Reasoning with Large Language Models, a Survey

ASKE PLAAT, ANNIE WONG, SUZAN VERBERNE, JOOST BROEKENS, NIKI VAN STEIN, and THOMAS BÄCK, Leiden University, Netherlands

Language models with billions of parameters exhibit in-context learning abilities, enabling few-shot learning on tasks that the model was not specifically trained for. Although the models achieve breakthrough performance on language tasks, traditional models do not perform well on reasoning benchmarks. However, a new prompt-based approach, Chain-of-thought, has demonstrated strong reasoning abilities on these benchmarks.

The research on LLM reasoning abilities started with the question whether LLMs can solve grade school math word problems, and has expanded to other tasks in the past few years. This paper reviews the field of prompt-based reasoning with LLMs. We propose a taxonomy that identifies different ways to generate, evaluate, and control multi-step reasoning. We provide an in-depth coverage of core approaches and open problems, and we propose a research agenda for the near future. We highlight the relation between reasoning and reinforcement learning, and find that self-improvement and self-reflection are possible through the judicious use of prompts. True self-improvement and self-reflection, to go from reasoning *with* LLMs to reasoning *by* LLMs, remains future work.

CCS Concepts: • **Computing methodologies** → **Natural language processing**.

ACM Reference Format:
Aske Plaat, Annie Wong, Suzan Verberne, Joost Broekens, Niki van Stein, and Thomas Bäck. 2025. Reasoning with Large Language Models, a Survey. 1, 1 (February 2025), 35 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Transformer-based Large Language Models (LLMs) that are trained on large datasets have achieved breakthrough performance at text generation tasks that directly build on next token prediction [103, 135, 145]; they are very good at natural language understanding (GLUE, SQUAD, Xsum) [89, 107, 138, 139], translation [67, 94, 116], question answering [127], and other language generation tasks. The success of models such as ChatGPT [92] is impressive.

Transformer-based generative language models whose size is beyond hundreds of billions parameters are not only good at language generation, they also enable a new type of machine learning, called *in-context learning* [14]. In-context learning, also known as prompt-based learning, is an emergent ability that occurs in LLMs beyond a certain size (hundreds of billions of parameters—less, with judicious prompting) that have been finetuned for conversational responses [145]. In-context learning is inference-time, prompt-based, few-shot learning with instructions. As opposed to finetuning, model parameters are not adapted by in-context learning.

Language generation tasks are solved well by LLMs with prompt-based learning. On the other hand, reasoning tasks, such as grade school math word problems, are more difficult for LLMs [27]. Spurred-on by the impressive performance on language tasks, much research has focused on understanding the reason for the poor performance of LLMs on reasoning tasks, and how it can be improved. Among this research, the Chain-of-thought paper by Wei et al. [146] stands

Authors' address: Aske Plaat; Annie Wong; Suzan Verberne; Joost Broekens; Niki van Stein; Thomas Bäck, LIACS, Leiden University, Netherlands.

Manuscript submitted to ACM

out. This work, and later work by Kojima et al. [68], showed that adding a simple instruction to the prompt, *Let's think step by step*, can provoke an LLM to perform the required intermediate reasoning steps. Subsequently, performance on benchmarks has increased markedly. Much of the performance increase of models such as OpenAI o1, o3 and DeepSeek R1 is attributed to reasoning methods as reviewed here, with reinforcement learning playing an increasingly important role [23, 33, 46, 58, 117, 149, 154].

The line of research into LLM-reasoning was initiated with grade school math word problems, with the GSM8K benchmark [27]. Soon, other reasoning domains were included, such as reasoning about computer code, robotic movement, and games. Many works have been published that build on Chain-of-thought [25]. In this paper, we survey the literature using a straightforward taxonomy. We discuss papers based on several reasoning benchmarks, and directly-related follow up work on reasoning.

While other works focus on the nature of reasoning and its definition [24, 55, 160], and on evaluating reasoning in LLMs [88], we focus on prompt-based reasoning algorithms and provide a research agenda with opportunities for future research. At the end of this survey, we also discuss connections to other fields, such as self-reflection and metacognition (or thinking about thinking [34]).

The main contributions of this paper are: (1) we provide a survey of relevant approaches in prompt-based reasoning with LLMs on grade school math word problems and closely related reasoning domains, (2) we propose a taxonomy based on the reasoning literature (step generation, step evaluation, and control of reasoning steps), and (3) we formulate a research agenda for reasoning with LLMs.

1.1 Selection of Papers and Organization of the Survey

The papers in this survey were selected as follows. We started by selecting papers on the ability to solve math word benchmarks (as a proxy for reasoning ability), that contained the search terms *reasoning* and *large language model* in their title or abstract, with a focus on papers that reference the Chain-of-thought paper. Reasoning with LLMs initially was aimed at solving math world problems. It is now wider, including benchmarks and approaches for computer code, game play, puzzles, robot movement, and webpage navigation (see Table 2). We selected recent papers (two years prior to the writing of the survey) that show experimental results on selected benchmark datasets.

We focus on prompt-based, in-context learning, methods based on Chain-of thought, that are used in reasoning LLMs such as OpenAI o1 and o3 [58, 149, 154]. We also include papers that work by finetuning or supervised learning that have contributed to the Chain-of-thought approaches.

This survey is organized as follows. Section 2 provides background information on the most relevant developments in LLMs, including in-context learning. Of great importance are the benchmarks that are used in this field (Section 2.3). Next, in Section 3 we provide a taxonomy of the field, where we discuss the approaches in detail. Then, in Section 4 we discuss our findings in a broader perspective. We also discuss the relation between reasoning and work on self-reflection and metacognition. This section concludes with an agenda for future research. Finally, Section 5 concludes the survey.

2 BACKGROUND: REASONING WITH LLMS

Reasoning has a long history in AI, in logical inference, and in other fields, such as commonsense reasoning. Before we dive into the works on reasoning, we review some background terminology on LLMs. Our overview is brief. Excellent recent general surveys on LLMs are, for example, Minaee et al. [86] and Zhao et al. [166]. We discuss the generic training pipeline for LLMs, we discuss how in-context learning works, and we discuss commonly used benchmarks. We start with the generic language model training pipeline.

2.1 Language Model Training Pipeline

LLMs are typically constructed in a sequence of stages, from data preparation, through training, to inference. The training pipeline for most LLMs is quite elaborate. We will now list a brief pipeline of the most commonly used stages, based on the survey by Minaee et al. [86].

In training an LLM, the first stage is to *acquire* a large, general, unlabeled, high-quality text corpus. Some considerations on the selection of the texts are discussed in Brown et al. [14]. The next stage is *pretraining* the transformer model [135] on this large corpus. This stage yields a generative language model. Pretraining is done using a self-supervised autoregressive approach on the unlabeled dataset (text corpus). Then the general model is *finetuned* to a specific (narrow) task. This can be done using supervised learning with a new labeled dataset consisting of prompts and answers (supervised finetuning, SFT) [86, 145], or reinforcement learning [23, 125], specific for the task at hand. A small number of the surveyed papers also address the finetuning stage. A specific form of finetuning is *instruction tuning*, to improve instruction following for a certain task. Instruction tuning is supervised by a labeled dataset of instruction prompts and corresponding outputs. Depending on the purpose of the model the next step is *alignment* of the finetuned model with user expectations (preference alignment). Alignment is usually performed to ensure that the model produces more ethically and socially acceptable answers. The machine learning method that is commonly used in this stage Reinforcement Learning with Human Feedback [92] or Direct Preference Optimization [105]. Optionally, model training can be *optimized* to improve cost-effectiveness, for example, with low-rank optimization [53], mixed precision training [85], quantization [61], or knowledge distillation [45, 155].

Once the model has been trained in the steps described above, its behavior can be further specialized during the *inference* stage. Here, the model is used to provide an answer to the prompt. The inference-time stage is post-training, no model parameters are changed anymore [14, 32]; *in-context learning*, or *prompt-learning*, takes place in this stage. This is the stage on which most of the surveyed papers focus, using prompts for the LLM to perform a complex multi-step reasoning task. The following section provides a brief introduction to in-context learning.

2.2 In-Context Learning

In large models, beyond hundreds of billions of parameters, a new kind of learning has emerged, that has been called *in-context learning* or *prompt-learning* [14]. It occurs not when the model is trained, but when it is used, at inference time. Since no parameters are changed in this stage, it is not a *model training* stage; in-context-learning "learns" from information that is already encoded in the trained model parameters and the context in the prompt, not by training the model anymore. In-context learning is often able to give good results with few examples, so-called *few-shot* learning. The large size of the model, containing rich and general knowledge, is enabling the few-shot learning (see Dong et al. [32] for a survey).

In in-context learning, a prompt, consisting of a piece of demonstration context, is concatenated with a query question, and is given to the language model, for prediction [79]. For example, when the task is emotion recognition in a social media post, "I missed the bus today," can be followed by "I felt so [___]", and the model could answer with "bad". Alternatively, for translation, we could follow "I missed the bus today," by "French: [___]" to request a translation [79]. The prompt contains background information that is recognized by the model, selecting the desired model context. In-context learning works when language models contain enough knowledge, allowing them to generalize on the (few) examples provided in the prompt.

Prompts that contain a few examples are said to perform few-shot learning. Prompts that contain only instructions with zero examples are said to perform zero-shot learning. In-context learning takes place at inference time, after the computationally intensive training stages where parameters have been pretrained and finetuned, when the model is queried by the user to provide answers. No parameters are changed anymore with in-context learning. This is quite different from the common approach in supervised deep learning—or self-supervised deep learning—where large datasets are used during training to update model parameters with backward propagation in lengthy and costly training epochs [44]. Indeed, in-context learning takes place fully at inference time, no parameters are trained, instead, *learning* now refers to adjusting the answers to the context of the prompt and the internal knowledge acquired during training. Common approaches to few-shot learning, such as metalearning, do include training and finetuning of parameters to achieve generalization, and are computationally expensive (see, for example, [38] or [52, 59] for a survey). In-context learning, in comparison, is computationally cheap.

Prompts provide a user-friendly interface to LLMs. The success of in-context learning tends to be quite sensitive to the way a prompt is formulated; a new field called *prompt engineering* has emerged to help human users learn how to make LLMs do what we want them to do [43, 103, 112, 145].

2.3 Reasoning Benchmarks

Progress in artificial intelligence is measured by benchmarks. Benchmarks define the goal that researchers aim to achieve in their experiments. In natural language processing, a wide array of benchmarks exist to measure progress, such as on question answering (for example, CommonsenseQA [126]), word prediction (for example, LAMBADA [93]), translation (for example, WMT'22 [67]), language understanding (for example, GLUE [138, 139]), and text summarization (for example, Xsum [89]). Transformer architectures were first popularized by encoder models such as BERT [30], for understanding tasks, such as named entity recognition and classification. Subsequently, decoder models such as GPT 2-4 [1, 14, 103] showed impressive progress on natural language generation tasks.

The field of LLMs is quite active. Many different benchmarks exist, and providing a comprehensive overview of all relevant benchmarks is beyond the scope of this survey. We will mention relevant benchmarks for testing the reasoning abilities of LLMs. The research on reasoning with LLMs started with math word problem benchmarks. The benchmark that is most frequently associated with reasoning with LLMs is a dataset of grade school math word problems GSM8K [27]. GSM8K was created with the aim of providing high quality, high diversity, moderate difficulty, problems and solutions in natural language. It consists of 8500 human-written math problems. Language models struggled to achieve good performance on this dataset before Chain-of-thought was introduced. An example of a math word task follows. *Problem:* Beth bakes 4 trays with two dozen batches of cookies in a week. If these cookies are shared amongst 16 people equally, how many cookies does each person consume? *Answer:* $4 \times 2 \times 12/16 = 6$.

Other benchmarks of similar math word problems are the SVAMP varying structures benchmarks [95], the ASDiv dataset of diverse math problems [84], the AQuA dataset of algebraic word problems [76], and the MAWPS benchmark [69]. Table 1 summarizes the accuracy of Chain-of-thought on these basic math word problems, against the baseline of GPT-3 175B as LLM [146], as percentage of benchmark questions answered correctly. We see that Chain-of-thought performs well against the baseline of GPT-3 on some benchmarks, but there is certainly room for further improvement on others.

In addition to the initial set of math word benchmarks, further reasoning approaches have been introduced that test performance in other fields of reasoning. Benchmarks have been developed on Computer code comprehension (Human evaluation, Spider [162], Transcoder [110]), Robotic movement (Alfworld [122], Kitchen [3]), Puzzle solving (Game24

Benchmark	GPT-3 175B	Chain-of-thought [146]
GSM8K	15.6	46.9
ASDiv	70.3	71.3
MAWPS	72.7	87.1
SVAMP	65.7	68.9
AQuA	24.8	35.8

Table 1. Accuracy of GPT-3 and Chain-of-thought on popular Math Word Problems benchmarks [146]

[159]), Creative writing [159]), Gaming (Checkmate problems[156], MineCraft [36]), and Webpage navigation (WebShop [157]). These benchmarks are used by other approaches in our survey, as we will see in more detail in Section 3.

3 PROMPT GENERATION, EVALUATION AND CONTROL

This survey examines how LLMs based on the transformer architecture can be prompted to solve reasoning tasks. The Chain-of-thought paper shows how a simple command could prompt an LLM to perform reasoning steps, yielding much better performance in math word problems. Since then much research has further explored this approach, trying to build stronger general problem solvers for other types of reasoning problems.

A typical approach to solve a complex problem is to subdivide it into smaller steps and to solve those. This approach is related to classical divide and conquer [6]. It consists of three stages. New steps are (1) *generated*, (2) *evaluated*, and the search of the generated steps is (3) *controlled* in some way. The in-context reasoning approaches that we survey follow a general three-stage pipeline [82]:

(1) **Generate**: prompt the model for the generation of steps,
(2) **Evaluate**: prompt for evaluation of the generated steps,
(3) **Control**: prompt for control of the number of steps that are generated and how deep ahead the reasoning process will look.

