot reasoners. *Advances in Neural Information Processing Systems*, 35:22199–22213, 2022.

I. Kotseruba and J. K. Tsotsos. 40 years of cognitive architectures: core cognitive abilities and practical applications. *Artificial Intelligence Review*, 53(1):17–94, 2020.

J. E. Laird. *The Soar cognitive architecture*. MIT press, 2019.

J. E. Laird. Introduction to Soar. *arXiv preprint arXiv:2205.03854*, 2022.

J. E. Laird, P. S. Rosenbloom, and A. Newell. Chunking in Soar: The anatomy of a general learning mechanism. *Machine Learning*, 1:11–46, 1986.

J. E. Laird, A. Newell, and P. S. Rosenbloom. Soar: An architecture for general intelligence. *Artificial Intelligence*, 33(1):1–64, 1987.

J. E. Laird, K. R. Kinkade, S. Mohan, and J. Z. Xu. Cognitive robotics using the Soar cognitive architecture. In *CogRob @ AAAI*, 2012.

H. Le, Y. Wang, A. D. Gotmare, S. Savarese, and S. C. H. Hoi. Coderl: Mastering code generation through pretrained models and deep reinforcement learning. *Advances in Neural Information Processing Systems*, 35:21314–21328, 2022.

P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

R. Li, L. B. Allal, Y. Zi, N. Muennighoff, D. Kocetkov, C. Mou, M. Marone, C. Akiki, J. Li, J. Chim, Q. Liu, E. Zheltonozhskii, T. Y. Zhuo, T. Wang, O. Dehaene, M. Davaadorj, J. Lamy-Poirier, J. Monteiro, O. Shliazhko, N. Gontier, N. Meade, A. Zebaze, M.-H. Yee, L. K. Umapathi, J. Zhu, B. Lipkin, M. Oblokulov, Z. Wang, R. Murthy, J. Stillerman, S. S. Patel, D. Abulkhanov, M. Zocca, M. Dey, Z. Zhang, N. Fahmy, U. Bhattacharyya, W. Yu, S. Singh, S. Luccioni, P. Villegas, M. Kunakov, F. Zhdanov, M. Romero, T. Lee, N. Timor, J. Ding, C. Schlesinger, H. Schoelkopf, J. Ebert, T. Dao, M. Mishra, A. Gu, J. Robinson, C. J. Anderson, B. Dolan-Gavitt, D. Contractor, S. Reddy, D. Fried, D. Bahdanau, Y. Jernite, C. M. Ferrandis, S. M. Hughes, T. Wolf, A. Guha, L. von Werra, and H. de Vries. Starcoder: may the source be with you! *ArXiv*, abs/2305.06161, 2023.

Y. Li, D. H. Choi, J. Chung, N. Kushman, J. Schrittwieser, R. Leblond, Tom, Eccles, J. Keeling, F. Gimeno, A. D. Lago, T. Hubert, P. Choy, C. de, M. d'Autume, I. Babuschkin, X. Chen, P.-S. Huang, J. Welbl, S. Gowal, Alexey, Cherepanov, J. Molloy, D. J. Mankowitz, E. S. Robson, P. Kohli, N. de, Freitas, K. Kavukcuoglu, and O. Vinyals. Competition-level code generation with alphacode. *Science*, 378:1092 – 1097, 2022.

J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9493–9500, 2023a.

P. P. Liang, C. Wu, L.-P. Morency, and R. Salakhutdinov. Towards understanding and mitigating social biases in language models. In *International Conference on Machine Learning*, pages 6565–6576, 2021.

T. Liang, Z. He, W. Jiao, X. Wang, Y. Wang, R. Wang, Y. Yang, Z. Tu, and S. Shi. Encouraging divergent thinking in large language models through multi-agent debate. *arXiv preprint arXiv:2305.19118*, 2023b.

F. Lieder and T. L. Griffiths. Resource-rational analysis: Understanding human cognition as the optimal use of limited computational resources. *Behavioral and Brain Sciences*, 43:e1, 2020.

B. Y. Lin, Y. Fu, K. Yang, P. Ammanabrolu, F. Brahman, S. Huang, C. Bhagavatula, Y. Choi, and X. Ren. Swiftsage: A generative agent with fast and slow thinking for complex interactive tasks. *arXiv preprint arXiv:2305.17390*, 2023.

