Published: 09 December 2025
Online AM: 05 November 2025
Accepted: 02 November 2025
Revised: 25 October 2025
Received: 17 July 2024

SURVEY

Multi-Step Reasoning with Large Language Models, a Survey

ASKE PLAAT, Leiden University, Leiden, Zuid-Holland, Netherlands

ANNIE WONG, Leiden University, Leiden, Zuid-Holland, Netherlands

SUZAN VERBERNE, Leiden University, Leiden, Zuid-Holland, Netherlands

JOOST BROEKENS, Leiden University, Leiden, Zuid-Holland, Netherlands

NIKI VAN STEIN, Leiden University, Leiden, Zuid-Holland, Netherlands

THOMAS H W BÄCK, Leiden University, Leiden, Zuid-Holland, Netherlands

Multi-Step Reasoning with Large Language Models, a Survey

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

Large language models (LLMs) with billions of parameters exhibit in-context learning abilities, enabling few-shot learning on tasks that the model was not specifically trained for. Traditional models achieve breakthrough performance on language tasks, but do not perform well on basic reasoning benchmarks. However, a new in-context learning approach, Chain-of-thought, has demonstrated strong multi-step 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 article reviews the field of multi-step 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 find that multi-step reasoning approaches have progressed beyond math word problems, and can now successfully solve challenges in logic, combinatorial games, and robotics, sometimes by first generating code that is then executed by external tools. Many studies in multi-step methods use reinforcement learning for finetuning, external optimization loops, in-context reinforcement learning, and self-reflection.

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

Additional Key Words and Phrases: Large language models, chain of thought, reasoning, self-reflection, reinforcement learning, prompt learning

ACM Reference Format:
Aske Plaat, Annie Wong, Suzan Verberne, Joost Broekens, Niki Van Stein, and Thomas Bäck. 2025. Multi-Step Reasoning with Large Language Models, a Survey. *ACM Comput. Surv.* 58, 6, Article 160 (December 2025), 35 pages. https://doi.org/10.1145/3774896

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

Authors' Contact Information: Aske Plaat (corresponding author), Computer Science, Leiden University, Leiden, ZH, Netherlands; e-mail: aske.plaat@gmail.com; Annie Wong, Computer Science, Leiden University, Leiden, ZH, Netherlands; e-mail: a.s.w.wong@liacs.leidenuniv.nl; Suzan Verberne, Computer Science, Leiden University, Leiden, ZH, Netherlands; e-mail: s.verberne@liacs.leidenuniv.nl; Joost Broekens, Computer Science, Leiden University, Leiden, ZH, Netherlands; e-mail: d.j.broekens@liacs.leidenuniv.nl; Niki Van Stein, Computer Science, Leiden University, Leiden, ZH, Netherlands; e-mail: n.van.stein@liacs.leidenuniv.nl; Thomas Bäck, Computer Science, Leiden University, Leiden, Zuid-Holland, Netherlands; e-mail: t.h.w.baeck@liacs.leidenuniv.nl.

ACM 0360-0300/2025/12-ART160
https://doi.org/10.1145/3774896

prediction [121, 160, 172]; they are good at natural language understanding (GLUE, SQUAD, Xsum) [104, 125, 163, 164], translation [74, 111, 136], **question answering** (**QA**) [150], and other language generation tasks. The success of models such as ChatGPT [107] 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* [16]. 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 [172]. 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, tasks that require reasoning, such as grade school math word problems, are more difficult for LLMs [31]. 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** (**CoT**) paper by Wei et al. [173] stands out. This work, and later work by Kojima et al. [75], 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 to answer difficult problems in a multi-step approach. Subsequently, performance on math word benchmarks has increased markedly. Much of the performance increase of models such as OpenAI o1, o3 [62] and DeepSeek R1 [50, 137] is attributed to multi-step reasoning methods as reviewed here, with **reinforcement learning** (**RL**) playing an increasingly important role, both for in-context learning and finetuning [26, 38, 177, 182].

The line of research into multi-step LLM-reasoning was started with grade school math word problems, with the GSM8K benchmark [31]. Soon, other reasoning domains were included, such as reasoning about advanced math problems, computer code, robotic movement, and games. Many works have been published that build on CoT [27]. In this article, we survey the literature using a straightforward taxonomy. We discuss papers based on several reasoning benchmarks, as well as directly-related follow up work.

Having started with basic math word problems, multi-step LLM reasoning approaches now perform logic reasoning, planning, combinatorial games, and robotic actions, amongst others. Some approaches do so by prompting the LLM to generate code that is then interpreted by external tools. Many studies in multi-step methods are using RL, for finetuning, **in context reinforcement learning** (**ICRL**), external optimization loops, and self-reflection. The main contributions of this article are: (1) we provide a survey of relevant approaches in multi-step reasoning with LLMs, (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.

This survey is organized as follows. Section 2.1 provides background information on the most relevant developments in LLMs, including in-context learning and finetuning. Of great importance are the benchmarks that are used in this field (Section 2.4). Next, in Section 3 we provide a taxonomy of the field, where we discuss the main approaches in detail. Then, in Section 4 we discuss our findings in a broader perspective. We also discuss the relation between multi-step 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, Scope, and Selection of Papers

Reasoning has a long and active history in AI, in logical inference, and in other fields, such as commonsense and analogical reasoning [83, 100]. Indeed, some of the early criticism on LLMs was that they could not reason, that they made obvious and basic reasoning errors, showing a lack of

understanding. Berglund et al. [10] provide the example of a model that is trained to report that "Valentina Tereshkova was the first woman to travel to space," but is not able to answer the question, "Who was the first woman to travel to space?" pointing to a lack of semantic understanding. 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 [178]. More recently, LLMs are also shown to struggle with analogies and analogical reasoning [83, 100] (see Sections 3.3.3 and 4.3.3). Different kinds of reasoning play an important role in LLM research.

2.1 Scope

The question whether LLMs can reason prompted the development of the GSM8k benchmark [31] in 2021. GSM8k is a benchmark of 8500 easy reasoning problems (of the type: "Question: Natalia sold clips to 48 of her friends in April, and then she sold half as many clips in May. How many clips did Natalia sell altogether in April and May? Answer: 72"). This benchmark was specifically designed to test if LLMs can answer basic mathematical multi-step reasoning questions, or if they can not reason at all. Initially, performance was poor, but a year later Wei et al. [173] showed that by providing multi-step examples in the prompt—a *CoT*—much better performance was possible. Many works followed, and reasoning in language models has become an active area of study. In this survey, we focus on prompt-based multi-step reasoning algorithms. After starting with math word problems, work has more recently been extended to combinatorial games, puzzles, robotics, logic, and other fields of reasoning. At the end of this survey, we discuss connections to other fields, such as self-reflection and ICRL.

Before we discuss the works on reasoning, this section reviews background terminology on LLMs. Our overview is brief and we focus on the multi-step reasoning approaches that were inspired by CoT. Excellent related surveys exist. For a longer overview of general LLM topics, see, for example, the surveys Minaee et al. [99] and Zhao et al. [194]. For surveys that focus on the nature of reasoning and its definition, see [28, 58, 189]. For surveys on evaluating logical reasoning in LLMs, see [91, 102, 109]. For a survey on RL in reasoning, see [182], for reasoning in language see [189], and for a survey on CoT itself, see [27].