This three-stage pipeline is the basis of our taxonomy. Other taxonomies focus on LLM in general [86, 166], on reinforcement learning in reasoning [154], on reasoning in language [160], or on Chain-of-thought itself [25]. We will now discuss the three stages more deeply; for ease of reference they are numbered according to the Subsection in which they are described in more detail (3.1, 3.2, 3.3). Please also refer to Figure 1, or Table 2, for a diagram of the categories and subcategories of different approaches for the generation, evaluation, and control of reasoning steps.[1]

(3.1) Generation. The first stage is to create a prompt that instructs the LLM to generate reasoning steps. The problem must be split into substeps. This can be achieved with a problem-specific prompt that contains elements of the problem, such as: "First calculate how many marbles Mary had originally, then how many her friend had, and finally how many they had together." In general, it is possible to prompt an LLM to fill in the blanks in a step-by-step fashion. In the papers that we discuss, there are three main approaches for generating the step-by-step prompt, numbered with the Subsection in which the approaches are described. First, the prompt may be handcrafted for the problem by the researchers: *(3.1.1) hand-written prompt.* Second, the prompt or prompts may come from a source that is external to the model, such as another model or dataset: *(3.1.2) external knowledge-based prompt.* Third, the model itself can be prompted to generate a

[1]We show the approaches in the Figure in their main category only. Some approaches show innovations in two categories, and are shown twice. (Since all approaches have a generation, an evaluation, and a control aspect, all could in principle occur three times, and all three columns can be found in Table 2).

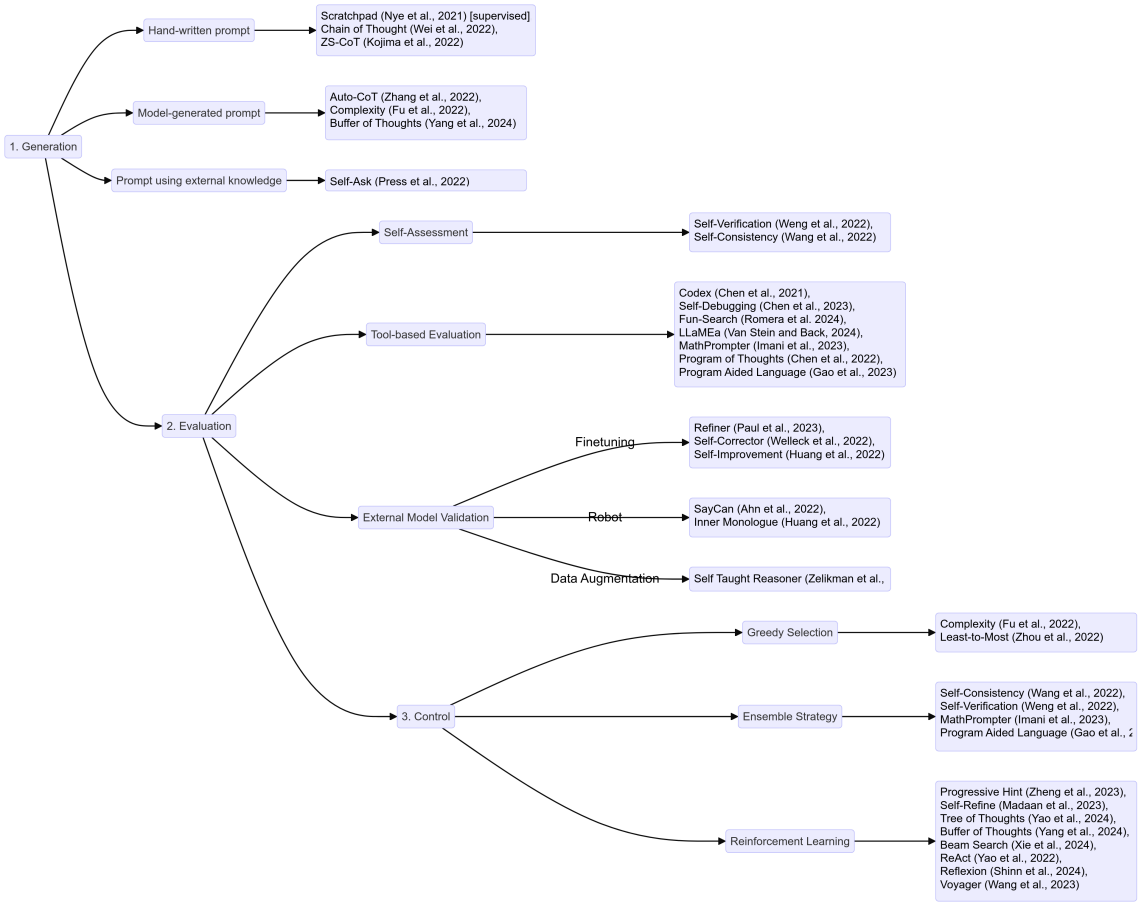

Fig. 1. Taxonomy of LLM-Reasoning Approaches: Prompt Generation, Evaluation, and Control

(series of) prompt(s) to analyze the problem *(3.1.3) model-generated prompt*. As we will see, all three approaches have their advantages and disadvantages.

Generating the subproblem-steps is the first stage that is necessary for in-context learning to perform reasoning. Each paper in our survey performs at least this stage of the reasoning pipeline. In some of the early papers (around 2022) it is the only stage of the pipeline that is performed.

(3.2) Evaluation. After the prompt has been generated and the model has answered it, the next stage is to evaluate this answer. Again, we see three main approaches for substep evaluation. First, the steps may be evaluated by the model itself: *(3.2.1) self-assessment*. Second, an external program can be used to evaluate the steps. For example, when the steps are expressed as computer code, an external interpreter or compiler can be used to check the validity and the outcome: *(3.2.2) tool-based evaluation*. Finally, an external model can be used, LLM or otherwise. For example, in robotics, an external physics model can determine if certain actions are physically possible: *(3.2.3) external model validation*.

Table 2. Taxonomy of approaches: Generation, Evaluation, and Control. Reported benchmark results: '=' is absolute score, '+' is improvement to a baseline

Approach	Domain	(3.1) Step generation	(3.2) Step evaluation	(3.3) Step control	Result
Scratchpad [91]	math word	**hand-wr/superv**	-	greedy/1prompt	PolyEval +19%, Python +21%
Chain-of-thought [146]	math word	**hand-written**	-	greedy/1prompt	GSM8K +39%, SVAMP +10%, ASDiv +2%, AQuA +11%, MAWPS +14%, CSQA +1.8%, StrategyQA +0.2%
ZS-CoT [68]	math word	**hand-written**	-	greedy/1prompt	MultiArith =89%, GSM8K =70%
Auto-CoT [165]	math word	**model-generated**	-	clustering	MultiArith +0.3%, GSM8K +1%, AddSub +3.5%, AQuA +0.7%, SingleEq +0.4%, SVAMP +0.6%, CSQA +1%, StrategyQA +0%, Letter +0.7%, Coin +2.7%
Complexity [40]	math word	**hand-written**	self-consistency	greedy/1prompt	GSM8K +7%, MultiArith +3%, Penguins +3%
Self-ask [101]	math word	**ext knowledge**	LLM	multi-hop q.	Bamboogle =60%, 2Wiki =40%, Musique =15%
Self-verification [148]	math word	hand-written	**back-verify**	ensemble	GSM8K +4%, SingleEq +2%, AddSub +4%, MultiArith +3%, AQuA +3%, SVAMP +1%
Self-consistency [143]	math word	hand-written	**majority**	ensemble	GSM8K +18%, SVAMP +11%, AQuA +12%, StrategyQA +6%, ARC-c +4%
Codex [16]	code	-	**tool-based**	-	HumanEval =70%
Self-debugging [19]	code	hand-written	**tool-based**	greedy	Spider +9%, MBPP +12%, TransCoder +12%
Fun-search [108]	code	hand-written	**tool-based**	evolutionary alg	cap set 8 =512
LLaMEa [134]	code	hand-written	**tool-based**	evolutionary alg	BBOB +11%
MathPrompter [60]	math	hand-written	**tool-based**	ensemble	MultiArith =92%
Program-of-thoughts [17]	math word	hand-wr, Codex	**Python+Consist.**	split reason/cmput	GSM8K =71%, SVAMP =85%, ASDIV =85%, AddSub =92%, MultiArith = 99%
Program-aided-lang [41]	math word	hand-wr, Codex	**NLP/Python**	ensemble	GSM8K =72%, SVAMP =79%, ASDIV =79%, SingleEQ =96%, SingleOP =94%, AddSub = 92%, MultiArith = 99%, Penguins = 93%
Refiner [96]	math word	finetune	**critic model**	gen/crit feedback	SVAMP =72%, GSM8K =78%
Self-correction [147]	math word	finetune	**corrector model**	gen/corr feedback	MathProgSynth =24%, LexConstrGen =98%, ToxicityControl =0.0%
Self-improvement [54]	math word	finetune	**self-assessment**	CoT/consistency	GSM8K =82%, DROP =83%, ARC-c =90%, OpenBookQA =94%, ANLI-A3 =68%
Say-can [3]	robot	model-generated	**external model**	greedy	Kitchen =31%
Inner-monologue [57]	robot	hand-written	**various**	greedy	TableTop =90%, Kitchen =60%
Self-taught-reasoner [163]	math word	finetune	**augmentation**	greedy/feedback	CommonsenseQA =72%
Least-to-most [169]	math word	hand-written	self-assessment	**curriculum**	SCAN =99%
Progressive-hint [167]	math word	model-generated	self-assessment	**stable prompt**	AddSub +2%, MultiArith +0%, SingleEQ +2%, SVAMP +3%, GSM8K +5%, AQuA +1%
Self-refine [82]	math word	model-generated	self-assessment	**greedy/feedback**	Sentiment +32%, Dialog +49%, CodeOptim +8%, CodeRead +28%, MathReason +0%, AcronymGen +25%, ConstrainedGen +30%
Tree-of-thoughts [159]	puzzles	model-generated	self-assessment	**BFS/DFS**	Game24 =74%, CreativeWriting , Crossword
Buffer-of-thoughts [156]	math word	thought template	self-assessment	**buffer manager**	Game24 +11%, GeoShapes +20%, Checkmate +51%
Beam-search [153]	math word	model-generated	self-assessment	**Beam Search**	GSM8K +6%, AQuA +9%, StrategyQA +5%
ReAct [158]	action	external knowledge	self-assessment	**reinf learning**	ALFWorld =34%, WebShop =10%
Reflexion [121]	decision	model-generated	ext model	**reinf learning**	HumanEval =91%
Voyager [141]	Minecraft	model-generated	Minecraft	**reinf learning**	15 x faster

(3.3) Control. The third stage is control. A reasoning process that consists of multiple steps is a sequential decision process [77]. When a single chain of reasoning steps is generated, the control flow of the reasoning process is simple: greedily evaluate the first step and then the next one, if present. The control flow of the reasoning process may also be more intricate. Some reasoning problems can be divided into multiple subproblems. To execute, evaluate and combine the results of all substeps, a separate controller may be needed. This controller can be a prompt or an external algorithm.

Again, we distinguish three approaches. Most papers use a *(3.3.1) greedy selection* approach: a single prompt with a single chain of steps is generated, and these steps are directly executed and followed. The second approach is to generate an *(3.3.2) ensemble strategy* of reasoning steps, evaluate them, combine the individual results, and present them as the result of the ensemble. Finally, a full tree-search or a *(3.3.3) reinforcement learning* (RL) algorithm can be used as scaffolding. In this case, when a step is followed and evaluated, the LLM can roll back and try a different reasoning step, as in a search [98]. Going further, a full reinforcement learning approach can be used [99, 125] to find an optimal policy for the sequential decision process. A Markov Decision Process of state, action, transition, and reward function is specified, and step control can become a process where prompts are generated dynamically.

Domain. Initially, papers are applied to math word problems (natural language descriptions of math problems). Math problems were the original inspiration for the experiments with reasoning in LLMs. Other application domains include autonomous agents, robotic movement, generating computer programs, and playing computer games. We will discuss these in more detail in the descriptions of the individual approaches in this section.

Taxonomy Table. Table 2 lists main papers of this survey. We show the domain they work on, the type of prompt generation, the evaluation of the result, and the control method. These three categories of approaches—indicated by their Sections (3.1) generation, (3.2) evaluation, (3.3) control—are shown in the table as groups divided by horizontal lines. The first group in the Table, from Scratchpad to Self-ask, focuses on creating a prompt that *generates* the reasoning steps. The entries in the cells of this column are shown in bold, highlighting the focus of the approaches. The approaches in this group are the start of the field of LLM-reasoning. The Chain-of-thought approach is especially an inspiration for many works. The prompts are often written manually by the researchers for each problem; the steps are encoded in one prompt, and step control is greedy. There is no specific evaluation of the steps, other than comparing results to the benchmark. The Scratchpad approach is special in that it uses supervised learning, not prompt-learning; the work showed that LLMs can generate internal reasoning steps by supervised learning, paving the way for prompt-based works.

The second group, from Self-verification to Self-taught-reasoner, focuses on *evaluation* of the reasoning steps in the prompt. This column is shown in bold in the table. The approaches in this group aim to improve the Chain-of-thought results by reducing error accumulation that occurs when multiple steps are taken in a reasoning chain. A variety of step control methods is used by these approaches, which is discussed in more detail later. Note that not all approaches use natural language problems. For example, the subgroup of Codex to Program-aided-language focuses on formal languages. They generate code or math equations, typically in Python, to formalize the steps of the reasoning problem, or as result of the task. LLMs are quite good at code generation, and these approaches typically achieve good performance. The use of code also allows the approaches to call external programs such as interpreters and debuggers to evaluate the correctness of the reasoning steps that are generated.

There is also a special subgroup, Refiner to Self-improvement, that uses finetuning in addition to prompt learning. Here, new data is generated based on reasoning exemplars, which is then used to further train the model. The extra data is often generated as a separate dataset, sometimes called *critic* or *corrector*.

There are two approaches, Say-can and Inner-monologue, whose application domain is control of robot movement. Robotic movement is constrained by the laws of physics (both in the body of the robot as in aspects of its environment). The laws of physics are learned and used to ground the reasoning steps in reality (to reduce hallucination).

The third group, Least-to-most to Voyager, addresses step *control* (shown in bold in this column). Whereas in the previous approaches the reasoning steps are written in a single, static, prompt, these approaches generate the steps in multiple, dynamic, prompts. This allows control of the space of reasoning steps. Various search control approaches are used, all in the form of an external algorithm that performs calls to the LLM with different prompts. The control methods range from simple greedy and depth-first search to elaborate beam search and reinforcement learning schemes.