P. Lindes and J. E. Laird. Toward integrating cognitive linguistics and cognitive language processing. In *Proceedings of the 14th International Conference on Cognitive Modeling (ICCM)*, 2016.

B. Liu, Y. Jiang, X. Zhang, Q. Liu, S. Zhang, J. Biswas, and P. Stone. Llm+ p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*, 2023a.

J. Liu, D. Shen, Y. Zhang, B. Dolan, L. Carin, and W. Chen. What Makes Good In-Context Examples for GPT-3 ? *arXiv preprint arXiv:2101.06804*, 2021.

P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9), 2023b. ISSN 0360-0300.

R. Liu, J. Wei, S. S. Gu, T.-Y. Wu, S. Vosoughi, C. Cui, D. Zhou, and A. M. Dai. Mind's eye: Grounded language model reasoning through simulation. *arXiv preprint arXiv:2210.05359*, 2022.

Z. Ma, Y. Mei, and Z. Su. Understanding the benefits and challenges of using large language model-based conversational agents for mental well-being support. *arXiv preprint arXiv:2307.15810*, 2023.

A. Madaan, N. Tandon, P. Gupta, S. Hallinan, L. Gao, S. Wiegreffe, U. Alon, N. Dziri, S. Prabhumoye, Y. Yang, et al. Self-refine: Iterative refinement with self-feedback. *arXiv preprint arXiv:2303.17651*, 2023.

A. A. Markov. The theory of algorithms. *Trudy Matematicheskogo Instituta Imeni VA Steklova*, 42:3–375, 1954.

M. G. Mattar and N. D. Daw. Prioritized memory access explains planning and hippocampal replay. *Nature Neuroscience*, 21(11):1609–1617, 2018.

J. Meier, R. Rao, R. Verkuil, J. Liu, T. Sercu, and A. Rives. Language models enable zero-shot prediction of the effects of mutations on protein function. *bioRxiv*, 2021.

G. Mialon, R. Dessì, M. Lomeli, C. Nalmpantis, R. Pasunuru, R. Raileanu, B. Rozière, T. Schick, J. Dwivedi-Yu, A. Celikyilmaz, et al. Augmented language models: a survey. *arXiv preprint arXiv:2302.07842*, 2023.

S. Mohan and J. Laird. Learning goal-oriented hierarchical tasks from situated interactive instruction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28, 2014.

S. Mohan, A. H. Mininger, J. R. Kirk, and J. E. Laird. Acquiring grounded representations of words with situated interactive instruction. *Advances in Cognitive Systems*, 2:113–130, 2012.

R. Nakano, J. Hilton, S. Balaji, J. Wu, L. Ouyang, C. Kim, C. Hesse, S. Jain, V. Kosaraju, W. Saunders, et al. WebGPT: Browser-Assisted Question-Answering with Human Feedback. *arXiv preprint arXiv:2112.09332*, 2021.

K. Narasimhan, R. Barzilay, and T. Jaakkola. Deep transfer in reinforcement learning by language grounding. In *Journal of Artificial Intelligence Research (JAIR)*, 2018.

S. Nason and J. E. Laird. Soar-RL: Integrating reinforcement learning with Soar. *Cognitive Systems Research*, 6(1):51–59, 2005.

A. Newell. Studies in problem solving: Subject 3 on the crypt-arithmetic task DONALD+ GERALD= ROBERT. Technical report, Carnegie Mellon University, 1967.

A. Newell. Physical symbol systems. *Cognitive science*, 4(2):135–183, 1980.

A. Newell. Précis of unified theories of cognition. *Behavioral and Brain Sciences*, 15(3):425–437, 1992.

A. Newell and H. A. Simon. *Human problem solving.* Prentice-Hall, 1972.

A. Newell, P. S. Rosenbloom, and J. E. Laird. Symbolic architectures for cognition. *Foundations of cognitive science*, pages 93–131, 1989.

K. Nguyen, Y. Bisk, and H. Daumé III. A framework for learning to request rich and contextually useful information from humans. In *ICML*, July 2022a.

K. X. Nguyen, D. Misra, R. Schapire, M. Dudík, and P. Shafto. Interactive learning from activity description. In *International Conference on Machine Learning*, pages 8096–8108, 2021.

K. X. Nguyen, Y. Bisk, and H. D. Iii. A framework for learning to request rich and contextually useful information from humans. In *International Conference on Machine Learning*, pages 16553–16568, 2022b.