The papers for 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 *LLM* in their title or abstract, with a focus on papers that reference the CoT paper. Although multi-step 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 CoT, that are used in reasoning LLMs such as OpenAI o1 and o3 [62, 177, 182]. We include a few papers that work with external algorithms, finetuning, or supervised learning, that have contributed significantly to the in-context approaches.

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.

2.2 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. [99].

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. [16]. The next

stage is *pretraining* the transformer model [160] on this large corpus. This stage yields a generative language model. Pretraining is done using self-supervised autoregressive training on the unlabeled dataset (text corpus). Then the general model is *finetuned* to a specific (narrow) task, for example, QA. This can be done using supervised learning with a new labeled dataset consisting of prompts and answers (supervised finetuning, SFT), for example with low-rank optimization [57, 99, 172], or RL [26, 147]. 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, preventing, for example, hate speech. A machine learning method that is commonly used in this stage is RL with Human Feedback [107], Direct Preference Optimization [123], or RL with Verifiable Rewards [79]. Optionally, model training can be *optimized* to improve cost-effectiveness, for example, mixed precision training [98], quantization [65], or knowledge distillation [49, 184].

Once the model has been trained in the steps described above, it can be used further in 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 [16, 37]; *in-context learning*, or *prompt-learning*, takes place in this stage.

In-context learning has a low barrier of entry. Since there is no need for expensive pretraining or finetuning, this form of advanced prompt engineering has become a popular training method. 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.

2.3 In-Context Learning

In large models, beyond hundreds of billions of parameters, a new kind of machine learning has emerged, that has been called *in-context learning* or *prompt-learning* [16]. It occurs not when the model's parameters are trained, but when the model is used, at inference time. Since no parameters are changed at this stage, it is not a *model training* stage; in-context-learning "learns" inside the context, or prompt, using information that is already encoded in the trained model parameters and 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, learning from the few examples in the prompt in combination with the knowledge of the model. The large size of the model, containing rich and general knowledge, is enabling the few-shot learning (see Dong et al. [37] 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 decoder model, for text generation [92]. 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 [92]. 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.

Contexts that contain a few examples are said to perform few-shot learning. Contexts that contain only an instruction, 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

Table 1. Accuracy of GPT-3 and CoT on Popular Math Word Problems Benchmarks [173]

Benchmark	GPT-3 175B	Chain-of-thought [173]
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

used during training to update model parameters with backward propagation in lengthy and costly training epochs [48]. Indeed, in-context learning takes place fully at inference time, no parameters are trained, instead, *learning* now refers to adjusting the answers to the examples in 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, [42] or [56, 63] for a survey). In-context learning, in comparison, is computationally cheap, and has become a popular research approach.

Prompts provide a user-friendly interface to LLMs. The success of in-context learning tends to be quite sensitive to the way in which a prompt is formulated; a new field called *prompt engineering* has emerged to help human users to learn how to make LLMs do what they want them to do [47, 121, 131, 172]. The current survey thus discusses advanced prompt engineering methods.

2.4 Multi-Step 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 QA (for example, CommonsenseQA [149]), word prediction (for example, LAMBADA [110]), translation (for example, WMT'22 [74]), language understanding (for example, GLUE [163, 164]), and text summarization (for example, Xsum [104]).

The field of LLMs is quite active. We will mention relevant benchmarks for testing the multi-step reasoning abilities of LLMs. The research on reasoning with LLMs started with math word problem benchmarks. The benchmark that is most frequently associated with multi-step reasoning with LLMs is the dataset of grade school math word problems GSM8K [31]. 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 CoT 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 [112], the ASDiv dataset of diverse math problems [97], the AQuA dataset of algebraic word problems [88], and the MAWPS benchmark [76]. Table 1 summarizes the accuracy of CoT on these basic math word problems, against the baseline of GPT-3 175B as LLM [173], as percentage of benchmark questions answered correctly. We see that CoT 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 Advanced mathematical questions [32], Computer code comprehension (Human evaluation, Spider [190], Transcoder [128]), Robotic movement (Alfworld [143], Kitchen [2]), Puzzle solving

(Game24 [187]), Creative writing [187]), Gaming (Checkmate problems[185], MineCraft [40]), and Webpage navigation (WebShop [186]). These benchmarks are used by other approaches in our survey, as we will see in more detail in Section 3.

The scope of this survey is limited to multi-step reasoning approaches. Other studies have published more challenging benchmarks, to study the performance of CoT and self-reflection in LLMs in puzzles and (logic) games [108, 129, 175].

In this survey we will mention benchmark results as reported by the surveyed papers, for the sake of concreteness. However, it can be difficult to directly compare different benchmarking results, since there may be differences in the way the experiments were conducted. Kamoi et al. [70] provide a critical analysis of benchmarking problems in LLM reasoning. Despite these challenges, general conclusions can be drawn. Section 4.1 provides guidance on how to choose reasoning approaches for different problem types.

3 Step Generation, Evaluation, and Control

This survey examines how LLMs based on the transformer architecture can be prompted to solve multi-step reasoning tasks. The CoT paper shows how a simple command can prompt an LLM to perform reasoning steps, yielding much better performance in math word problems. 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 [95]:

(1) **Generate**: prompt the model to generate reasoning steps,
(2) **Evaluate**: prompt the model to evaluate the generated steps,
(3) **Control**: prompt the model to control 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. 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, and to 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 has, and finally how many they have 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 (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.

[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).