The last column of the Table summarizes reported benchmark results. The '=' symbol indicates absolute scores on the benchmarks, while '+' indicates relative improvement in percentage points over a baseline LLM, typically GPT-3.5. Results vary strongly, both between approaches and within a single approach between benchmarks. Also, different LLMs were used, from early stage to more mature models, open and commercial, and the baselines differ. For some

```
Input:
2 9 + 5 7

Target:
<scratch>
2 9 + 5 7 ,  C: 0
2 + 5 , 6 C: 1  # added 9 + 7 = 6 carry 1
, 8 6 C: 0  # added 2 + 5 + 1 = 8 carry 0
0 8 6
</scratch>
8 6
```

Fig. 2. Example of input and target for supervised learning on a *long addition* problem of adding two numbers. The carry is recorded in the C: digit. Comments (after #) are not part of the learning target [91]

benchmarks, such as the Creative Writing benchmark in Tree-of-thoughts, results are best reported qualitatively. The source papers provide more measurement details.

In conclusion, we see a diverse array of methods that often achieve high performance in reasoning on their respective domains. To better understand the approaches, we discuss them in more detail, starting with the generation of steps.

3.1 Generation of Steps

Originally, LLMs performed poorly on math word problems such as GSM8K [27]. Different approaches were tried unsuccessfully, for example scaling up the size of the LLM [104]. The LLM architecture, based on transformers, is designed to produce a single token. When we prompt such an architecture to produce an answer, it does so. What we should do instead, is to prompt it to follow intermediate steps, answer those, and thus work towards the final answer, just as a student is taught to break down a complex problem into smaller steps. We should guide the model to explicitly produce intermediate steps, and combine the intermediate results. This idea was used by Nye et al. [91] in Scratchpads, a transformer model that performs multi-step computations by asking it to emit intermediate computation steps into a *scratchpad*. They train the model by supervised learning (not prompt-based in-context learning). Figure 2 shows an example. On experiments with addition, polynomial evaluation, and Python code execution, versions that produced the intermediate steps on a scratchpad performed considerably better than versions that did not, going from 35% to 95%, from 32% to 51%, and from 30% to 42% accuracy, respectively.

If supervised learning can produce intermediate steps, would prompt learning be able to do so too?

3.1.1 Hand-written Prompt. This question was studied by Wei et al. [146], amongst others. A basic way to instruct an LLM to generate steps by prompt-learning is to manually write a prompt for the large language model to follow the reasoning steps. When the LLM is prompted to rephrase information from the question as intermediate reasoning steps in its answer, the LLM performed much better than when it was prompted to answer a math problem directly, without reproducing information from the question in its answer in multiple steps. The example from the Chain-of-thought paper is shown in Figure 3. Table 1 shows that the largest accuracy increase is on GSM8K, from 16% to 47%.

The performance improvement by Chain-of-thought has caused much excitement and has opened up further research on reasoning with LLMs. In the original paper the prompts were handwritten by the researchers for the individual

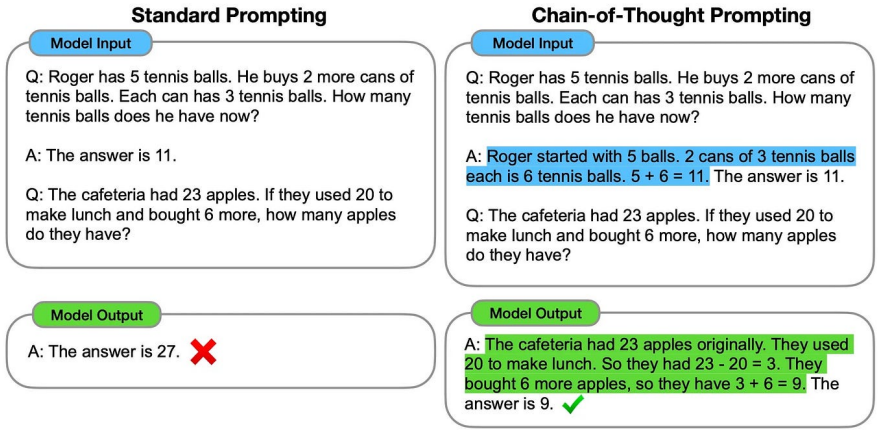

Fig. 3. Chain-of-though Prompting. In blue at the top the prompt, in green at the bottom the answer. When shown the longer example prompt, the LLM follows the longer example when answering the question [146]

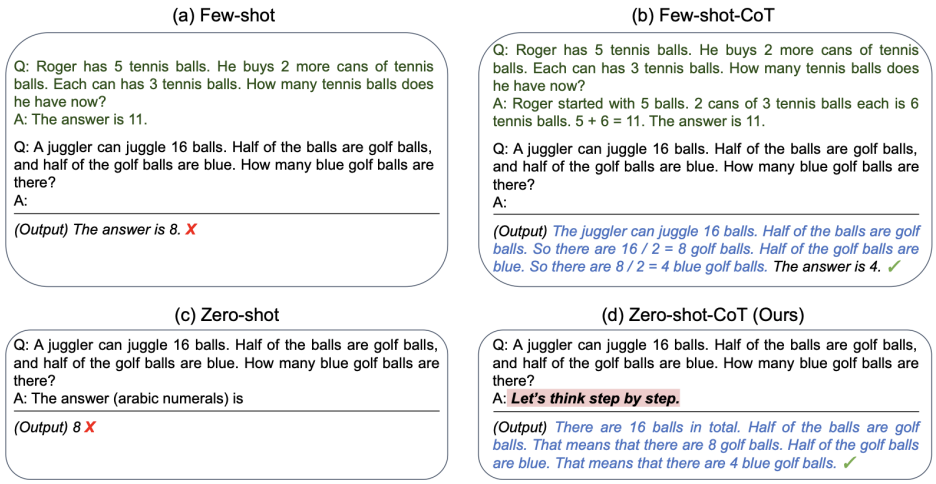

Fig. 4. Zero-shot Chain-of-thought: *Let's think step by step* [68]

types of problems, and evaluations are conducted with benchmarks (not by an LLM).[2] In a later work the prompts were generated automatically by the LLM [165], and evaluated.

Kojima et al. [68] go a step further. They show that the addition of a single text to the prompt (*Let's think step by step*) significantly improves performance. Since this text does not contain problem-related elements, it is as a form of zero-shot learning. Figure 4 compares the approaches. Experiments further show that with this addition to the prompt

[2]The Chain-of-thought idea is about prompt generation, not about the evaluation or the search control of the reasoning steps. Hence, in Table 2 Chain-of-thought is labeled as *greedy* without an evaluation.

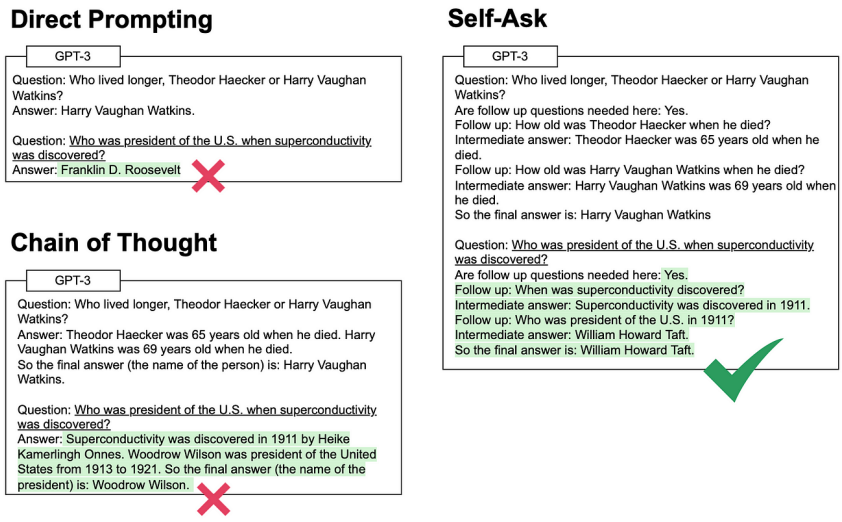

Fig. 5. Self-Ask asks follow-up questions, and uses an external search engine [101]

significant performance gains are achieved on a diverse set of reasoning benchmarks, including arithmetic, symbolic, and logical reasoning (achieving 70% accuracy on GSM8K/PaLM when Self-consistency is also included).

The Chain-of-thought idea itself is inspired by earlier work where natural language steps are generated for arithmetic reasoning [27, 76], and the use of formal languages for reasoning [4, 18, 20, 109].

3.1.2 Prompt using External Knowledge. Chain-of-thought prompts are written manually, by the researchers, an approach that does not scale. We can also use external information about the problem to improve the prompt. Press et al. [101] study how subproblems are related to the main problem, which they call *compositional reasoning*. They study how often a model is able to answer the subproblems, but not the overall problem. This difference is called the compositionality gap. They find that in GPT-3, as model size increases, the single-hop question-answering performance improves faster than the multi-hop performance: while more powerful models memorize and recall more factual knowledge, no improvement in their compositional reasoning occurs. The ability to reason does not depend on the size of the model.

Subsequently, a method called *Self-ask* is proposed, that asks elicitive follow-up questions (like Chain-of-thought, but with the *follow up:* prompt), that the model then answers (see Figure 5). Self-ask can also use an external search engine to answer intermediate prompts, instead of the model. The initial subquestion is fed into the search engine, and the answer is processed by the model, which generates another subquestion, and so on, until it produces the final answer. Self-ask was tested on three benchmarks that were specifically designed for multi-hop questions. Although it performs only a few percentage points better than vanilla Chain-of-thought, it showed how external knowledge can be used in a reasoning setting.

3.1.3 Model-Generated Prompt. In addition to manually writing prompts or using external information, we can also let the LLM itself study the problem to write the best reasoning-prompt. An example of such self-improvement is

Auto-chain-of-thought [165]. This approach builds on the observation by Kojima et al. [68] that large language models are zero-shot reasoners. First, Auto-chain generates specific questions for a given dataset and partitions them into clusters. Then an external algorithm uses the model to generate examples that are sampled for diversity. The constructed demonstrations augment the in-context prompt. This approach also performed a few percentage points better than hand-written Chain-of-thought prompts, on ten benchmarks, using GPT-3 (see Table 2).

Fu et al. [40] introduce Complexity-based prompting. Inspired by Chain-of-thought and Self-consistency, their work specifically studies the impact of the complexity of the reasoning chain (the number of steps), and introduces a related reasoning approach (Complexity-based prompting). They find that prompts with the largest complexity perform best, and also that answers with the highest complexity are the best. Complexity-based prompting achieves somewhat higher performance on three math reasoning benchmarks: GSM8K improves 7 points, MathQA 6 points, and the Penguins benchmark from Big Bench Hard improve 3 percentage points.

We see that the initial approaches showed larger improvements than the later approaches. It is time to look at another category of approaches, that focus on the evaluation of reasoning steps.

3.2 Evaluation of Steps

After discussing prompts for the generation of reasoning steps, the next stage in the generation/evaluation/control pipeline is *evaluation* of the results of the steps. This stage focuses on reducing error accumulation of multi-step reasoning chains. We will start with approaches where the same model performs step-generation and step-evaluation.

3.2.1 Self-Assessment. When LLMs are prompted to perform reasoning steps, they perform a sequence of steps and predict multiple tokens. Performing a sequence of steps makes them sensitive to mistakes and vulnerable to error accumulation (logical, factual, ethical, or otherwise) [148, 151]. Several methods have been developed to prevent error accumulation. One approach is to create a new model to separately evaluate the results. Shen et al. [118] and Li et al. [75] train an external verifier to check results. In contrast, Weng et al. [148] propose an automated approach using evaluation by the same LLM, called Self-verification. They note that human reasoning also suffers from the problem of accumulating errors, and that in human reasoning we frequently revisit our thought process to verify the accuracy of our reasoning steps. The LLM is prompted to use the conclusion of the Chain-of-thought reasoning chain as a condition for solving the original problem and then compare the answer, going back to the original question. The LLM is given variations of its own conclusion and is instructed to choose the one with the highest similarity to the original question. (Note that there can be feedback issues using an LLM to evaluate itself, for a discussion see Zheng et al. [168].) Experiments are reported on GPT-3 [16] and on Instruct-GPT [92]. The accuracy of Chain-of-thought was improved by a few percentage points on arithmetic and general reasoning tasks (GSM8K 65%, AQuA 48%, SVAMP 77%).

A popular related approach is Self-consistency [143]. Self-consistency is a straightforward ensemble approach (a well-known machine learning technique to make a strong learner out of multiple weaker learners [12, 111]). Greedy single-path decoding is replaced by sampling diverse reasoning paths, evaluating them, and selecting the most consistent answer. Self-consistency asks the LLM to simply perform the same query multiple times, and takes the majority-vote of the answers, or decoding paths. Self-consistency works since complex reasoning problems typically allow different reasoning paths that lead to the correct answer. Figure 6 summarizes the approach. Self-consistency has been evaluated on arithmetic reasoning, commonsense reasoning and symbolic reasoning, on a variety of LLMs, including GPT-3 [14, 22, 128, 129]. Self-consistency further improves the performance of Chain-of-thought by 10-20 percentage points, and has been used as a baseline in many of the other approaches in this survey.

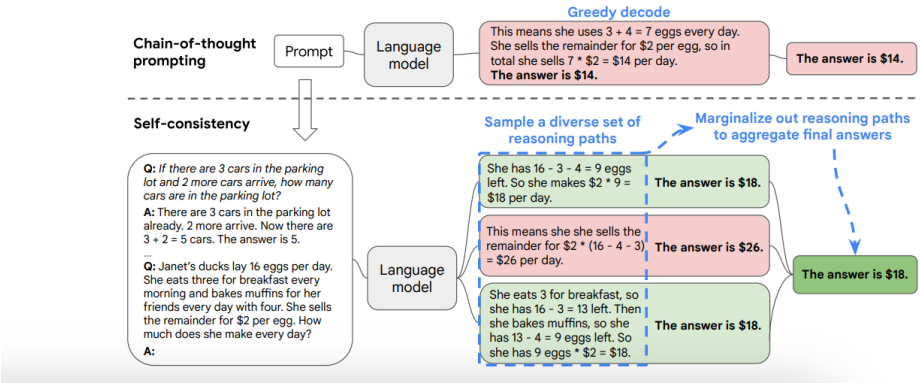

Fig. 6. Self-Consistency [143]

3.2.2 Tool-based Validation. Another approach to improve the accuracy of evaluating the reasoning steps is to switch from a natural to a formal language. The advantage of a formal language is that it is less ambiguous than a natural language. Examples are computer languages, such as Python, or mathematical equations. Using a formal language for reasoning is a popular approach, and we discuss seven papers. Many approaches generate the steps in Python, and the code can then be evaluated by a formal evaluator, such as a compiler, debugger, or interpreter.

LLMs have been quite successful in generating computer code from natural language prompts. Chen et al. [16] introduced Codex, a GPT model that was trained on publicly available code in the repository GitHub. A production version of this work was introduced under the name GitHub Copilot. Codex is able to generate correct programs from descriptions in natural language, such as commentary strings. Figure 7 shows examples that are produced by Codex.