T. T. Nguyen, T. T. Huynh, P. L. Nguyen, A. W.-C. Liew, H. Yin, and Q. V. H. Nguyen. A survey of machine unlearning. *arXiv preprint arXiv:2209.02299*, 2022c.

A. Ni, S. Iyer, D. Radev, V. Stoyanov, W.-t. Yih, S. Wang, and X. V. Lin. Lever: Learning to verify language-to-code generation with execution. In *International Conference on Machine Learning*, pages 26106–26128, 2023.

N. J. Nilsson. Shakey the robot. *Technical Note*, 1984.

R. Nogueira, W. Yang, J. Lin, and K. Cho. Document expansion by query prediction, 2019.

A. M. Nuxoll and J. E. Laird. Extending cognitive architecture with episodic memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1560–1564, 2007.

OpenAI. Gpt-4 technical report. *ArXiv*, abs/2303.08774, 2023.

L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

N. D. Palo, A. Byravan, L. Hasenclever, M. Wulfmeier, N. Heess, and M. Riedmiller. Towards a unified agent with foundation models. In *Workshop on Reincarnating Reinforcement Learning at ICLR 2023*, 2023.

A. Parisi, Y. Zhao, and N. Fiedel. Talm: Tool augmented language models. *arXiv preprint arXiv:2205.12255*, 2022.

J. S. Park, J. C. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein. Generative agents: Interactive simulacra of human behavior. *arXiv preprint arXiv:2304.03442*, 2023.

P. Pataranutaporn, V. Danry, J. Leong, P. Punpongsanon, D. Novy, P. Maes, and M. Sra. AI-generated characters for supporting personalized learning and well-being. *Nature Machine Intelligence*, 3(12):1013–1022, 2021.

E. L. Post. Formal reductions of the general combinatorial decision problem. *American Journal of Mathematics*, 65(2):197–215, 1943.

A. Pritzel, B. Uria, S. Srinivasan, A. P. Badia, O. Vinyals, D. Hassabis, D. Wierstra, and C. Blundell. Neural episodic control. In *International conference on machine learning*, pages 2827–2836, 2017.

C. Qian, X. Cong, C. Yang, W. Chen, Y. Su, J. Xu, Z. Liu, and M. Sun. Communicative agents for software development. *arXiv preprint arXiv:2307.07924*, 2023.

Y. Qin, S. Liang, Y. Ye, K. Zhu, L. Yan, Y. Lu, Y. Lin, X. Cong, X. Tang, B. Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*, 2023.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

A. Z. Ren, A. Dixit, A. Bodrova, S. Singh, S. Tu, N. Brown, P. Xu, L. Takayama, F. Xia, J. Varley, et al. Robots that ask for help: Uncertainty alignment for large language model planners. *arXiv preprint arXiv:2307.01928*, 2023.

O. J. Romero, J. Zimmerman, A. Steinfeld, and A. Tomasic. Synergistic integration of large language models and cognitive architectures for robust ai: An exploratory analysis. *arXiv preprint arXiv:2308.09830*, 2023.

B. Rozière, J. Gehring, F. Gloeckle, S. Sootla, I. Gat, X. Tan, Y. Adi, J. Liu, T. Remez, J. Rapin, A. Kozhevnikov, I. Evtimov, J. Bitton, M. P. Bhatt, C. C. Ferrer, A. Grattafiori, W. Xiong, A. D'efossez, J. Copet, F. Azhar, H. Touvron, L. Martin, N. Usunier, T. Scialom, and G. Synnaeve. Code llama: Open foundation models for code. *ArXiv*, abs/2308.12950, 2023.

O. Rubin, J. Herzig, and J. Berant. Learning to retrieve prompts for in-context learning. *arXiv preprint arXiv:2112.08633*, 2021.

E. Russek, D. Acosta-Kane, B. van Opheusden, M. G. Mattar, and T. Griffiths. Time spent thinking in online chess reflects the value of computation. *PsyArXiv*, 2022.

D. Sadigh, A. D. Dragan, S. Sastry, and S. A. Seshia. Active preference-based learning of reward functions. In N. M. Amato, S. S. Srinivasa, N. Ayanian, and S. Kuindersma, editors, *Robotics: Science and Systems XIII*, 2017.

W. Saunders, C. Yeh, J. Wu, S. Bills, L. Ouyang, J. Ward, and J. Leike. Self-critiquing models for assisting human evaluators. *arXiv preprint arXiv:2206.05802*, 2022.