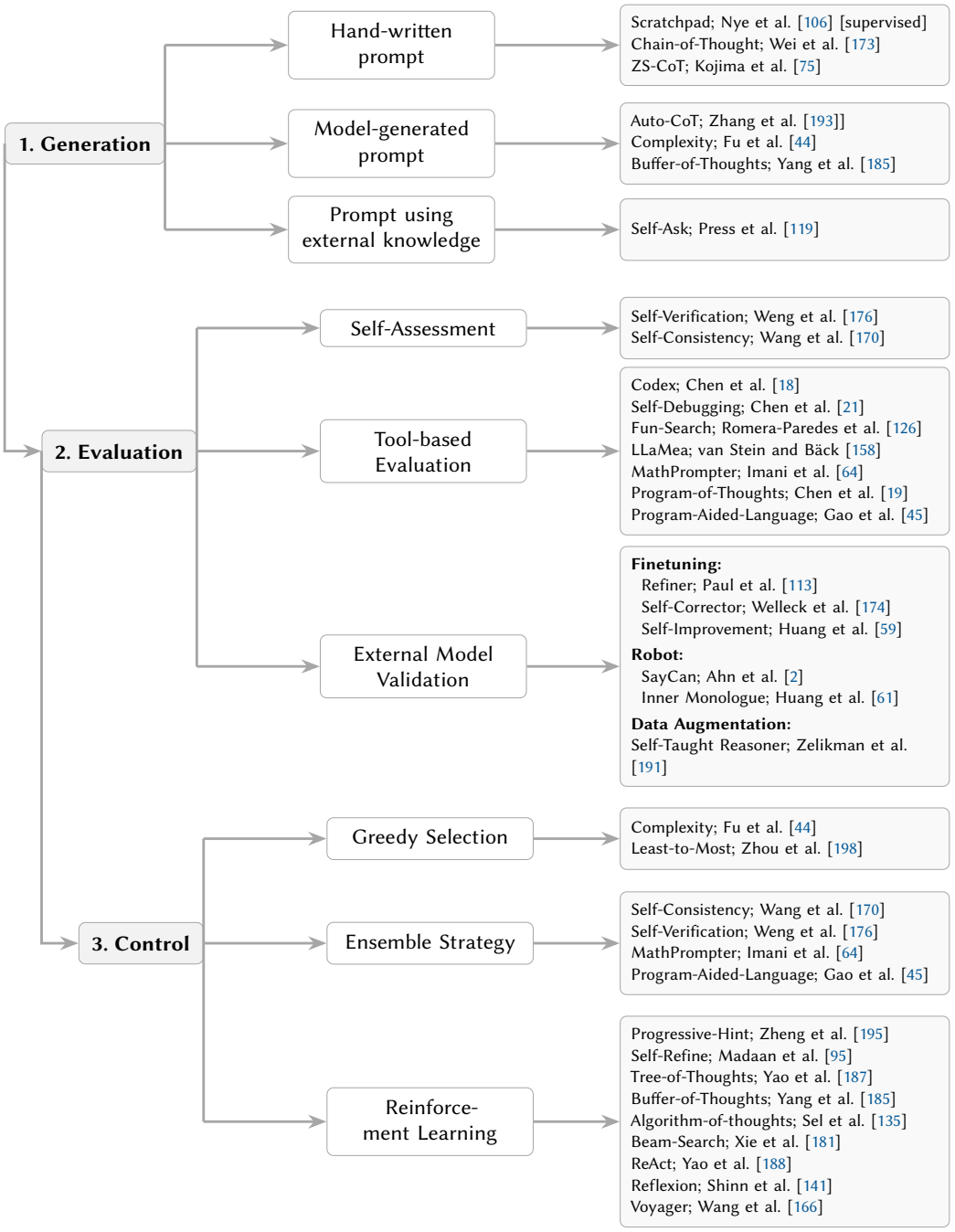

Fig. 1. Taxonomy of LLM-Reasoning approaches: prompt generation, evaluation, and control.

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.

Table 2. Taxonomy of Approaches: Generation, Evaluation, and Control

Approach	Domain	(3.1) Step generation	(3.2) Step evaluation	(3.3) Step control	Result
Scratchpad [106]	math word	**hand-wr/superv**	-	greedy/1prompt	PolyEval +19%, Python +21%
Chain-of-thought [173]	math word	**hand-written**	-	greedy/1prompt	GSM8K +39%, SVAMP +10%, ASDiv +2%, AQuA +11%, MAWPS +14%, CSQA +1.8%, StrategyQA +0.2%
ZS-CoT [75]	math word	**hand-written**	-	greedy/1prompt	MultiArith =89%, GSM8K =70%
Auto-CoT [193]	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 [44]	math word	**hand-written**	self-consistency	greedy/1prompt	GSM8K +7%, MultiArith +3%, Penguins +3%
Self-ask [119]	math word	**ext knowledge**	LLM	multi-hop q.	Bamboogle =60%, 2Wiki =40%, Musique =15%
Self-verification [176]	math word	hand-written	**back-verify**	ensemble	GSM8K +4%, SingleEq +2%, AddSub +4%, MultiArith +3%, AQuA +3%, SVAMP +1%
Self-consistency [170]	math word	hand-written	**majority**	ensemble	GSM8K +18%, SVAMP +11%, AQuA +12%, StrategyQA +6%, ARC-c +4%
Codex [18]	code	-	**tool-based**	-	HumanEval =70%
Self-debugging [21]	code	hand-written	**tool-based**	greedy	Spider +9%, MBPP +12%, TransCoder +12%
Fun-search [126]	code	hand-written	**tool-based**	evolutionary alg	cap set 8 =512
LLaMEa [158]	code	hand-written	**tool-based**	evolutionary alg	BBOB +11%
MathPrompter [64]	math	hand-written	**tool-based**	ensemble	MultiArith =92%
Program-of-thoughts [19]	math word	hand-wr, Codex	**Python+Consist.**	split reason/cmput	GSM8K =71%, SVAMP =85%, ASDIV =85%, AddSub =92%, MultiArith = 99%
Program-aided-lang [45]	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 [113]	math word	finetune	**critic model**	gen/crit feedback	SVAMP =72%, GSM8K =78%
Self-correction [174]	math word	finetune	**corrector model**	gen/corr feedback	MathProgSynth =24%, LexConstrGen =98%, ToxicityControl =0.0%
Self-improvement [59]	math word	finetune	**self-assessment**	CoT/consistency	GSM8K =82%, DROP =83%, ARC-c =90%, OpenBookQA =94%, ANLI-A3 =68%
Say-can [2]	robot	model-generated	**external model**	greedy	Kitchen =31%
Inner-monologue [61]	robot	hand-written	**various**	greedy	TableTop =90%, Kitchen =60%
Self-taught-reasoner [191]	math word	finetune	**augmentation**	greedy/feedback	CommonsenseQA =72%
Least-to-most [198]	math word	hand-written	self-assessment	**curriculum**	SCAN =99%
Progressive-hint [195]	math word	model-generated	self-assessment	**stable prompt**	AddSub +2%, MultiArith +0%, SingleEQ +2%, SVAMP +3%, GSM8K +5%, AQuA +1%
Self-refine [95]	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 [187]	puzzles	model-generated	self-assessment	**BFS/DFS**	Game24 =74%, CreativeWriting , Crossword
Buffer-of-thoughts [185]	math word	thought template	self-assessment	**buffer manager**	Game24 +11%, GeoShapes +20%, Checkmate +51%
Algorithm-of-thoughts [135]	puzzles	model-generated	self-assessment	**in-context RL**	GSM8K =89%, StrategyQA =84%, Crossword
Beam-search [181]	math word	model-generated	self-assessment	**Beam Search**	GSM8K +6%, AQuA +9%, StrategyQA +5%
ReAct [188]	action	external knowledge	self-assessment	**reinf learning**	ALFWorld =34%, WebShop =10%
Reflexion [141]	decision	model-generated	ext model	**reinf learning**	HumanEval =91%
Voyager [166]	Minecraft	model-generated	Minecraft	**reinf learning**	15 x faster

Reported benchmark results: '=' is absolute score, '+' is improvement to a baseline.

(3.2) Evaluation. After the prompt has been generated and the model has answered it, the next stage is to evaluate the quality of this answer. Such evaluation is often necessary to improve the answer and perform well on the benchmark. 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 of 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*.

(3.3) Control. The third stage is control. A reasoning process that consists of multiple steps is a sequential decision process [89]. 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)* RL algorithm can be used as scaffolding [116]. In this case, when a step is followed and evaluated, the LLM can roll back and try a different reasoning step. Going further, a full RL approach can be used [117, 147] to find an optimal policy for the sequential decision process. In general a Markov Decision Process of state, action, transition, and reward function can be specified, and step control can become a process where prompts are generated dynamically under the control of an external RL algorithm, or as ICRL [82, 101].