The work on Codex is used as a basis for further research on reasoning in LLMs. Human programmers, when writing code, typically follow a cycle of writing some code, executing it to look for errors, and then using the feedback to improve the code. This step-by-step approach is followed in Self-debugging [19]. It follows the same steps of code generation, code execution, and code explanation (see Figure 8). Self-debugging is able to identify mistakes in its own code by investigating the execution results, and can also provide an explanation of the generated code, in natural language. It achieves strong performance: the text-to-SQL Spider benchmark improves 9 points, and the C++ to Python Transcoder benchmark improves by 12 percentage points.

Several works generate working code tuned for solving specific problems automatically, without human feedback. Romera-Paredes et al. [108] introduced FunSearch, an approach that integrates formal methods and LLMs to enhance mathematical reasoning and code generation. FunSearch uses a genetic approach with multiple populations of candidate solutions (programs), which are evaluated using a function depending on the problem specification. In addition to the evaluation function, also an initial program is given to the LLM in the first prompt. After evaluating a number of generated programs from the starting prompt, a new prompt is created, in an iterative fashion, combining a selection of sampled programs sorted according to their evaluation score, and the LLM is requested to generate a new program. Another work leverages evolutionary computation methods to generate and optimize evolutionary algorithms [134]. This approach, LLaMEA (Large Language Model Evolutionary Algorithm), utilizes LLMs to design and optimize evolutionary algorithms. The approach uses LLMs to generate initial algorithmic structures, which are then refined through mutation

```python
def incr_list(l: list):
    """Return list with elements incremented by 1.
    >>> incr_list([1, 2, 3])
    [2, 3, 4]
    >>> incr_list([5, 3, 5, 2, 3, 3, 9, 0, 123])
    [6, 4, 6, 3, 4, 4, 10, 1, 124]
    """
    return [i + 1 for i in l]
```

```python
def solution(lst):
    """Given a non-empty list of integers, return the sum of all of the odd elements
    that are in even positions.

    Examples
    solution([5, 8, 7, 1]) ==>12
    solution([3, 3, 3, 3, 3]) ==>9
    solution([30, 13, 24, 321]) ==>0
    """
    return sum(lst[i] for i in range(0,len(lst)) if i % 2 == 0 and lst[i] % 2 == 1)
```

```python
def encode_cyclic(s: str):
    """
    returns encoded string by cycling groups of three characters.
    """
    # split string to groups. Each of length 3.
    groups = [s[(3 * i):min((3 * i + 3), len(s))] for i in range((len(s) + 2) // 3)]
    # cycle elements in each group. Unless group has fewer elements than 3.
    groups = [(group[1:] + group[0]) if len(group) == 3 else group for group in groups]
    return "".join(groups)

def decode_cyclic(s: str):
    """
    takes as input string encoded with encode_cyclic function. Returns decoded string.
    """
    # split string to groups. Each of length 3.
    groups = [s[(3 * i):min((3 * i + 3), len(s))] for i in range((len(s) + 2) // 3)]
    # cycle elements in each group.
    groups = [(group[-1] + group[:-1]) if len(group) == 3 else group for group in groups]
    return "".join(groups)
```

Fig. 7. Codex example [16]

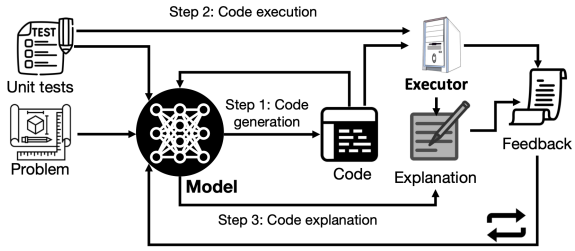

Fig. 8. Self-Debugging control flow [19]

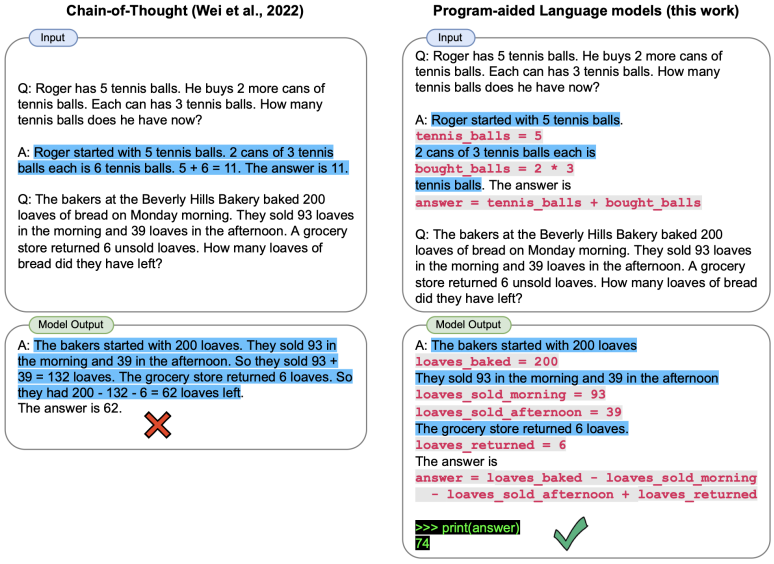

Fig. 9. Program-aided-language [41]

and selection. This enhances the efficiency of algorithm design, particularly in fields requiring innovative and adaptive solutions, improving accuracy on the *Black-Box Optimization Benchmark [48] (BBOB)* suite by 11 pecentage points. A key difference between FunSearch and LLaMEA is that LLaMEA uses a sample-efficient elitism strategy by iteratively improving the best-so-far solution, requiring significantly fewer prompt evaluations than the large-population strategy proposed in FunSearch. Evolutionary approaches let the LLM discover new algorithms, solving existing problems in new ways, or solving entirely new problems. Another method, Evolution-of-heuristics [78], was proposed for evolving code snippets for guided local search to solve combinatorial optimization problems, such as the Traveling Salesperson Problem.

To improve prompt-based reasoning, Codex is used in an ensemble approach named MathPrompter [60]. This approach generates multiple algebraic expressions or Python functions, which then solve the same math problem. The results are compared, just like in Self-consistency and Self-verification, raising the confidence level in the results. MathPrompter achieved state-of-the-art accuracy on the MultiArith dataset (78.7% $\rightarrow$ 92.5%), evaluated on GPT-3 175B.

Two other approaches that use a formal language are Program-of-thought [17] and Program-aided-language [41]. Both approaches use the LLM to generate Python and then use an interpreter to evaluate the result. The approaches are similar although Program-aided-language uses generic prompts, and has been tested on more benchmarks. Figure 9 illustrates the Program-aided-language approach. When the evaluation of the reasoning steps is offloaded to the Python interpreter, decomposing the natural language problem into executable code-steps remains the only task for the LLM. (Earlier work in math word problems showed how to decompose a problem and reach an answer [76].) Gao et al. [41] provide extensive experimental evidence about the synergy between the neural LLM and the symbolic interpreter. Experiments are performed on 13 mathematical, symbolic, and algorithmic reasoning tasks, achieving more accurate results than much larger models (for example, Program-aided-language reported 72% on GSM8K and 93% on Penguins).

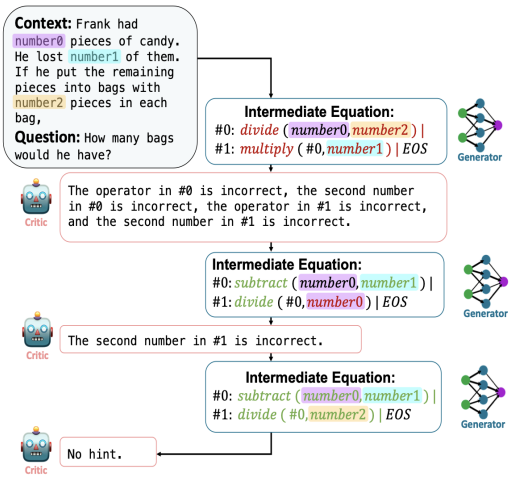

Fig. 10. Refiner [96]

3.2.3 External Model Validation. We have seen many examples of succesful prompt-based in-context reasoning and evaluation (at inference time—where no parameters were changed). We will now look at reasoning approaches that follow a more traditional parameter training approach. All approaches evaluate the output of the model and generate corrective data. That data is then added to the training pipeline, and the model is subsequently finetuned.

Finetuning. The Refiner approach [96] uses a generator model and a critic model that provide fine-grained feedback on reasoning errors. The generator generates multiple reasoning hypotheses, and the critic evaluates results by randomly selecting a hypothesis for feedback. The generator model is then finetuned based on its reasoning errors. A small supervised model is used to overcome the cold-start problem. Figure 10 shows an example of how the critic provides feedback to the generator. The approach achieves 78% accuracy on GSM8K and also works well on related problems.

Welleck et al. [147] follow a similar approach which they call Self-correction. Here, the corrector is a separate model specialized in refining the outputs of the generator. Unlike Refiner, where the generator is finetuned based on the critic, Self-correction finetunes the corrector to rectify errors in the hypotheses produced by the generator. Self-corrector is not applied to math word problems, but to program synthesis, where a small corrector reduces toxicity to 0%.

A third finetuning approach is Self-improvement [54]. Here, too, the base model data is augmented by LLM-generated rationales, and then finetuned. They achieve 82% accuracy on GSM8K and similarly high scores on question answering and adversarial benchmarks. Noteworthy in all three finetuning approaches is that LLMs are capable of improving themselves by training on their own generated output, and that stability problems from feedback loops are overcome.

Dataset Augmentation. The final finetuning approach that we discuss uses dataset augmentation. In Self-taught-reasoner [163], an intermediate reasoning is generated, called a *rationale*. Rationales are shown to be valuable across diverse tasks such as mathematical and commonsense reasoning, code evaluation, social bias inference, and natural language inference. First an augmentation dataset is created by attempting to solve the original dataset. Next, the dataset is augmented using rationalizations and ground-truth answers to problems the model failed to solve. Finally, the model

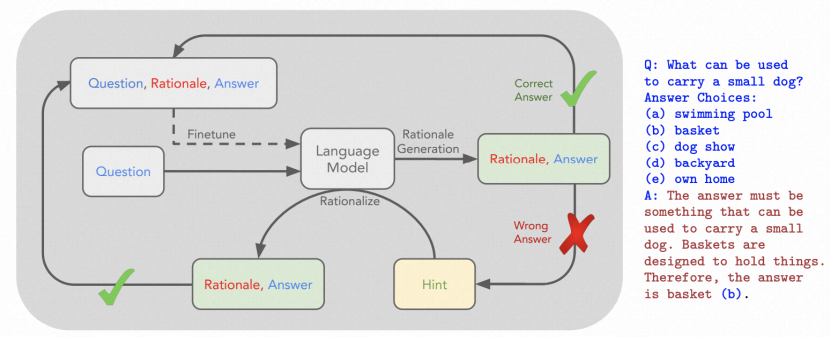

Fig. 11. Self-Taught-Reasoner [163]

Fig. 12. Say-Can compared to other language models [3]

is finetuned on the combined dataset. Figure 11 illustrates the approach. Self-taught-reasoner performs comparably (72%) to finetuning a 30 times larger model on CommonsenseQA.

Reasoning about Robot Behavior. In addition to math word problems, computer code, and common sense, prompt-based reasoning has also been used to improve robot behavior. Language models contain a large amount of information about the real world [3]. In theory, this should allow them to reason realistically about robotic behavior. However, the models do not have knowledge about specific embodied aspects of a particular robot. If we could compare a Scratchpad-like list of intermediate reasoning steps with a list of possible movements of the robot in its environment, then we could prevent the model from suggesting impossible joint movements and actions, and prevent failures.

Say-can [3] learns a value function [64] of the behavior of a robot and its environment using temporal difference reinforcement learning [124]. This value function is combined with prompt-based reasoning by the language model, to constrain it from suggesting impossible actions. The goal of Say-can is to ground language in robotic affordances. In contrast to Scratchpad, which used supervised learning, the affordance model is learned interactively by reinforcement learning, and then applied using prompt-based learning on the LLM. The language model has high-level semantic knowledge about the task (Say). The learned affordance function (Can) provides an environment-grounding on what is possible. Say-can achieves a 31% success rate on 101 real-world robotic kitchen tasks (see Figure 12).

Where Say-can learns affordances as a separate function, Inner-monologue [57] formulates robotic planning directly as part of the language prompt, internally. The input consists of many elements: textual descriptions from InstructGPT

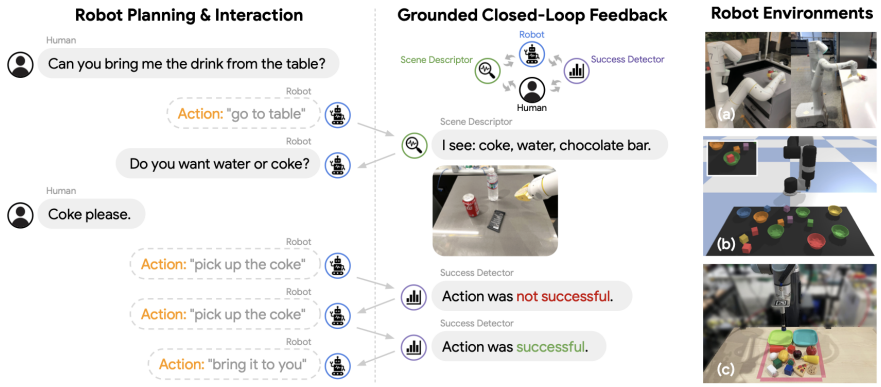

Fig. 13. Architecture of Inner-Monologue [57]

[14] for multi-step planning, scripted modules for object recognition, success detection, task-progress scene description, and language-conditioned pick-and-place primitives, similar to CLIPort [123]. Figure 13 gives an example of the working of Inner-monologue. The language feedback that is thus generated significantly improves performance on three benchmarks, achieving 90% accuracy on simulated and real table top rearrangement tasks and 60% on the kitchen environment. There are many other studies into robotic behavior. An approach related to Inner-monologue is Chain-of-tools, which proposes a plan-execute-observe pipeline to ground reasoning about tool behavior [119, 120].

This concludes our discussion of the second stage of the reasoning pipeline, *evaluation* of the reasoning steps.