T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, L. Zettlemoyer, N. Cancedda, and T. Scialom. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023.

T. Shi, A. Karpathy, L. Fan, J. Hernandez, and P. Liang. World of Bits: An Open-Domain platform for web-based agents. In *International Conference on Machine Learning*, pages 3135–3144, 2017.

N. Shinn, F. Cassano, B. Labash, A. Gopinath, K. Narasimhan, and S. Yao. Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366*, 2023.

M. Shridhar, X. Yuan, M.-A. Côté, Y. Bisk, A. Trischler, and M. Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020.

I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg. Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11523–11530, 2023.

T. Sumers, R. Hawkins, M. K. Ho, T. Griffiths, and D. Hadfield-Menell. How to talk so AI will learn: Instructions, descriptions, and autonomy. *Advances in Neural Information Processing Systems*, 35:34762–34775, 2022.

T. Sumers, K. Marino, A. Ahuja, R. Fergus, and I. Dasgupta. Distilling internet-scale vision-language models into embodied agents. In *Proceedings of the 40th International Conference on Machine Learning*, pages 32797–32818, 2023.

T. R. Sumers, M. K. Ho, R. D. Hawkins, K. Narasimhan, and T. L. Griffiths. Learning rewards from linguistic feedback. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 6002–6010, 2021.

R. Sun. Desiderata for cognitive architectures. *Philosophical Psychology*, 17(3):341–373, 2004.

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction.* MIT press, 2018.

O. Tafjord, B. Dalvi, and P. Clark. Proofwriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3621–3634, 2021.

M. Tambe, W. L. Johnson, R. M. Jones, F. Koss, J. E. Laird, P. S. Rosenbloom, and K. Schwamb. Intelligent agents for interactive simulation environments. *AI magazine*, 16(1):15–15, 1995.

M. Tang, S. Yao, J. Yang, and K. Narasimhan. Referral augmentation for zero-shot information retrieval, 2023a.

Q. Tang, Z. Deng, H. Lin, X. Han, Q. Liang, and L. Sun. ToolAlpaca: Generalized Tool Learning for Language Models with 3000 Simulated Cases. *arXiv preprint arXiv:2306.05301*, 2023b.

S. Tellex, T. Kollar, S. Dickerson, M. Walter, A. Banerjee, S. Teller, and N. Roy. Understanding natural language commands for robotic navigation and mobile manipulation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 25, pages 1507–1514, 2011.

J. Tuyls, S. Yao, S. Kakade, and K. Narasimhan. Multi-stage episodic control for strategic exploration in text games. *arXiv preprint arXiv:2201.01251*, 2022.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

G. Wang, Y. Xie, Y. Jiang, A. Mandlekar, C. Xiao, Y. Zhu, L. Fan, and A. Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023a.

L. Wang, C. Ma, X. Feng, Z. Zhang, H. Yang, J. Zhang, Z. Chen, J. Tang, X. Chen, Y. Lin, W. X. Zhao, Z. Wei, and J.-R. Wen. A survey on large language model based autonomous agents, 2023b.

L. Wang, N. Yang, and F. Wei. Query2doc: Query expansion with large language models. *arXiv preprint arXiv:2303.07678*, 2023c.

R. Wang, P. Jansen, M.-A. Côté, and P. Ammanabrolu. Scienceworld: Is your agent smarter than a 5th grader? *arXiv preprint arXiv:2203.07540*, 2022a.

X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, and D. Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022b.

J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, E. H. Chi, T. Hashimoto, O. Vinyals, P. Liang, J. Dean, and W. Fedus. Emergent abilities of large language models. *Transactions on Machine Learning Research*, 2022a. ISSN 2835-8856. Survey Certification.

J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. Chi, Q. Le, and D. Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022b.

L. Weng. Llm-powered autonomous agents. *lilianweng.github.io*, Jun 2023. URL `https://lilianweng.github.io/posts/2023-06-23-agent/`.

J. Weston, S. Chopra, and A. Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.

T. Winograd. Understanding natural language. *Cognitive psychology*, 3(1):1–191, 1972.

R. E. Wray, J. R. Kirk, J. E. Laird, et al. Language models as a knowledge source for cognitive agents. *arXiv preprint arXiv:2109.08270*, 2021.