Taxonomy Table. Table 2 lists the 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 CoT 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 in-context 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 CoT 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 that are trained on computer code achieve good performance at code generation, Chen et al. [18] report up to 77% accuracy on the HumanEval dataset. Gao et al. [45] leverage this ability by translating problems into code, achieving between 61% and 99% accuracy on various mathematics tasks. 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

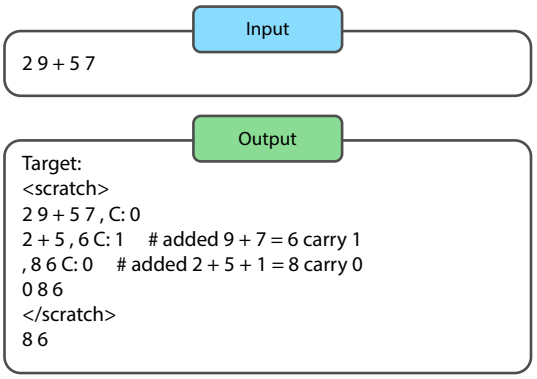

Fig. 2. Example of input and target for supervised learning on a *long addition* problem of adding two numbers. The carry is recorded in the C: digit. Comments (after #) are not part of the learning target (adapted from [106]).

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 RL 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 benchmarks, such as the Creative Writing benchmark in Tree-of-thoughts, results are best reported qualitatively. The source papers provide more measurement details, Section 4.1 discusses when to which approach to use for different applications.

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 [31]. Different approaches were tried unsuccessfully, for example scaling up the size of the LLM [122]. 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 toward 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. [106] 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. [173], amongst others. A basic way to instruct an LLM to generate steps by prompt-learning is to manually write a prompt

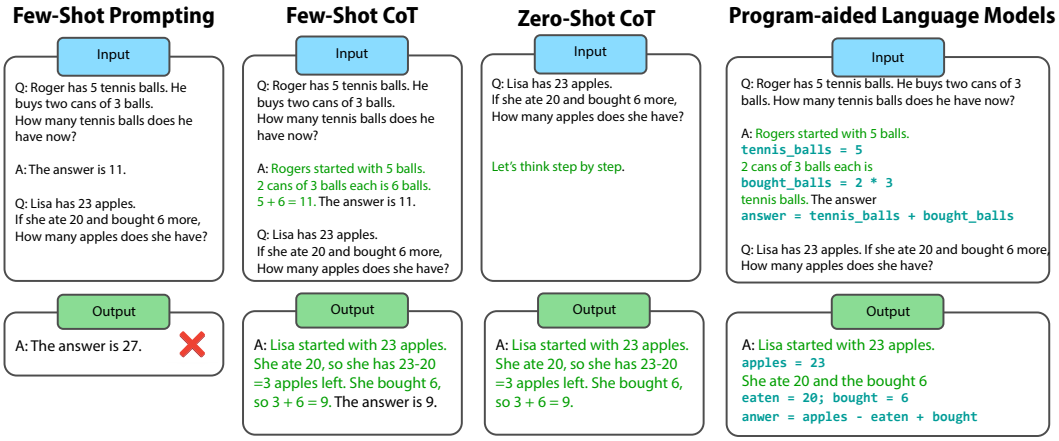

Fig. 3. Different CoT prompting techniques. At the top the prompts, at the bottom the answers. When shown the longer example prompt, the LLM follows the longer example when answering the question (Few-Shot CoT [173]). Without example answer and using *Let's think step by step* results in similar answers (Zero-Shot CoT [75]). With Program-aided-language models [45] similar reasoning can be achieved.

for the LLM 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 CoT paper is shown in Figure 3. Table 1 shows that the largest accuracy increase is on GSM8K, from 16% to 47%.

The idea that an LLM can be made to follow step-by-step instructions, and the performance improvement by CoT have caused much excitement and have opened up further research on reasoning with LLMs. In the original paper the prompts were handwritten by the researchers for the individual 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 [193], and evaluated.

Kojima et al. [75] 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 3 (third column) compares the approaches. Experiments further show that with this addition to the prompt significant performance gains are also achieved on other reasoning benchmarks, including arithmetic, symbolic, and logical reasoning (achieving 70% accuracy on GSM8K/PaLM when Self-consistency is also included).

The CoT idea itself is inspired by earlier work where natural language steps are generated for arithmetic reasoning [31, 88], and the use of formal languages for reasoning [3, 20, 23, 127].

3.1.2 Prompt Using External Knowledge. CoT prompts are written manually, by the researchers, an approach that does not scale. We can use external information about the problem to improve the prompt. Press et al. [119] 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 QA performance improves faster than the multi-hop

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

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 CoT, but with the *follow up:* prompt), that the model then answers. 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 CoT, 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 [193]. This approach builds on the observation by Kojima et al. [75] that LLMs 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 CoT prompts, on 10 benchmarks, using GPT-3 (see Table 2).

Fu et al. [44] introduce Complexity-based prompting. Inspired by CoT 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 improves 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) [176, 179]. 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. [138] and Li et al. [87] train an external verifier to check results. In contrast, Weng et al. [176] 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 CoT 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

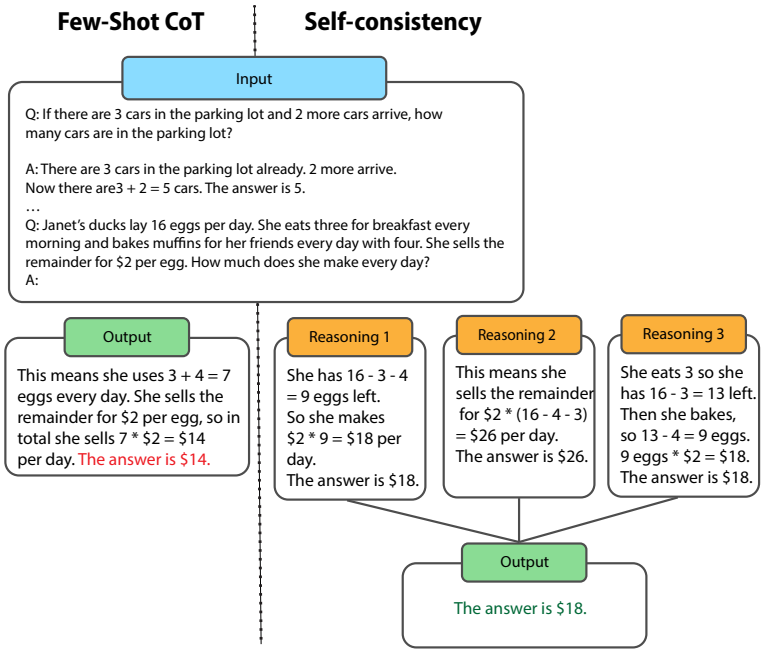

Fig. 4. Self-Consistency (Adapted from [170]), showing majority voting over answers that the model produces.

et al. [196].) Experiments are reported on GPT-3 [18] and on Instruct-GPT [107]. The accuracy of CoT 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 [170]. Self-consistency is a straightforward ensemble approach (a well-known machine learning technique to make a strong learner out of multiple weaker learners [13, 130]). 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 4 summarizes the approach. Self-consistency has been evaluated on arithmetic reasoning, commonsense reasoning and symbolic reasoning, on a variety of LLMs, including GPT-3 [16, 25, 151, 153]. Self-consistency further improves the performance of CoT by 10–20 percentage points, and has been used as a baseline in many of the other approaches in this survey.