3.3 Control of Steps

The third stage is *control*. This stage controls how many sub-steps are generated, and how deep into the future the reasoning chain is generated. There are three main approaches: (3.3.1) *greedy selection*, which generates a step and then follows it, (3.3.2) *ensemble strategy*, which generates a set of possible next steps, and (3.3.3) a (*reinforcement learning*) search which generates multiple options for the steps, traversing a search tree with backtracking, controlling an exponential search space [154].

3.3.1 Greedy Selection. Most earlier works on prompt-based reasoning follow the greedy approach: generate a single prompt with a single sequence of steps and follow them. Among the greedy reasoners are Chain-of-thought, Auto-CoT, and Zero-shot CoT. Inner Monologue and Say-Can also use greedy reasoning.

In Least-to-most prompting [169], the key idea is to break down a complex problem into simpler subproblems and then solve these in sequence, explicitly encoding them in the prompt, related to Complexity-based prompting. In Least-to-most the answers to previously solved subproblems help in finding the answer, as a curriculum [8]. On symbolic manipulation, compositional generalization, and math reasoning, Least-to-most prompting generalizes well, achieving 99% accuracy on a compositional generalization benchmark. Figure 14 illustrates the idea.

3.3.2 Ensemble Strategy. The second kind of reasoning control is based on an ensemble of (sequences of) reasoning steps. For most problems, multiple different options exist for the next step. When all or some of these are generated and evaluated, then the consensus result can be reported as the outcome of an ensemble of steps. Self-consistency [143] and

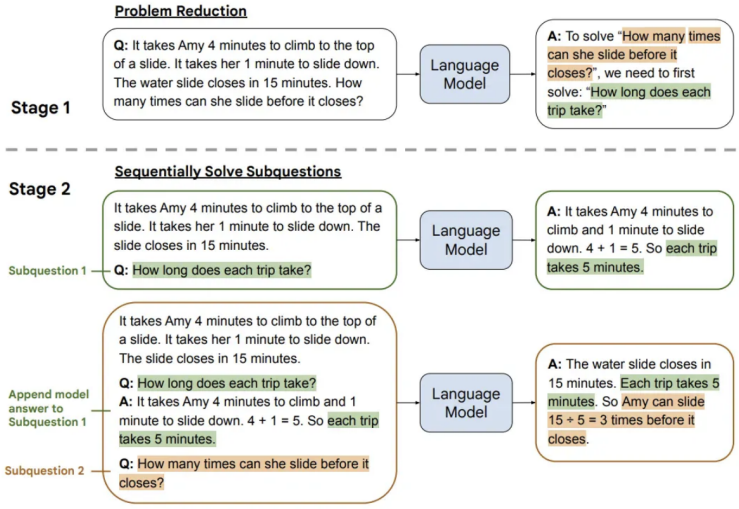

Fig. 14. Least-to-most prompting [169]

Self-verification [148] (in Section 3.2.1) are popular ensemble approaches to evaluate the results of reasoning steps, in which greedy single-path decoding used in Chain-of-thought prompting is replaced by a diverse set of paths. Taking this further, Chain-of-experts uses a mixture-of-experts ensemble for complex combinatorial problems [152]. PAL and MathPrompter also use the ensemble approach. The ensemble approach is popular in reasoning with LLM.

3.3.3 Reinforcement Learning. In reasoning, often multiple valid steps are possible, but pursuing all possibilities over multiple trajectories may lead to an infeasible number of possibilities. The third kind of reasoning control is to use a full-fledged controller that can traverse a tree, or perform reinforcement learning to do so [64, 99, 125]. When decomposing the problem, multiple alternative steps are generated that can be searched multiple steps into the future. Then, backtracking can be performed, allowing alternative steps to be tried.

Where greedy and ensemble processes can be controlled with a prompt by the LLM, this third category is more complex, and an external algorithm is used to control the reasoning process. The external algorithms call the LLM as a subroutine prompting it to perform requested tasks. The external algorithm allows more complex reasoning control, but we are now beyond prompt-based self-reasoning: control has been given to an algorithm that is external to the LLM and external to prompt-learning.

We start our discussion of control strategies with depth-first and breadth-first search, then go to beam search, and then to full reinforcement learning. A complex reasoning space can be traversed with a search algorithm. Tree-of-thoughts [159] uses breadth-first or depth-first search to dynamically follow different reasoning steps. The evaluation part in Tree-of-thoughts is performed with a prompt to the LLM. Together, the trio of generation, evaluation, and control allow systematic exploration of the space of reasoning steps with look-ahead and backtracking. The authors compare their approach to Chain-of-thought and Self-consistency on the Game of 24, Creative writing, and Mini crossword, achieving

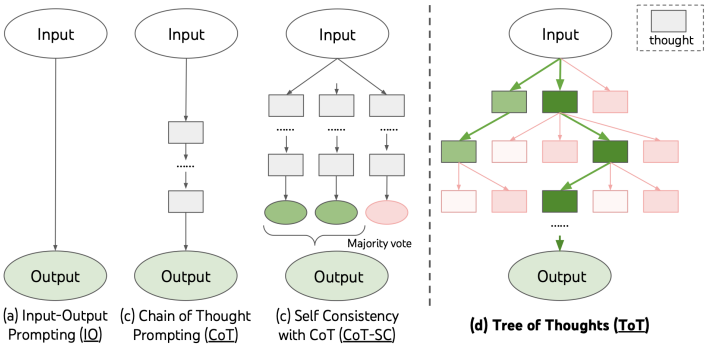

Fig. 15. Reasoning structure of Chain-of-Thought, Self-Consistency, and Tree-of-Thoughts [159]

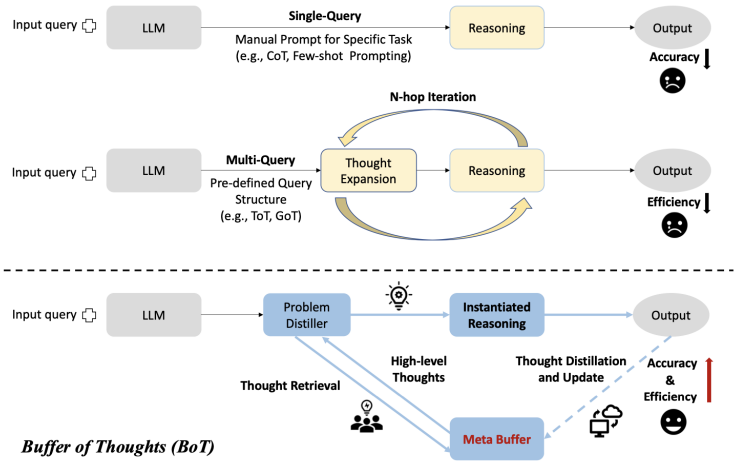

Fig. 16. Chain-of-Thought, Self-Consistency, and Buffer of Thoughts [156]

an accuracy of 74% on a Game of 24 task. The other tasks are evaluated qualitatively. Figure 15 illustrates the different reasoning structures.[3]

Another approach, Buffer-of-thoughts [156], goes a step towards metareasoning [42]. It introduces a meta-buffer that stores high-level *thought-templates*. These thought-templates are derived from a variety of tasks. Figure 16 compares the Buffer-of-thoughts approach to other approaches such as Chain-of-thought and Tree-of-thoughts. Buffer-of-thoughts outperforms other methods in puzzles such as Game of 24 (by 11%) and checkmating (by 51%). Thought templates are related to metacognition (thinking about thinking), which is further discussed in Section 4.2.3.

A related search method is Beam-search-for-reasoning [153]. When the space of possible reasoning paths is large, Beam-search searches a promising part of this space. It uses self-evaluation to control exploration and to evaluate

[3]A similarly named approach is Graph-of-thoughts [11]. Graph-of-thoughts allows more general reasoning graphs, providing a formal framework, where the different elements can then be specified manually.

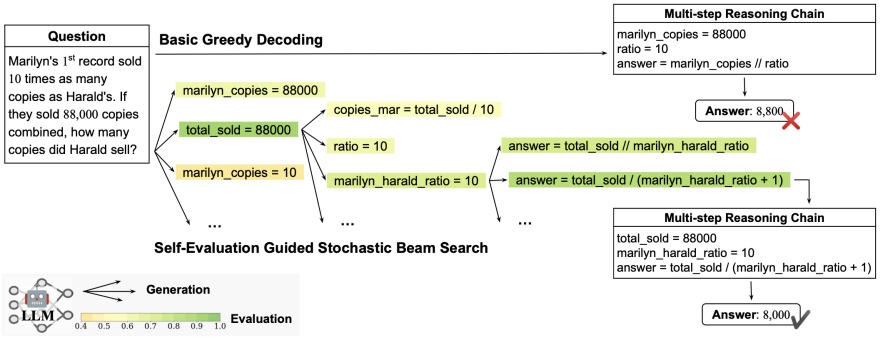

Fig. 17. Self-evaluation in multi-step reasoning in Beam-Search [153]

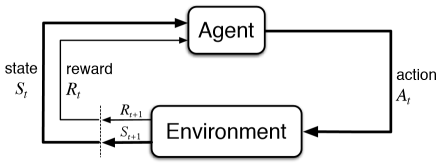

Fig. 18. Reinforcement Learning [125]

(decode) reasoning steps. Figure 17 illustrates how Beam-search uses self-evaluation in multi-step reasoning. Beam search uses Program-aided-language models for math word problems [41]. Using a Codex backbone [16], it surpasses the few-shot baselines by 6.34%, 9.56%, and 5.46% on the GSM8K, AQuA, and StrategyQA benchmarks, respectively.

Reinforcement learning is another step in the sophistication of optimization algorithms. It learns by interactive sampling, improving its policy based on rewards from the environment [125]. To use reinforcement learning, the reasoning problem is formulated as a Markov Decision Process: the agent-algorithm creates a prompt (an *action*), to sample a step (t) and get an answer (*state*, *reward*) from the environment-model (see Figure 18). The answer can then be used to improve the prompt (next action), using the rewards to improve its policy of best actions for each state. The approaches that use reinforcement learning also do so in the form of an external algorithm.

Progressive-hint-prompting (PHP) uses reinforcement learning to interactively improve prompts [167]. Figure 19 illustrates the approach. PHP calls the LLM with dynamic prompts, using previously generated answers as hints, to progressively prompt the LLM towards the correct answers. It works as follows: (1) given a question (prompt), the LLM provides a base answer, and (2) by combining the question and answer, the LLM is queried and obtains a subsequent answer. We (3) repeat operation (2) until the answer becomes stable, as a regular policy-optimizing reinforcement learning algorithm. The authors have combined PHP with Chain-of-thought and with Self-consistency. Using GPT-4, state-of-the-art performance was achieved on grade school math questions (95%), simple math word problems (91%) and algebraic question answering (79%).

Another approach that is motivated by improving answers from feedback, is Self-refine [82]. Like PHP, the LLM generates an initial output and provides feedback for its answer, using the LLM to refine itself, iteratively. Figures 20 and 21 illustrate the approach. Self-refine prompts the LLM in three ways: (1) for initial generation, (2) for feedback,

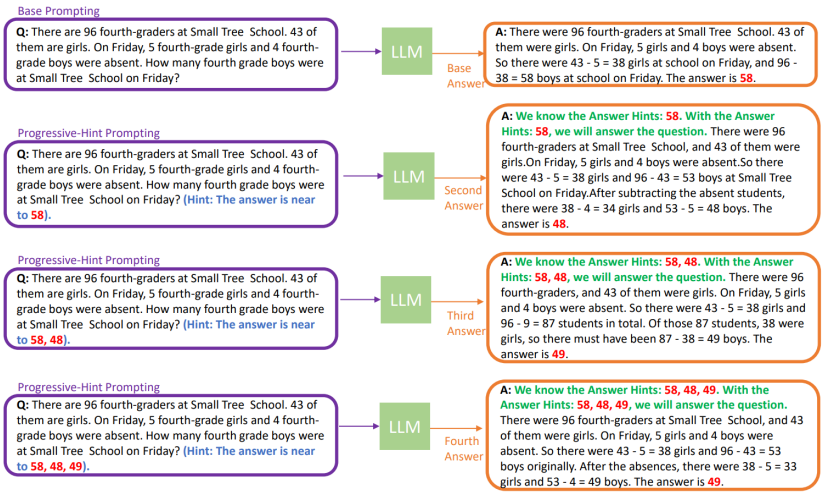

Fig. 19. Progressive Hint Prompting Example [167]

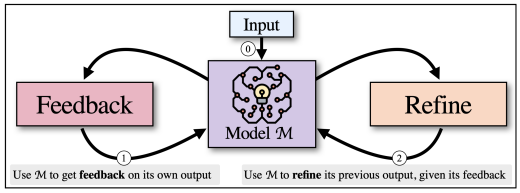

Fig. 20. Self-Refine Architecture [82]

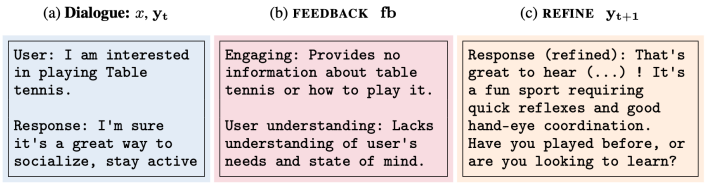

Fig. 21. Self-Refine Example [82]

and (3) for refinement, following a greedy reasoning chain. Self-refine has been used with GPT-3.5 and GPT-4 as base LLMs, and has been benchmarked beyond math word problems on dialogue response generation [5], code optimization, code readability improvement, math reasoning, sentiment reversal, acronym generation, and constrained generation, showing substantial improvements over the base models (typically around 30 percentage points, see Table 2) .