Q. Wu, G. Bansal, J. Zhang, Y. Wu, S. Zhang, E. Zhu, B. Li, L. Jiang, X. Zhang, and C. Wang. Autogen: Enabling next-gen llm applications via multi-agent conversation framework. *arXiv preprint arXiv:2308.08155*, 2023.

T. Wu, E. Jiang, A. Donsbach, J. Gray, A. Molina, M. Terry, and C. J. Cai. Promptchainer: Chaining large language model prompts through visual programming. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts*, pages 1–10, 2022a.

T. Wu, M. Terry, and C. J. Cai. AI chains: Transparent and controllable human-AI interaction by chaining large language model prompts. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pages 1–22, 2022b.

Y. Xie, T. Xie, M. Lin, W. Wei, C. Li, B. Kong, L. Chen, C. Zhuo, B. Hu, and Z. Li. Olagpt: Empowering llms with human-like problem-solving abilities. *arXiv preprint arXiv:2305.16334*, 2023.

B. Xu, X. Liu, H. Shen, Z. Han, Y. Li, M. Yue, Z. Peng, Y. Liu, Z. Yao, and D. Xu. Gentopia: A collaborative platform for tool-augmented llms. *arXiv preprint arXiv:2308.04030*, 2023a.

B. Xu, Z. Peng, B. Lei, S. Mukherjee, Y. Liu, and D. Xu. Rewoo: Decoupling reasoning from observations for efficient augmented language models. *arXiv preprint arXiv:2305.18323*, 2023b.

B. Xu, A. Yang, J. Lin, Q. Wang, C. Zhou, Y. Zhang, and Z. Mao. ExpertPrompting: Instructing Large Language Models to be Distinguished Experts. *arXiv preprint arXiv:2305.14688*, 2023c.

J. Yang, A. Prabhakar, K. Narasimhan, and S. Yao. Intercode: Standardizing and benchmarking interactive coding with execution feedback. *arXiv preprint arXiv:2306.14898*, 2023a.

S. Yang, O. Nachum, Y. Du, J. Wei, P. Abbeel, and D. Schuurmans. Foundation models for decision making: Problems, methods, and opportunities. *arXiv preprint arXiv:2303.04129*, 2023b.

S. Yao and K. Narasimhan. Language agents in the digital world: Opportunities and risks. *princeton-nlp.github.io*, Jul 2023. URL `https://princeton-nlp.github.io/language-agent-impact/`.

S. Yao, R. Rao, M. Hausknecht, and K. Narasimhan. Keep CALM and explore: Language models for action generation in text-based games. *arXiv preprint arXiv:2010.02903*, 2020.

S. Yao, H. Chen, J. Yang, and K. Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022a.

S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022b.

S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023.

E. Zelikman, Y. Wu, J. Mu, and N. Goodman. STaR: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

A. Zeng, M. Attarian, B. Ichter, K. Choromanski, A. Wong, S. Welker, F. Tombari, A. Purohit, M. Ryoo, V. Sindhwani, et al. Socratic models: Composing zero-shot multimodal reasoning with language. *arXiv preprint arXiv:2204.00598*, 2022.

Y. Zhang, S. Sun, M. Galley, Y.-C. Chen, C. Brockett, X. Gao, J. Gao, J. Liu, and W. B. Dolan. Dialogpt: Large-scale generative pre-training for conversational response generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 270–278, 2020.

V. Zhong, A. W. Hanjie, S. Wang, K. Narasimhan, and L. Zettlemoyer. SILG: The Multi-domain Symbolic Interactive Language Grounding Benchmark. *Advances in Neural Information Processing Systems*, 34: 21505–21519, 2021.

H. Zhou, M. Huang, T. Zhang, X. Zhu, and B. Liu. Emotional chatting machine: Emotional conversation generation with internal and external memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

S. Zhou, U. Alon, F. F. Xu, Z. Jiang, and G. Neubig. Docprompting: Generating code by retrieving the docs. In *The Eleventh International Conference on Learning Representations*, 2022a.

S. Zhou, F. F. Xu, H. Zhu, X. Zhou, R. Lo, A. Sridhar, X. Cheng, Y. Bisk, D. Fried, U. Alon, et al. WebArena: A Realistic Web Environment for Building Autonomous Agents. *arXiv preprint arXiv:2307.13854*, 2023.

Y. Zhou, A. I. Muresanu, Z. Han, K. Paster, S. Pitis, H. Chan, and J. Ba. Large language models are human-level prompt engineers. *arXiv preprint arXiv:2211.01910*, 2022b.