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.[3] 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.

[3]In addition to the languages discussed in this survey, there is a long tradition in AI of using formal reasoning based on logic. The interest in CoT has also stimulated research into external symbolic logic solvers that are called by the LLM [51, 109, 132, 145, 148, 183, 197], or where LLMs transform the problem in a logic problem. The use of LLMs for generating formal languages is further discussed in Section 4.4.

LLMs have been quite successful in generating computer code from natural language prompts. Chen et al. [18] 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 often able to generate correct programs from descriptions in natural language, such as commentary strings.

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 [21]. It follows the same steps of code generation, code execution, and code explanation. 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 by 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. [126] 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 [158]. 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 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* suite [52] (BBOB) by 11 percentage 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 [90], 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 Math-Prompter [64]. 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% → 92.5%), evaluated on GPT-3 175B.

Two other approaches that use a formal language are Program-of-thought [19] and Program-aided-language [45]. 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 3 (fourth column) illustrates the Program-aided-language approach. When the evaluation of the reasoning steps is off-loaded 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 [88].) Gao et al. [45] 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).

3.2.3 External Model Validation. We have seen many examples of successful 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 [113] 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. The approach achieves 78% accuracy on GSM8K and also works well on related problems.

Welleck et al. [174] 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 [59]. Here, too, the base model data is augmented by LLM-generated rationales, and then finetuned. The authors achieve 82% accuracy on GSM8K and similarly high scores on QA 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 [191], 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** (**NLI**). 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 is finetuned on the combined dataset. 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 [2]. 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 [2] learns a value function [68] of the behavior of a robot and its environment using temporal difference RL [146]. 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 RL, 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.

Where Say-can learns affordances as a separate function, Inner-monologue [61] formulates robotic planning directly as part of the language prompt, internally. The input consists of many elements: textual descriptions from InstructGPT [16] 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 [142]. 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 [139, 140].

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 (*RL*) search which generates multiple options for the steps, traversing a search tree with backtracking, controlling an exponential search space [182].

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 CoT, Auto-CoT, and Zero-shot CoT. Inner Monologue and Say-Can also use greedy reasoning.

In Least-to-most prompting [198], 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.

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 [170] and Self-verification [176] (in Section 3.2.1) are popular ensemble approaches to evaluate the results of reasoning steps, in which greedy single-path decoding used in CoT 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 [180]. Program-aided-language 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 RL to do so [68, 117, 147]. 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.

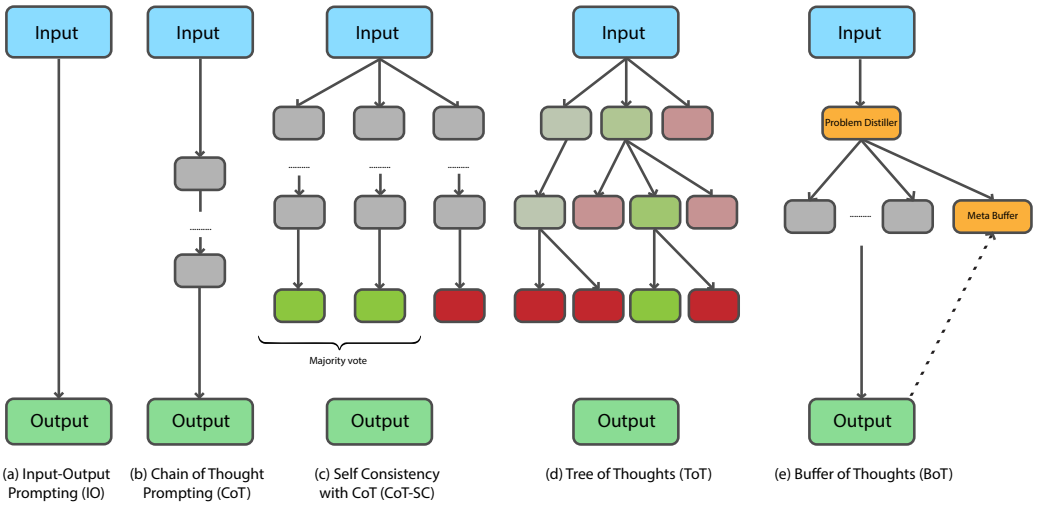

Fig. 5. Reasoning structure of Simple prompting (a), CoT (b), Self-Consistency (c), Tree-of-Thoughts (d) and Buffer of Thoughts (e); Prompts in (a) consist of an instruction, and, perhaps, few-shot examples of answers, providing no reasoning guidance; CoT (b) guides the model with few-shot examples of reasoning steps; Self-Consistency (c) takes the majority vote of existing answers in the model; Tree-of-thoughts (d) uses an external prompt-optimizing algorithm to guide the model to perform a tree search over reasoning steps, exploring multiple alternatives; Buffer-of-thoughts (e) uses a problem distiller that stores high-level thoughts distilled from different problems in a meta-buffer, which is actively managed for capacity.

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 RL. A complex reasoning space can be traversed with a search algorithm. Tree-of-thoughts [187] uses breadth-first or depth-first search to dynamically follow different reasoning steps. Tree-of-thoughts calls the LLM with two different types of prompt: to generate the sub-problems, and to evaluate them. Together, the trio of generation (LLM-prompt-1), evaluation (LLM-prompt-2), and control (search-algorithm) allow systematic exploration of the space of reasoning steps with look-ahead and backtracking. The authors compare their approach to CoT and Self-consistency on the Game of 24, Creative writing, and Mini crossword, achieving an accuracy of 74% on a Game of 24 task. The other tasks are evaluated qualitatively. Figure 5 illustrates the different reasoning structures.[4]

Another approach, Buffer-of-thoughts [185], goes a step toward metareasoning [46]. It introduces a meta-buffer that stores high-level *thought-templates*. These thought-templates are derived from a variety of tasks by a problem distiller. They are then instantiated for specific tasks. To address size restrictions of the LLM context window, the meta-buffer is managed for capacity and thoughts can be combined. Figure 5 compares the Buffer-of-thoughts approach to other approaches such as CoT

[4]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.

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 4.3.3.

A related search method is Beam-search-for-reasoning [181]. 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 (decode) reasoning steps. Beam search uses Program-aided-language models for math word problems [45]. Using a Codex backbone [18], it surpasses the few-shot baselines by 6.34%, 9.56%, and 5.46% on the GSM8K, AQuA, and StrategyQA benchmarks, respectively.

RL is another step in the sophistication of optimization algorithms. It learns by interactive sampling, improving its policy based on rewards from the environment [147]. To use RL, the reasoning problem is formulated as a Markov Decision Process: the agent-algorithm creates a prompt for a next step (an *action*), to sample a step (*t*) and get an answer (*state*, *reward*) from the environment-model. 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 RL also do so in the form of an external algorithm.

Progressive-hint-prompting (**PHP**) uses RL to interactively improve prompts [195]. PHP calls the LLM with dynamic prompts, using previously generated answers as hints, to progressively prompt the LLM toward 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 RL algorithm. The authors have combined PHP with CoT 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 QA (79%).