Another approach that combines reinforcement learning and LLMs is ReAct [158]. Most works focus on reasoning by the LLM, not on actions by an agent. The goal of ReAct is to combine progress in reasoning with action plan generation. (Or, to put it differently, other approaches use RL to improve LLM-reasoning, ReAct uses LLMs to improve RL agent

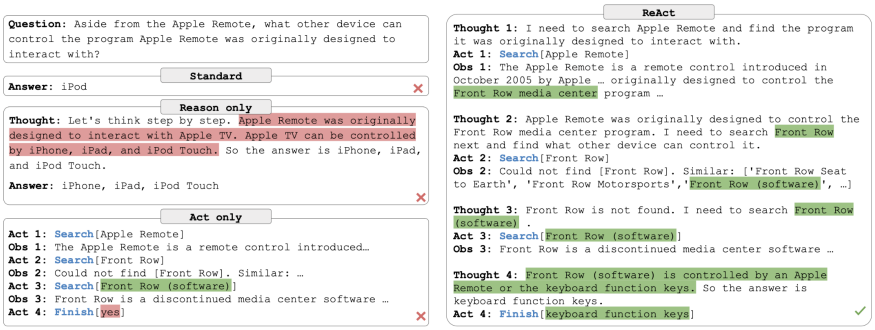

Fig. 22. Comparison of four prompting strategies: Standard, Chain of Thought (reason), Action only, and ReAct [158]

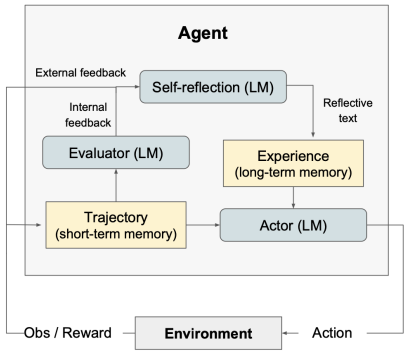

Fig. 23. Architecture of Reflexion [121]

policies.) ReAct uses Chain-of-thought prompt-learning as part of an RL framework that also uses external knowledge sources (Wikipedia) and finetuning; for error reduction, grounding, and for reducing hallucination. The framework allows hand-written prompts. Figure 22 shows four different prompting strategies. On two interactive decision making benchmarks (Alfworld and WebShop), ReAct outperforms imitation and reinforcement learning methods by an absolute success rate of 34% and 10% respectively.

The ReAct work has been developed further. Reflexion [121] creates AI agents that learn by reflecting on failures and enhance their results, much like humans do. Reflexion uses three language models: actor, evaluator, and reflector. It works as follows: (1) an actor generates text and actions, (2) an evaluator model scores the outputs produced by the actor, and (3) a self-reflection model generates verbal reinforcement cues to assist the actor to self-improve (see Figure 23). For the actor, Chain-of-thought and ReAct can be used. Reflexion is evaluated on decision-making, reasoning, and coding tasks. Improvements of 10-20 percentage points are reported.

To conclude this overview of reinforcement learning approaches, we discuss an application in the games domain. Voyager [141] is an agent for the game of Minecraft that uses an iterative prompting mechanism that generates code for embodied control. The agent includes self-verifcation an a skill library to maximize exploration. The goal of is to

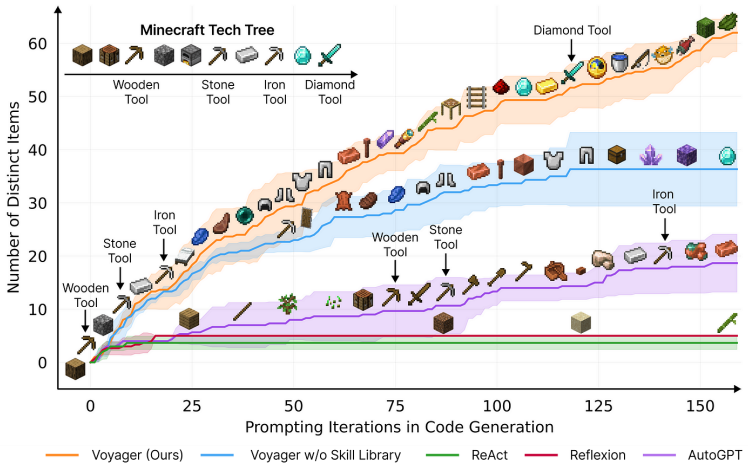

Fig. 24. Performance of Voyager in Minecraft [141]

discover diverse items in Minecraft, a form of novelty search [35]. Voyager performs well, reaching high scores by acquiring many tools (see Figure 24) 15 times faster than the baseline.

The applications of reinforcement learning in LLM reasoning are many, and the connections run deep [102]. Wang et al. [142] use the similarity between RL timesteps and LLM reasoning steps to jointly train a value function together with the LLM policy by optimizing the soft Bellman equation, achieving 85% accuracy on GSM8K and 81% on Alfworld. Du et al. [33], Guo et al. [46] replace supervised finetuning by reinforcement learning, integrating it with reasoning, achieving large efficiency gains in the training pipeline in the DeepSeek r1 and Kimi models.

4 DISCUSSION

We have reviewed many reasoning approaches. It is now time to reflect on the approaches, discuss limitations, and to look for promising areas of future work. First we discuss issues concerning hallucination, faithful reasoning, and scaling. Then we discuss what LLMs can and cannot do. Then, we highlight connections with sequential decision processes and metacognition, and we end with a research agenda.

4.1 Hallucination, Faithfulness and Scaling

Most works on reasoning in LLMs are experimental in nature. The success of in-context learning and Chain-of-thought has prompted efforts to provide deeper insight into the reasoning processes in language models. Saparov and He [113] introduce a synthetic question/answer dataset designed to evaluate the reasoning abilities of LLMs. The work showed that LLMs are capable of reasoning to a certain degree, but that Chain-of-thought struggles with proof trees with a wide branching factor. In another study, Wang et al. [140] also aim to increase our understanding of how Chain-of-thought works. The authors find that it continues to work even with invalid steps in the reasoning chain. They also find that the order of the reasoning steps is important for good results. Prompts should be relevant to the question, and coherent (steps should be in the correct order). Jin et al. [62] study the impact of reasoning step length on LLMs, finding a strong positive correlation between the length of the prompt and reasoning abilities. These works highlight the risk of hallucination in LLM-reasoning, for the model to see steps that are not there. Next, we discuss works on errors in the

Chain-of-thought approach, studying whether the reasoning of the LLM is faithful, or that it gives the right answer for the wrong reason.

4.1.1 Faithfulness. Chain-of-thought approaches prompt a language model to take certain steps to solve the problem that the prompt specifies. One can ask the question, whether those steps are indeed the steps that the model has followed (faithful reasoning) or whether it took another road to arrive at the same answer (unfaithful reasoning). A few studies measure the faithfulness of reasoning by LLMs. Lanham et al. [71] notes that just like organic reasoners, a model's reasoning may be post-hoc, it may be constructed after a certain conclusion has been found. By deliberately adding mistakes to the chain of thought, the authors measure the faithfulness of the model. They find a wide variation of post-hoc reasoning, with a tendency of larger models to be less faithful. Like regular LLMs, when not properly grounded, (Chain-of-thought) reasoning suffers from hallucination [56].

Another study adds deliberate bias to the prompt. For example, in a multiple-choice setting, they always make answer (A) the correct answer [131]. They find that a bias towards wrong answers can cause significant drops in accuracy, and that models frequently generate Chain-of-though explanations rationalizing wrong answers. The authors further note that, insofar as language models are trained on human-written explanations, that explanations may be incomplete or wrong. Human explanations may omit crucial steps of the causal chain, may provide an unfaithful account of the human reasoning process, or may be aimed at convincing others, instead of providing the true causes of a decision.

To address issues of faithfulness, Lyu et al. [81] propose Faithful-chain-of-thought. This approach involves two stages. First, the natural language query is translated into a formal symbolic language. Second, the problem-solving stage processes the formal language, and can explain the reasoning steps it has thus taken. For the symbolic language, Python, Datalog, or PDDL is suggested. Another approach, mechanistic interpretability, studies methods to target individual representations inside the LLM, to see if the expected behavior occurs in practice [9, 106].

Faithfulness studies tell us more about how models reason. Further surveys on this topic are Chuang et al. [26], Luo et al. [80], Mondorf and Plank [88], Paul et al. [97],

4.1.2 Scaling. The emergent abilities of LLMs have prompted research into the nature of scaling and reasoning with LLMs, and, specifically, how reasoning capabilities can be transferred to smaller language models. Scaling laws of LLMs are an active area of study, see for example [50, 51, 66]. Given the computational cost of training LLMs, there is much interest in transferring knowledge to small language models. Comprehensive surveys on knowledge distillation are Gu et al. [45], Xu et al. [155]. For reasoning specifically, Magister et al. [83] have studied reasoning in small language models, using a student model that learns from a teacher model, by finetuning. Another study related to Self-taught-reasoner focuses on explanation in small language models, achieving similar results [74].

Other works focus on prompt distillation for retrieval [29], recommendation [73], embodied agents [21], and LLM graph reasoning [164]. Distillation of reasoning to smaller models can work surprisingly well in situations with more explicit instructions. Distillation is also proposed for bringing results of reasoning from large models to small models [161], which brings us to the topic of metacognition (Section 4.2.3).

4.2 Limitations: What LLMs Can and Cannot do

The capabilities of LLMs are impressive. LLMs can be seen as large text-based surrogate models of the world (or the world how we describe it on the internet), and thus allow reasoning about a large variety of contexts and problems. Reasoning tasks, such as math word problems, were one of the capabilities that LLMs could not achieve, until recently. Let us look more closely at what language models currently can and cannot do.

4.2.1 What Can LLMs Do? With the right prompt, LLMs are able to solve many of the problems in grade school math word benchmarks and beyond. Prompt-based learning is able to perform reasoning tasks such as math word problems, robotic movement, and Python code generation, at inference time, without expensive parameter training.

A taxonomy of generate-evaluate-control is able to describe the structure of the current LLM reasoning literature. Furthermore, the accuracy of the reasoning chains can be improved with ensemble methods, or self-verification. Hallucination can be reduced by grounding the model with external models, such as for robotic affordances, and information retrieval from search engines and Wikipedia. Going a step further, using external control algorithms (such as search or reinforcement learning) as scaffolding, dynamic prompts can use the LLMs to perform complex and dynamic reasoning patterns. Note that the reasoning control is now two layers away from the core LLM: an external control algorithm, on top of in-context-learning, dynamically generating prompts for the LLM. This is reasoning *with* prompts *with* LLMs, not *by*.

At this point, it is interesting to note the confluence of the two schools of classical artificial intelligence, symbolic and connectionist.[4] Search and reinforcement learning are rooted in the symbolic tradition, while LLMs are rooted in the connectionist tradition. The literature in this survey combines the two traditions. High performance reasoning is created with a (symbolic) searcher/learner on top of a (connectionist) LLM. In other fields similar combinations can be seen (for example, AlphaFold [15, 63] and retrosynthesis of molecules [115]). The LLM helps ground symbolic reasoning methods in language; symbolic methods help create prompts that let the LLM perform dynamic reasoning.

We note that benchmarks such as GSM8K have been central for the progress of the field, and that while reasoning started with math word problems, the field has extended to robotics, autonomous agents, games, and most emphatically computer code. Formal languages play an important role in the intermediate multi-step reasoning chains.

A side effect of the work on reasoning is the emergence of a new few-shot learning approach for sequential decision-making processes (SDP)[77]. Traditionally these processes are solved by explicit reinforcement learning (such as DQN [87], PPO [114] and SAC [47]), achieving good results, but suffering from high sample complexity for larger problems [100]. The emergence of few-shot in-context learning for solving SDPs opens a research avenue to find out what SDPs few-shot prompt-learning will be able to solve.

4.2.2 What Can LLMs Not Do? Now that grade school math word problems are largely solvable, harder reasoning benchmarks in other domains are appearing [2]. Another line of research argues that LLMs cannot reason, providing examples where LLMs fail, and discussing potential reasons. Berglund et al. [10] show that LLMs can fail to generalize in surprising ways. They provide the example of a model that is trained to report that "Valentina Tereshkova was the first woman to travel to space," but will not automatically be able to answer the question, "Who was the first woman to travel to space?" pointing to a lack of semantic understanding by LLMs. Other work suggests that results are less generalizable and transferable than often assumed, showing how base-10 arithmetic skills do not transfer to base-9 arithmetic problems [150]. The question which problems LLMs can and cannot solve will continue to motivate researchers.

Other works study the dangers of the size of LLMs. Bender et al. [7] mention the environmental risks associated with the large computational training demands, as well as the difficulty of understanding the training data, for example in the context of bias. Furthermore, there are ethical, legal, and copyright concerns regarding the data that LLMs are

[4]Reasoning and planning have been studied since the start of artificial intelligence, starting with logic and reasoning [90], search algorithms in puzzles and board games [70, 98], robot planning [37], classical machine learning such as decision trees and support vector machines [12, 28, 39], through knowledge representation and the semantic web [133]. Ever since the success of the connectionist approach [44, 72] (deep learning, including LLMs) researchers have tried to join the two approaches.

trained on. Finally, to prevent putting too much trust in the outcome of LLMs, we should understand their failure modes better, such as the well-publicized problems of hallucination (inventing facts that look right but are not). Here, mechanistic interpretability can be used to explore LLM representations and understand where they go wrong [9, 106].

Most of the reasoning capabilities exhibited by LLMs are due to the great representational powers of the transformer architecture, and how in-context learning is able to harness them. Prompt engineering and prompt control play a crucial role in the kind of reasoning that we have seen in the papers. Models can be instructed to write their own reasoning prompts; however, such Auto-GPT or Auto-CoT prompts need careful evaluation, verification, and grounding in the real world, to prevent degeneration into a hallucinatory world of their own. Models can also be instructed to interact with the world, and become the tool of external scaffolding that evaluates, controls and improves the prompts. Some of what we experience as reasoning *by* the LLM, is controlled by the prompt or the scaffolding algorithm. It is an open question if prompt learning is able get the LLM to create a prompt to exhibit dynamic reasoning by itself.

From the symbolic planning field there is a critical view on the reasoning and planning abilities of LLMs [132] giving examples of planning failures. They argue that LLMs are better used instead to improve heuristic elements of traditional planners, such as PDDL [65], to strengthen traditional symbolic planning approaches.

Some of the names of the approaches surveyed in this paper are suggestive of self-awareness and self-reflective capabilities. True (human) self-reflection, or metacognition, is still largely outside the capabilities of current LLMs. LLMs can be prompted to reason, to take small steps, to self-evaluate, and their search process can be controlled by an external algorithm. The self-reflective type of "intelligence" is written into the prompt by the prompt engineer or the control algorithm. We are unaware of any LLM that has been made to reflect on, or even control, its reasoning processes, controlling how many reasoning steps it should take, or limiting its reasoning once the answer had become good enough. True self-reflection remains future work, although some steps have been taken, as we will discuss next.

4.2.3 Reasoning towards Metacognition. Human thought exhibits the ability to reason about self, we are able to think about our own thinking processes. Metacognition studies this phenomenon [136]. Prompted by the success of Chain-of-thought and related works, metacognition has also been studied in the context of LLMs [130].

Many reasoning approaches highlight self-reflective aspects in their names and in how they work. The prompts that prompt the models to reason are being improved with the outcome of the reasoning process, and in Buffer-of-thoughts thought-templates are used that are derived from other reasoning processes. Wang and Zhao [144] study Metacognitive-prompting. Inspired by Chain-of-thought and Self-consistency, they create manually designed prompts to increase the understanding of language models. Figure 25 illustrates the relation between metacognitive human thought processes and metacognitive LLM prompting. Another work, again inspired by Chain-of-thought and Self-consistency, connects psychology and LLMs. Didolkar et al. [31] study metacognitive capabilities of LLMs in mathematical problem solving, both on GSM8K and on the harder MATH problems [49]. First, the model is prompted to find a skill name for each problem instance in the dataset. For 7000 instances of GSM8K, 500 skill names were found by the model. Next, these 500 names are clustered down to 22 skills. They find that by using the names of these 22 skills in Chain-of-thought-like prompts, more problems are solved than with standard Chain-of-Thought/Self-consistency/PAL prompts. Examples of the 22 skill names are *multiplication-and-addition*, *basic-arithmetic*, *subtraction*, and *algebra*. Interestingly, the authors find that the skill exemplar repository that is trained on a strong model (GPT-4), also down-translates to a weak model (GPT-3). The performance of the weak model benefits from the skill-name-enhanced prompts.

Metacognitive reasoning with LLMs is still in its early stages.

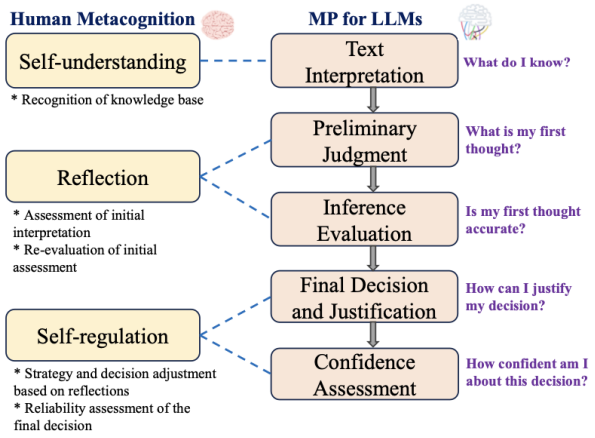

Fig. 25. Metacognitive Prompting and the link with human metacognitive processes [144]

4.3 Research Agenda

At the end of this discussion, we list promising topics for future work. Reasoning with LLMs is an active field of research. It brings together elements of symbolic reasoning, connectionism, natural language, autonomous agents, affective reasoning [13] and metacognition. First we discuss topics for the field of LLM-reasoning itself, then we discuss more general machine learning topics that are important for progress in LLM-reasoning, and finally we discuss more longer term, fundamental topics.

- *Implicit Control*—Search control beyond greedy search is often implemented as an external reinforcement learning algorithm. Is it possible to incorporate the control stage of the reasoning pipeline into one static prompt?
- *Inference-time finetuning*—Can we improve performance further by augmenting finetuning with inference-time reasoning results, as done by [54, 96, 147, 163]? Can we integrate reasoning and finetuning [23, 33, 46]?
- *Code*—Progress in reasoning using formal languages and computer code has been quite promising. GitHub Copilot is a success. Further integration of LLM-reasoning with software engineering tools is a promising area of research that can have a large practical impact on how software is written.
- *Grounding*—Reasoning in LLMs has been successfully applied in autonomous agents, robotics, and games. A challenge is the grounding of the reasoning process in the environment. How can we help LLMs to actively find new information when the reasoning outcome is uncertain? Is the future reasoning-LLM a search engine [137]?

Generic topics in machine learning that also influence prompt-based reasoning research are:

- *Benchmarks*—Progress in LLMs depends on the availability of the right benchmarks. As the field has progressed beyond math word problems, other benchmarks become prevalent, with more difficult and diverse tasks.
- *Faithfulness*—Our theoretical understanding of prompt-based reasoning with LLMs is incomplete. The research on faithfulness highlights one example of our lack of understanding. In general, more insight into the working of multi-step in-context learning in LLMs is dearly needed.
- *Small language models*—Efficiency is an important element for wide adoption of language models. Important topics are distillation of reasoning to small language models and an understanding of scaling laws. Reinforcement learning approaches [117] can improve the efficiency of the finetuning/reasoning pipeline greatly [33, 46].

- *Few-shot Reinforcement Learning*—Small reasoning problems can be solved with few-shot in-context learning. Can we solve larger sequential decision processes, reducing the sample complexity in reinforcement learning?

For longer term future work, the following more fundamental questions are important:

- *Symbolic and Connectionist Computation*—How can we further improve LLM-reasoning: how can LLMs benefit from symbolic reasoning prompts and how can LLMs help ground symbolic reasoning in language?
- *Metacognition*—Much of the research into reasoning guides the model how it should solve a problem. Is it helpful to introduce named concepts for different kinds of reasoning, such as metareasoning [42]? Can the model find these concepts by itself [46]?

5 CONCLUSION

Prompt-based in-context learning is an efficient machine learning method, requiring no parameter updates to the LLM. While achieving good performance on language tasks, performance on reasoning tasks was lacking. Reasoning tasks, such as math word problems, are typically solved in a step-by-step fashion. Recently prompts have been developed that guide an LLM to "think step by step" (Chain-of-thought), and to evaluate and verify the step results. The performance of reasoning with LLMs has improved greatly, and the field has progressed beyond math word problems. Together, the surveyed methods allow the LLM to follow high-quality multi-step reasoning chains. Python code or other formal languages have been used successfully to reduce the error in reasoning steps. Also, in the field of autonomous agents and robotic action, good performance has been achieved by grounding reasoning answers in the environment and the physical constraints of robotic movement.

For complex reasoning tasks a large number of reasoning steps may be generated. To control the size of the reasoning space dynamically, external scaffolding algorithms can be used. Often, variations on search algorithms are used, and especially reinforcement learning. The symbolic and connectionist AI traditions come together in reasoning prompts and search algorithms that help LLM neural networks solve natural language math word and related problems. Inference-time reasoning results can be used to augment finetuning, in a feedback loop.

LLMs hallucinate and suffer from bias, and their use poses different ethical dangers. In multi-step reasoning methods, self-verification methods have been developed to reduce error-accumulation, yet ethical dangers may remain. Additionally, retrieval augmentation methods ground LLM output directly in sources such as Wikipedia.

Among the most popular reasoning benchmarks in this survey is GSM8K, which contains 8500 grade school math word problems. With LLMs such as GPT-3, reasoning approaches show an improvement of 20-50% points over standard prompting methods, and some even more. The success of reasoning with LLMs has attracted more applications, and with them, benchmarks are diverging.

The field of reasoning with LLMs is quite new, and theoretical understanding is lacking in important areas, such as faithful reasoning (models may sometimes find the right answer for the wrong reason). There may be more opportunities for efficiency gains by integrating the training of reasoning models, using reinforcement learning. Although prompt-based learning allows few-shot learning at inference time, the computational needs of LLM pretraining and finetuning are still high, hence the interest in small language models. Reasoning skills that work in large models can often be distilled to small models.

Human thought is capable of metacognition, we can think about our thinking process. Many of the names of the approaches in this survey suggest a link to metacognition (Reflexion, Self-refine, Self-improvement, Inner-monologue). The first preliminary experiments of language models that reason about their reasoning skills have appeared.

LLM-reasoning is an active field of research that shows great progress. Based on current limitations and open questions we provide a research agenda highlighting opportunities for further progress with reinforcement learning, implicit control, inference time finetuning, and small language models, amongst others.

ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their valuable suggestions that have considerably improved the paper.

REFERENCES

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. Large language models for mathematical reasoning: Progresses and challenges. *arXiv preprint arXiv:2402.00157*, 2024.

[3] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on Robot Learning*, 2022.

[4] Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, 2019.

[5] Arian Askari, Roxana Petcu, Chuan Meng, Mohammad Aliannejadi, Amin Abolghasemi, Evangelos Kanoulas, and Suzan Verberne. Self-seeding and multi-intent self-instructing llms for generating intent-aware information-seeking dialogs. *arXiv preprint arXiv:2402.11633*, 2024.

[6] Richard Bellman. Dynamic programming. *science*, 153(3731):34–37, 1966.

[7] Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pages 610–623, 2021.

[8] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

[9] Leonard Bereska and Efstratios Gavves. Mechanistic interpretability for ai safety–a review. *arXiv preprint arXiv:2404.14082*, 2024.

[10] Lukas Berglund, Meg Tong, Max Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. The reversal curse: Llms trained on" a is b" fail to learn" b is a". In *International Conference on Learning Representations*, 2024.

[11] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17682–17690, 2024.

[12] Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

[13] Joost Broekens, Bernhard Hilpert, Suzan Verberne, Kim Baraka, Patrick Gebhard, and Aske Plaat. Fine-grained affective processing capabilities emerging from large language models. In *2023 11th Intl Conf on Affective Computing and Intelligent Interaction (ACII)*, pages 1–8. IEEE, 2023.

[14] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[15] Patrick Bryant, Gabriele Pozzati, and Arne Elofsson. Improved prediction of protein-protein interactions using alphafold2. *Nature communications*, 13(1):1265, 2022.

[16] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[17] Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions of Machine Learning Research; arXiv preprint arXiv:2211.12588*, 2023.

[18] Xinyun Chen, Chen Liang, Adams Wei Yu, Denny Zhou, Dawn Song, and Quoc V Le. Neural symbolic reader: Scalable integration of distributed and symbolic representations for reading comprehension. In *International Conference on Learning Representations*, 2019.

[19] Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to self-debug. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=KuPixIqPiq.

[20] Ting-Rui Chiang and Yun-Nung Chen. Semantically-aligned equation generation for solving and reasoning math word problems. In *Association for Computational Linguistics*, 2019.

[21] Wonje Choi, Woo Kyung Kim, Minjong Yoo, and Honguk Woo. Embodied cot distillation from llm to off-the-shelf agents. In *Forty-first International Conference on Machine Learning*.

[22] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

[23] Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. Sft memorizes, rl generalizes: A comparative study of foundation model post-training. *arXiv preprint arXiv:2501.17161*, 2025.

[24] Zheng Chu, Jingchang Chen, Qianglong Chen, Weijiang Yu, Tao He, Haotian Wang, Weihua Peng, Ming Liu, Bing Qin, and Ting Liu. A survey of chain of thought reasoning: Advances, frontiers and future. In *Association for Computational Linguistics*, 2024.

[25] Zheng Chu, Jingchang Chen, Qianglong Chen, Weijiang Yu, Tao He, Haotian Wang, Weihua Peng, Ming Liu, Bing Qin, and Ting Liu. Navigate through enigmatic labyrinth a survey of chain of thought reasoning: Advances, frontiers and future. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1173–1203, 2024.

[26] Yu-Neng Chuang, Guanchu Wang, Chia-Yuan Chang, Ruixiang Tang, Fan Yang, Mengnan Du, Xuanting Cai, and Xia Hu. Faithlm: Towards faithful explanations for large language models. *arXiv preprint arXiv:2402.04678*, 2024.

[27] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[28] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20:273–297, 1995.

[29] Zhuyun Dai, Vincent Y Zhao, Ji Ma, Yi Luan, Jianmo Ni, Jing Lu, Anton Bakalov, Kelvin Guu, Keith B Hall, and Ming-Wei Chang. Promptagator: Few-shot dense retrieval from 8 examples. In *International Conference on Learning Representations*, 2023.

[30] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Association for Computational Linguistics*, 2019.

[31] Aniket Didolkar, Anirudh Goyal, Nan Rosemary Ke, Siyuan Guo, Michal Valko, Timothy Lillicrap, Danilo Rezende, Yoshua Bengio, Michael Mozer, and Sanjeev Arora. Metacognitive capabilities of llms: An exploration in mathematical problem solving. *arXiv preprint arXiv:2405.12205*, 2024.

[32] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. A survey on in-context learning. In *Association for Computational Linguistics*, 2023.

[33] Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1.5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.

[34] John Dunlosky and Janet Metcalfe. *Metacognition*. Sage Publications, 2008.

[35] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2019.

[36] Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, et al. Minedojo: Building open-ended embodied agents with internet-scale knowledge. *Advances in Neural Information Processing Systems*, 35:18343–18362, 2022.

[37] Richard E Fikes and Nils J Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4): 189–208, 1971.

[38] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[39] Peter Flach. *Machine learning: the art and science of algorithms that make sense of data*. Cambridge university press, 2012.

[40] Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. Complexity-based prompting for multi-step reasoning. In *The Eleventh International Conference on Learning Representations*, 2022.

[41] Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. In *International Conference on Machine Learning*, pages 10764–10799. PMLR, 2023.

[42] Peizhong Gao, Ao Xie, Shaoguang Mao, Wenshan Wu, Yan Xia, Haipeng Mi, and Furu Wei. Meta reasoning for large language models. *arXiv preprint arXiv:2406.11698*, 2024.

[43] Louie Giray. Prompt engineering with chatgpt: a guide for academic writers. *Annals of biomedical engineering*, 51(12):2629–2633, 2023.

[44] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[45] Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations*, 2023.

[46] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[47] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[48] Nikolaus Hansen and Raymond Ros. Black-box optimization benchmarking of newuoa compared to bipop-cma-es: on the bbob noiseless testbed. In *Proceedings of the 12th annual conference companion on Genetic and evolutionary computation*, pages 1519–1526, 2010.

[49] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. In *Advances in Neural Information Processing Systems*, 2021.

[50] Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B Brown, Prafulla Dhariwal, Scott Gray, et al. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*, 2020.

[51] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. In *Advances in Neural Information Processing Systems*, 2022.

[52] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):5149–5169, 2021.

[53] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

[54] Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. Large language models can self-improve. In *Association for Computational Linguistics*, 2023.

[55] Jie Huang and Kevin Chen-Chuan Chang. Towards reasoning in large language models: A survey. In *Assoc for Computational Linguistics*, 2023.

[56] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transaction on Information Systems, Volume 43, Issue 2; arXiv preprint arXiv:2311.05232*, 2025.

[57] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. In *Conference on Robot Learning*, 2022.

[58] Zhen Huang, Haoyang Zou, Xuefeng Li, Yixiu Liu, Yuxiang Zheng, Ethan Chern, Shijie Xia, Yiwei Qin, Weizhe Yuan, and Pengfei Liu. O1 replication journey–part 2: Surpassing o1-preview through simple distillation, big progress or bitter lesson? *arXiv preprint arXiv:2411.16489*, 2024.

[59] Mike Huisman, Jan N Van Rijn, and Aske Plaat. A survey of deep meta-learning. *Artificial Intelligence Review*, 54(6):4483–4541, 2021.

[60] Shima Imani, L Du, and H Shrivastava. Mathprompter: Mathematical reasoning using large language models. In *Association for Computational Linguistics*, 2023.