Another approach that is motivated by improving answers from feedback is Self-refine [95]. Like PHP, the LLM generates an initial output and provides feedback for its answer, using the LLM to refine itself, iteratively. Self-refine prompts the LLM in three ways: (1) for initial generation, (2) for feedback, 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 [4], 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 RL and LLMs is ReAct [188]. However, ReAct does so in a different way. 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 policies.) ReAct uses CoT 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. On two interactive decision making benchmarks (Alfworld and WebShop), ReAct outperforms imitation and RL methods by an absolute success rate of 34% and 10% respectively.

The ReAct work has been developed further. Reflexion [141] creates AI agents that learn by reflecting on failure 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 6). For the actor,

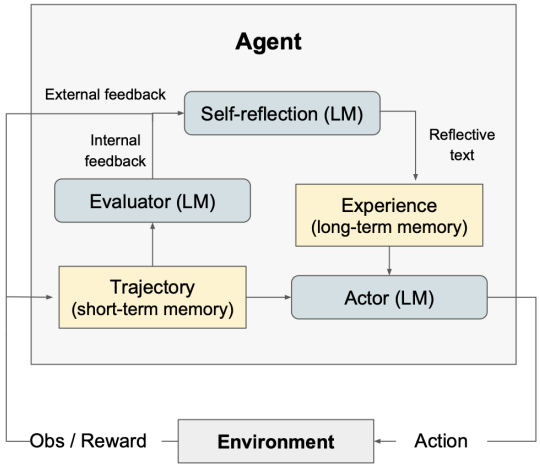

Fig. 6. Architecture of Reflexion [141], showing a close resemblance to the agent/environment structure of RL [147].

CoT and ReAct can be used. Reflexion is evaluated on decision-making, reasoning, and coding tasks. Improvements of 10–20 percentage points are reported.

The RL approaches that we discussed so far—React, Self-refine, Tree-of-thoughts, Buffer-of-thoughts, Reflexion—use external algorithms to manage state for the prompt improvement loop. A more elegant solution would be to perform RL fully in-context, within-prompt. Indeed, Krishnamurthy et al. [78] explicitly asked the question whether LLMs can explore in-context. This is the goal of the **Algorithm-of-thoughts** (**AoT**) approach [135]. Following work by Demircan et al. [35], Lee et al. [82], Wang et al. [169] that showed that transformer architectures can be pretrained and finetuned to perform ICRL, AoT aims to do so in general LLMs such as GPT-4, Claude and Gemini 1.5. They achieve results comparable to Tree-of thoughts on GSM8K, StrategyQA, and Crosswords. Achieving these results with in-context RL is a promising result for in-context learning. Other search-like in-context algorithms are studied by Schultz et al. [133] and Kempinski et al. [72]. Further works in games find that LLMs struggle with the difference between generating an algorithm for a problem, and executing that algorithm correctly (the *knowing-doing gap* [108, 129]), and struggle with the difference between using and mentioning game concepts [30, 159]. Work on implicit reasoning is ongoing, Li et al. [84] provide a survey.

To conclude this overview of RL approaches, we discuss an application in the games domain. Voyager [166] 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 and a skill library to maximize exploration. The goal is to discover diverse items in Minecraft, a form of novelty search [39]. Voyager performs well, reaching high scores by acquiring many tools (see Figure 7) 15 times faster than the baseline.

The applications of RL in LLM reasoning are many, and the connections run deep [120]. Wang et al. [168] use the similarity between RL timesteps and LLM reasoning steps to jointly train a value function together with the LLM policy by optimizing the Bellman equation, achieving 85% accuracy on GSM8K and 81% on Alfworld. Du et al. [38], Guo et al. [50] replace supervised finetuning by

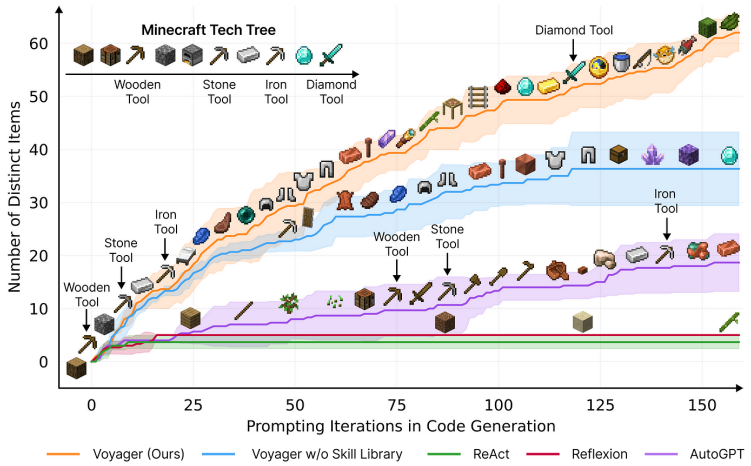

Fig. 7. Performance of Voyager in Minecraft [166]; Voyager performs well, reaching high scores by acquiring many tools.

RL, 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, to discuss limitations, and to look for promising areas of future work. First we discuss how to choose an approach for an application. Next, we discuss issues concerning hallucination, faithful reasoning, and scaling. Then we discuss what LLMs can and cannot do. We highlight connections with sequential decision processes, metacognition, and programming languages, and we end with a research agenda.

4.1 Matching Methods and Applications

This survey has discussed different approaches, applications, and benchmarks. It is often difficult to directly compare benchmark results between individual papers, due to differences in measurement setup. For example, Kamoi et al. [70] critically compare self-reflection studies. Despite the challenges in comparing benchmark results, we can provide general guidance on what approach to use for different types of applications.

For *simple reasoning situations*, that require a single sequence of reasoning steps, CoT is typically used. When the error rate becomes too high, verification methods such as Self-consistency are popular. Indeed, this combination is used in many modern reasoning LLMs such as DeepSeek and GPT-o1 [38, 50, 62, 79].

For *combinatorial puzzles and games* that would traditionally require backtracking, tree-search or RL solution methods must keep track of the enumeration state. However, an LLM without an external algorithm often fails to keep track of this state correctly between calls to the model. An approach such as Tree-of-thoughts, or ReAct, should be considered. Here, the state is managed externally, and the LLM is called for application-dependent functional processing. Various ICRL methods are being developed where state is managed in-context, as in, for example, AoT. When performance is not sufficient, finetuning or pretraining for the domain at hand can be used.

For *complex problems*, where a sequence of steps, or external control, fail to perform well, other reasoning approaches are needed. The LLM can be asked to generate a program for the

problem. In this approach, the LLM acts as a coding agent for its user. The LLM generates a program or a sequence of commands, which are then executed by an external tool, such as an interpreter, a robot, a planner, or a logic solver. In robotics, vision-language-action models have achieved impressive results [73], but also strong results in logistics [12], finance and medicine are reported [118].

4.2 Hallucination, Faithfulness, and Scaling

Reasoning by LLMs is not error-free, and many studies aim to provide deeper insight into the reasoning processes in language models. Saparov and He [132] 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 CoT struggles with proof trees with a wide branching factor. In another study, Wang et al. [165] aim to increase our understanding of how CoT works. The authors find that the order of the reasoning steps is important. Prompts should be relevant to the question, and coherent (steps should be in the correct order). Jin et al. [66] also study the impact of reasoning step length on LLMs, again finding a strong positive correlation between the length of the prompt and reasoning abilities. Next, we discuss works on errors in the CoT approach, studying whether the reasoning of the LLM is faithful, or that it gives the right answer for the wrong reason.

4.2.1 Faithfulness. CoT 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 with LLMs. Lanham et al. [80] note 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 CoT, 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, (CoT) reasoning suffers from hallucination [60].

Another study adds deliberate bias to the prompt. For example, in a multiple-choice setting, they always make answer (A) the correct answer [155]. They find that a bias toward wrong answers can cause significant drops in accuracy, and that models frequently generate Chain-of-though explanations rationalizing wrong answers. To address issues of faithfulness, Lyu et al. [94], Xu et al. [183] 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, 22, 124].

Faithfulness studies tell us more about how models reason. Further surveys on this topic are Chuang et al. [29], Luo et al. [93], Mondorf and Plank [102], Paul et al. [114],

4.2.2 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 [54, 55, 71]. Distillation of reasoning to smaller models can work surprisingly well in situations with more explicit instructions, and 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. [49], Xu et al. [184]. For reasoning specifically, Magister et al. [96] have studied reasoning in small language models, using a student model that learns from a teacher model,

by finetuning. Other works focus on prompt distillation for retrieval [34], recommendation [85], embodied agents [24], and LLM graph reasoning [192].

Another study related to Self-taught-reasoner focuses on explanation in small language models, also achieving good results of knowledge transfer [86].

4.3 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, such as in 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.

4.3.1 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 gaming, 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, and 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 as scaffolding (such as search or RL), dynamic prompts can use the LLMs to perform complex and dynamic reasoning patterns. Note that the reasoning control is now outside the core LLM: an external control algorithm, on top of in-context-learning, dynamically generating prompts for the LLM.

At this point, it is interesting to note the confluence of two schools of thought in artificial intelligence: symbolic and connectionist.[5] Search and RL are rooted in the symbolic tradition, while LLMs and deep learning are rooted in the connectionist tradition. The literature in this survey combines the two traditions. Higher 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 [17, 67] and retrosynthesis of molecules [134]). 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.

4.3.2 What Can LLMs Not Do? Now that grade school math word problems are largely solvable, harder reasoning benchmarks in other domains are appearing [1]. Most of the reasoning capabilities exhibited by LLMs are due to the 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 reasoning that we have seen in this survey. Models can be instructed to write their own reasoning prompts; however, such Auto-GPT or Auto-CoT prompts are prone to feedback-loop problems, and need careful evaluation, verification, and grounding in the real world, to prevent

[5]Reasoning and planning have been studied since the start of artificial intelligence, starting with logic and reasoning [105], search algorithms in puzzles and board games [77, 115, 116], robot planning [41], classical machine learning such as decision trees and support vector machines [13, 33, 43], through knowledge representation and the semantic web [157]. Ever since the success of the connectionist approach [15, 48, 81] (deep learning, including LLMs) researchers have tried to join the two approaches [144, 152, 167].

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 [118]. Some of what we experience as reasoning *by* the LLM is controlled by the prompt or the scaffolding algorithm. Studies into ICRL aim to answer the question if prompt learning is able get the LLM to create a prompt to exhibit dynamic reasoning by itself [35, 82, 108, 133].

Some studies on symbolic planning are critical on the abilities of LLMs [156], and show examples of planning failures, arguing that LLMs are better used to improve heuristic elements of traditional planners, such as PDDL [69], to strengthen traditional symbolic planning approaches.

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 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, 22, 124].

Some of the names of the approaches surveyed in this article 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.

4.3.3 Reasoning toward Metacognition. Human thought exhibits the ability to reason about self, we are able to think about our own thinking processes. Metacognition studies this phenomenon [161]. Prompted by the success of CoT and related works, metacognition has also been studied in the context of LLMs [154].

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 [171] study Metacognitive-prompting. Inspired by CoT and Self-consistency, they create manually designed prompts to increase the understanding of language models. Another work, again inspired by CoT and Self-consistency, connects psychology and LLMs. Didolkar et al. [36] study metacognitive capabilities of LLMs in mathematical problem solving, both on GSM8K and on the harder MATH problems [53]. 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 CoT-like prompts, more problems are solved than with standard CoT/Self-consistency/Program-aided-language 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 weaker model benefits from the skill-name-enhanced prompts. Begus et al. [5] propose a program for the study of metalinguistic abilities of LLMs. LLMs struggle with analogies [83, 100] and analogical reasoning. Metacognitive reasoning with LLMs is still in its early stages.

4.4 LLMs that Generate Computer Code

A persistent thread in the results in this survey is that LLMs are good at generating formal languages, such as PDDL, logic, math equations, and Python. The agentic approach in which LLMs are combined with external tools such as interpreters or debuggers often yields good performance [118]. LLMs can be used in different modes, which we will now discuss.

When LLMs are used in *direct* mode, a prompt is provided with an instruction, and an answer. This is the mode in which most casual users use an LLM.

LLMs can also be used in *algorithmic* mode, or CoT mode, where an extra element is added to the prompt that gives an example of which steps to take to arrive at the answer [135]. Muennighoff et al. [103] show how such test-time scaling can tradeoff model size and training time. Lee et al. [82], Schultz et al. [133], Wang et al. [169] show that the transformer architecture can be trained such that search and RL algorithms can be executed in-context.

LLMs do not preserve state between calls, and in order to allow for a natural dialogue to occur, chatbots such as ChatGPT insert the most recent history of preceding calls into each prompt before the model is called [16]. For harder strategic reasoning tasks more advanced state management solutions are necessary, that call the LLM in an explicit *external* mode [187, 188]. Approaches such as Tree-of-thoughts use the LLM in a stateless functional way, with the prompt containing the instructions, not unlike a natural language variant of a functional programming language such as LISP or Haskell.

Finally, an LLM can be used in *code generation* mode. The prompt specifies a problem, and the LLM is asked to generate computer code (such as Python or PDDL), to circumvent the knowing-doing gap [108]. The code may contain variables that manage the state correctly. The code can be executed directly, or first be checked and improved by a programmer. We expect that the use LLMs as code generators, and prompt-engineering as natural-language-programming, will continue to grow.

4.5 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 [14] and metacognition. First we discuss current 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.

Specific research topics for reasoning with LLMs are:

- *In Context Reinforcement Learning*—Search control beyond greedy search is often implemented as an external RL algorithm. Is it possible to incorporate the control stage of the reasoning pipeline into one static prompt, for implicit reasoning or ICRL? Studies indicate that models can be trained for decision making [35, 82], but general LLMs struggle to perform ICRL well [108, 129, 133, 159].
- *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. Retrieval augmented generative methods (RAG) can help LLMs to actively find new information when the reasoning outcome is uncertain. RAG, search, and agentic LLMs, are also active areas of research [34, 118, 162].