[61] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Computer vision and pattern recognition*, pages 2704–2713, 2018.

[62] Mingyu Jin, Qinkai Yu, Haiyan Zhao, Wenyue Hua, Yanda Meng, Yongfeng Zhang, Mengnan Du, et al. The impact of reasoning step length on large language models. In *Association for Computational Linguistics*, 2024.

[63] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589, 2021.

[64] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4: 237–285, 1996.

[65] Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Kaya Stechly, Mudit Verma, Siddhant Bhambri, Lucas Saldyt, and Anil Murthy. Llms can't plan, but can help planning in llm-modulo frameworks. In *International Conference on Machine Learning*, 2024.

[66] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[67] Tom Kocmi, Rachel Bawden, Ondřej Bojar, Anton Dvorkovich, Christian Federmann, Mark Fishel, Thamme Gowda, Yvette Graham, Roman Grundkiewicz, Barry Haddow, et al. Findings of the 2022 conference on machine translation (wmt22). In *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 1–45, 2022.

[68] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

[69] Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. Mawps: A math word problem repository. In *2016 conference of the north american chapter of the association for computational linguistics: human language technologies*, pages 1152–1157, 2016.

[70] Richard E Korf. *Algorithms and theory of computation handbook, 22-17*, chapter Artificial intelligence search algorithms. Citeseer, 1999.

[71] Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, et al. Measuring faithfulness in chain-of-thought reasoning. *arXiv preprint arXiv:2307.13702*, 2023.

[72] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[73] Lei Li, Yongfeng Zhang, and Li Chen. Prompt distillation for efficient llm-based recommendation. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 1348–1357, 2023.

[74] Shiyang Li, Jianshu Chen, Yelong Shen, Zhiyu Chen, Xinlu Zhang, Zekun Li, Hong Wang, Jing Qian, Baolin Peng, Yi Mao, et al. Explanations from large language models make small reasoners better. In *Association for the advancement of artificial intelligence*, 2024.

[75] Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. Making language models better reasoners with step-aware verifier. In *Association for Computational Linguistics*, 2023.

[76] Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Association for Computational Linguistics*, 2017.

[77] Michael Lederman Littman. *Algorithms for sequential decision-making*. Brown University, 1996.

[78] Fei Liu, Tong Xialiang, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. Evolution of heuristics: Towards efficient automatic algorithm design using large language model. In *Forty-first International Conference on Machine Learning*, 2024.

[79] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35, 2023.

[80] Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *International Conference on Learning Representations*, 2024.

[81] Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. Faithful chain-of-thought reasoning. In *Association for Computational Linguistics*, 2023.

[82] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36, 2023.

[83] Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. Teaching small language models to reason. In *Association for Computational Linguistics*, 2023.

[84] Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. A diverse corpus for evaluating and developing english math word problem solvers. In *Annual Meeting of the Association for Computational Linguistics*, 2020. URL https://api.semanticscholar.org/CorpusID:220047831.

[85] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. In *International Conference on Learning Representations*, 2018.

[86] Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. Large language models: A survey. *arXiv preprint arXiv:2402.06196*, 2024.

[87] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[88] Philipp Mondorf and Barbara Plank. Beyond accuracy: Evaluating the reasoning behavior of large language models–a survey. In *Conference on Language Modeling, Philadelphia*, 2024.

[89] Shashi Narayan, Shay B Cohen, and Mirella Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Association for Computational Linguistics*, 2018.

[90] Allen Newell and Herbert A Simon. Computer simulation of human thinking: A theory of problem solving expressed as a computer program permits simulation of thinking processes. *Science*, 134(3495):2011–2017, 1961.

[91] Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, et al. Show your work: Scratchpads for intermediate computation with language models. In *ICLR 2022 Workshop DL4C*, 2022.

[92] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

[93] Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The lambada dataset: Word prediction requiring a broad discourse context. In *Association for Computational Linguistics*, 2016.

[94] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[95] Arkil Patel, S Bhattamishra, and N Goyal. Are nlp models really able to solve simple math word problems? In *Association for Computational Linguistics*, 2021.

[96] Debjit Paul, Mete Ismayilzada, Maxime Peyrard, Beatriz Borges, Antoine Bosselut, Robert West, and Boi Faltings. Refiner: Reasoning feedback on intermediate representations. In *Association for Computational Linguistics*, 2024.

[97] Debjit Paul, Robert West, Antoine Bosselut, and Boi Faltings. Making reasoning matter: Measuring and improving faithfulness of chain-of-thought reasoning. In *Association for Computational Linguistics*, 2024.

[98] Aske Plaat. *Learning to play: reinforcement learning and games*. Springer Nature, 2020.

[99] Aske Plaat. *Deep reinforcement learning*. Springer, Singapore, 2022.

[100] Aske Plaat, Walter Kosters, and Mike Preuss. High-accuracy model-based reinforcement learning, a survey. *Artificial Intelligence Review*, 56(9): 9541–9573, 2023.

[101] Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Association for Computational Linguistics*, 2023.

[102] Moschoula Pternea, Prerna Singh, Abir Chakraborty, Yagna Oruganti, Mirco Milletari, Sayli Bapat, and Kebei Jiang. The rl/llm taxonomy tree: Reviewing synergies between reinforcement learning and large language models. *Journal of Artificial Intelligence Research, Volume 80; arXiv preprint arXiv:2402.01874*, 2024.

[103] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[104] Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*, 2021.

[105] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024.

[106] Daking Rai, Yilun Zhou, Shi Feng, Abulhair Saparov, and Ziyu Yao. A practical review of mechanistic interpretability for transformer-based language models. *arXiv preprint arXiv:2407.02646*, 2024.

[107] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Association for Computational Linguistics*, 2016.

[108] Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, et al. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024.

[109] Subhro Roy and Dan Roth. Solving general arithmetic word problems. In *Association for Computational Linguistics*, 2015.

[110] Baptiste Roziere, Marie-Anne Lachaux, Lowik Chanussot, and Guillaume Lample. Unsupervised translation of programming languages. *Advances in neural information processing systems*, 33:20601–20611, 2020.

[111] Omer Sagi and Lior Rokach. Ensemble learning: A survey. *Wiley interdisciplinary reviews: data mining and knowledge discovery*, 8(4):e1249, 2018.

[112] Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. A systematic survey of prompt engineering in large language models: Techniques and applications. *arXiv preprint arXiv:2402.07927*, 2024.

[113] Abulhair Saparov and He He. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. In *International Conference on Learning Representations*, 2023.

[114] John Schulman, Filip Wolski, P Dhariwal, A Radford, and O Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[115] Marwin Segler, M Preuss, and M Waller. Planning chemical syntheses with deep neural networks and symbolic ai. *Nature*, 555(7698):604–610, 2018.

[116] Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving neural machine translation models with monolingual data. In *Association for Computational Linguistics*, 2016.

[117] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[118] Jianhao Shen, Yichun Yin, Lin Li, Lifeng Shang, Xin Jiang, Ming Zhang, and Qun Liu. Generate & rank: A multi-task framework for math word problems. In *Association for Computational Linguistics*, 2021.

[119] Zhengliang Shi, Shen Gao, Xiuyi Chen, Yue Feng, Lingyong Yan, Haibo Shi, Dawei Yin, Zhumin Chen, Suzan Verberne, and Zhaochun Ren. Chain of tools: Large language model is an automatic multi-tool learner. *arXiv preprint arXiv:2405.16533*, 2024.

[120] Zhengliang Shi, Shen Gao, Xiuyi Chen, Lingyong Yan, Haibo Shi, Dawei Yin, Zhumin Chen, Pengjie Ren, Suzan Verberne, and Zhaochun Ren. Learning to use tools via cooperative and interactive agents. In *Association for Computational Linguistics*, 2024.

[121] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[122] Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021.

[123] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. In *Conference on robot learning*, pages 894–906. PMLR, 2022.

[124] Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3:9–44, 1988.

[125] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT Press, 2018.

[126] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Association for Computational Linguistics*, 2019.

[127] Yiming Tan, Dehai Min, Yu Li, Wenbo Li, Nan Hu, Yongrui Chen, and Guilin Qi. Can ChatGPT replace traditional KBQA models? An in-depth analysis of the question answering performance of the GPT LLM family. In *International Semantic Web Conference*, pages 348–367. Springer, 2023.

[128] Yi Tay, Mostafa Dehghani, Vinh Q Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Siamak Shakeri, Dara Bahri, Tal Schuster, et al. Ul2: Unifying language learning paradigms. In *International Conference on Learning Representations*, 2023.

[129] Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*, 2022.

[130] Jason Toy, Josh MacAdam, and Phil Tabor. Metacognition is all you need? using introspection in generative agents to improve goal-directed behavior. *arXiv preprint arXiv:2401.10910*, 2024.

[131] Miles Turpin, Julian Michael, Ethan Perez, and Samuel Bowman. Language models don't always say what they think: unfaithful explanations in chain-of-thought prompting. *Advances in Neural Information Processing Systems*, 36, 2024.

[132] Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. On the planning abilities of large language models-a critical investigation. *Advances in Neural Information Processing Systems*, 36:75993–76005, 2023.

[133] Frank Van Harmelen, Vladimir Lifschitz, and Bruce Porter. *Handbook of knowledge representation*. Elsevier, 2008.

[134] Niki van Stein and Thomas Bäck. Llamea: A large language model evolutionary algorithm for automatically generating metaheuristics. *IEEE Transactions on Evolutionary Computation; arXiv preprint arXiv:2405.20132*, 2024.

[135] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[136] Marcel VJ Veenman, Bernadette HAM Van Hout-Wolters, and Peter Afflerbach. Metacognition and learning: Conceptual and methodological considerations. *Metacognition and learning*, 1:3–14, 2006.

[137] Suzan Verberne. *Is the search engine of the future a chatbot?* Inaugural lecture, Leiden University, 2024.

[138] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Association for Computational Linguistics*, 2018.

[139] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32, 2019.

[140] Boshi Wang, Sewon Min, Xiang Deng, Jiaming Shen, You Wu, Luke Zettlemoyer, and Huan Sun. Towards understanding chain-of-thought prompting: An empirical study of what matters. In *Association for Computational Linguistics*, 2023.

[141] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions of Machine Learning Research; arXiv preprint arXiv:2305.16291*, 2024.

[142] Huaijie Wang, Shibo Hao, Hanze Dong, Shenao Zhang, Yilin Bao, Ziran Yang, and Yi Wu. Offline reinforcement learning for llm multi-step reasoning. *arXiv preprint arXiv:2412.16145*, 2024.

[143] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023.

[144] Yuqing Wang and Yun Zhao. Metacognitive prompting improves understanding in large language models. In *Association for Computational Linguistics*, 2024.

[145] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *Transactions of Machine Learning Research; arXiv preprint arXiv:2206.07682*, 2022.

[146] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[147] Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin Choi. Generating sequences by learning to self-correct. In *International Conference on Learning Representations*, 2023.

[148] Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. Large language models are better reasoners with self-verification. In *Association for Computational Linguistics*, 2023.

[149] Siwei Wu, Zhongyuan Peng, Xinrun Du, Tuney Zheng, Minghao Liu, Jialong Wu, Jiachen Ma, Yizhi Li, Jian Yang, Wangchunshu Zhou, et al. A comparative study on reasoning patterns of openai's o1 model. *arXiv preprint arXiv:2410.13639*, 2024.

[150] Zhaofeng Wu, Linlu Qiu, Alexis Ross, Ekin Akyürek, Boyuan Chen, Bailin Wang, Najoung Kim, Jacob Andreas, and Yoon Kim. Reasoning or reciting? exploring the capabilities and limitations of language models through counterfactual tasks. In *2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1819–1862, 2024.

[151] Yisheng Xiao, Lijun Wu, Junliang Guo, Juntao Li, Min Zhang, Tao Qin, and Tie-yan Liu. A survey on non-autoregressive generation for neural machine translation and beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[152] Ziyang Xiao, Dongxiang Zhang, Yangjun Wu, Lilin Xu, Yuan Jessica Wang, Xiongwei Han, Xiaojin Fu, Tao Zhong, Jia Zeng, Mingli Song, et al. Chain-of-experts: When llms meet complex operations research problems. In *12th International Conference on Learning Representations*, 2023.

[153] Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, James Xu Zhao, Min-Yen Kan, Junxian He, and Michael Xie. Self-evaluation guided beam search for reasoning. *Advances in Neural Information Processing Systems*, 36, 2024.

[154] Fengli Xu, Qianyue Hao, Zefang Zong, Jingwei Wang, Yunke Zhang, Jingyi Wang, Xiaochong Lan, Jiahui Gong, Tianjian Ouyang, Fanjin Meng, et al. Towards large reasoning models: A survey on scaling llm reasoning capabilities. *arXiv preprint arXiv:2501.09686*, 2025.

[155] Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. A survey on knowledge distillation of large language models. *arXiv preprint arXiv:2402.13116*, 2024.

[156] Ling Yang, Zhaochen Yu, Tianjun Zhang, Shiyi Cao, Minkai Xu, Wentao Zhang, Joseph E Gonzalez, and Bin Cui. Buffer of thoughts: Thought-augmented reasoning with large language models. In *Advances in Neural Information Processing Systems*, 2024.

[157] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022.

[158] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

[159] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36, 2024.

[160] Fei Yu, Hongbo Zhang, Prayag Tiwari, and Benyou Wang. Natural language reasoning, a survey. *ACM Computing Surveys*, 2023.

[161] Ping Yu, Jing Xu, Jason Weston, and Ilia Kulikov. Distilling system 2 into system 1. *arXiv preprint arXiv:2407.06023*, 2024.

[162] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, et al. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Association for Computational Linguistics*, 2018.

[163] Eric Zelikman, Jesse Mu, Noah D Goodman, and Yuhuai Tony Wu. Star: Self-taught reasoner bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[164] Yizhuo Zhang, Heng Wang, Shangbin Feng, Zhaoxuan Tan, Xiaochuang Han, Tianxing He, and Yulia Tsvetkov. Can llm graph reasoning generalize beyond pattern memorization? In *Association for Computational Linguistics*, 2024.

[165] Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. Automatic chain of thought prompting in large language models. In *International Conference on Learning Representations*, 2023.

[166] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.

[167] Chuanyang Zheng, Zhengying Liu, Enze Xie, Zhenguo Li, and Yu Li. Progressive-hint prompting improves reasoning in large language models. *arXiv preprint arXiv:2304.09797*, 2023.

[168] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36, 2024.

[169] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. Least-to-most prompting enables complex reasoning in large language models. In *International Conference on Learning Representations*, 2023.