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 implicit reasoning [84] and the working of multi-step in-context learning in LLMs is dearly needed.
- *Smaller language models*—Efficiency is an important element for wide adoption of language models. Smaller models have many advantages over larger models: a smaller environmental footprint, more accessible to people with fewer computational resources, and the possibility to finetune them. Models may have fewer parameters, or the parameters may be quantized with fewer bits. Unfortunately, with a smaller number of parameters, the models also have less reasoning power. It is important for future work to investigate this tradeoff.

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 reasoning prompts based on the symbolic AI literature, and how can LLMs help ground symbolic reasoning in language?
- *Multimodal World-Models with Embodied Grounding*—How can an LLM maintain a unified, continually updated representation that integrates text, vision, audio, and sensorimotor feedback, enabling it to reason over events across modalities, and refine its world-model through closed perception–action loops in real-world environments?
- *Norm-Sensitive and Value-Aligned Reasoning*—How do we represent, adapt, and audit diverse cultural norms and ethical values within the reasoning process, ensuring that each step of an LLM's CoT respects context-dependent moral constraints while remaining transparent and verifiable?
- *Reasoning in other languages than English*—The majority of the research into LLM reasoning is for English. Although there are efforts in creating benchmarks to test LLM capabilities in other languages,[6] these contain mainly NLP tasks such as QA and NLI. In addition, challenges related to low-resource languages (languages for which very limited training data is available) have not been addressed yet.
- *Reasoning in specialized domains*—In an effort to guide development and evaluation of new methods, research in AI has a strong focus on benchmarking: standardized datasets with a limited set of problems that are realistic to evaluate. In the real world, however, reasoning problems also occur in more challenging contexts. A few examples are: legal reasoning for the interpretation of case law or writing contracts; scientific reasoning for advancing scientific fields; and medical reasoning for AI-assisted diagnostics.

5 Conclusion

When LLMs of sufficient size are prompted with examples, they can perform few-shot learning in-context, providing an answer at inference time, without retraining model parameters. Although they achieve good performance on language tasks, simple prompting methods do not perform well on reasoning tasks—tasks that humans typically solve in a step by step fashion. CoT is an in-context prompting method to guide an LLM to "think step by step." CoT was originally developed for solving grade school math word problems. GSM8K, the most popular reasoning benchmarks

[6] https://github.com/NaiveNeuron/awesome-multilingual-llm-benchmarks

in this survey, contains 8500 grade school math word problems. With older LLMs such as GPT-3, reasoning approaches show an improvement of 20–50% points over standard prompting methods. This success has spawned many new reasoning approaches. A potential problem in the CoT method is that errors may accumulate over multiple reasoning steps. Prompting the LLM to reformulate problems in Python code (or another formal language) can successfully reduce the error rate. The success of reasoning methods has attracted more applications, and with them, benchmarks are diverging. 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.

Many current LLMs have adopted CoT approaches. In this survey we categorize the approaches on how they generate, evaluate, and control the reasoning steps. Inference-time reasoning methods are also used for LLM finetuning. **Reinforcement learning with verifiable rewards** (**RLVR**) and **group relative policy optimization** (**GRPO**) use inference-time reasoning results to augment finetuning for reasoning tasks.

For complex tasks the number of reasoning steps that is generated may be large. The size of the multi-step reasoning space can be controlled dynamically by external search or RL algorithms, often in the form of a wrapper of Python code that generates LLM-prompts. Many of the names of the approaches in this survey suggest a link to metacognition (*Reflexion, Self-refine, Self-improvement, Inner-monologue*)—the act of thinking about one's own thought process. The first preliminary experiments of language models that reason about their reasoning skills have appeared.

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 using an incorrect reasoning chain). LLMs hallucinate and suffer from bias, and their use poses ethical and societal dangers. Self-verification methods have been developed to reduce error-accumulation, and retrieval augmentation methods (RAG) ground LLM output directly in sources such as Wikipedia. However, ethical and societal dangers remain, especially since also reflection is not error-free.

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.

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. Among the items mentioned are RL for finetuning and self-reflection, LLM agents that generate code for external tools, multimodal world models, value-aligned reasoning, and small language models.

Acknowledgments

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

References

[1] Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. 2024. Large language models for mathematical reasoning: Progresses and challenges. arXiv:2402.00157. Retrieved from https://arxiv.org/abs/2402.00157

[2] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. 2022. Do as I can, not as I say: Grounding language in robotic affordances. arXiv:2204.01691. Retrieved from https://arxiv.org/abs/2204.01691

[3] Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. 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. *arXiv preprint arXiv:1905.13319*.

[4] Arian Askari, Roxana Petcu, Chuan Meng, Mohammad Aliannejadi, Amin Abolghasemi, Evangelos Kanoulas, and Suzan Verberne. 2024. SOLID: Self-seeding and multi-intent self-instructing LLMs for generating intent-aware information-seeking dialogs. arXiv:2402.11633. Retrieved from https://arxiv.org/abs/2402.11633

[5] Gasper Begus, Maksymilian Dabkowski, and Ryan Rhodes. 2025. Large linguistic models: Investigating LLMs' metalinguistic abilities. *IEEE Transactions on Artificial Intelligence* (2025).

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

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

[8] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*. 41–48.

[9] Leonard Bereska and Efstratios Gavves. 2024. Mechanistic interpretability for AI safety–A review. *Transactions on Machine Learning Research, arXiv preprint arXiv:2404.14082* (2024).

[10] Lukas Berglund, Meg Tong, Max Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. 2024. The reversal curse: LLMs trained on A is B fail to learn B is A. arXiv:2309.12288. Retrieved from https://arxiv.org/abs/2309.12288

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

[12] Bernd Bohnet, Azade Nova, Aaron T. Parisi, Kevin Swersky, Katayoon Goshvadi, Hanjun Dai, Dale Schuurmans, Noah Fiedel, and Hanie Sedghi. 2024. Exploring and benchmarking the planning capabilities of large language models. arXiv:2406.13094. Retrieved from https://arxiv.org/abs/2406.13094

[13] Leo Breiman. 2001. Random forests. *Machine Learning* 45 (2001), 5–32.

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

[15] Rodney A. Brooks. 1990. Elephants don't play chess. *Robotics and Autonomous Systems* 6, 1–2 (1990), 3–15.

[16] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems* 33 (2020), 1877–1901.

[17] Patrick Bryant, Gabriele Pozzati, and Arne Elofsson. 2022. Improved prediction of protein-protein interactions using AlphaFold2. *Nature Communications* 13, 1 (2022), 1265.

[18] 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. 2021. Evaluating large language models trained on code. arXiv:2107.03374. Retrieved from https://arxiv.org/abs/2107.03374

[19] Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions of Machine Learning Research*. arXiv:2211.12588. Retrieved from https://arxiv.org/abs/2211.12588

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

[21] Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. 2024. Teaching large language models to self-debug. In *Proceedings of the 12th International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=KuPixIqPiq

[22] Xi Chen, Aske Plaat, and Niki van Stein. 2025. How does chain of thought think? Mechanistic interpretability of chain-of-thought reasoning with sparse autoencoding. arXiv:2507.22928. Retrieved from https://arxiv.org/abs/2507.22928

[23] Ting-Rui Chiang and Yun-Nung Chen. 2019. Semantically-aligned equation generation for solving and reasoning math word problems. arXiv:1811.00720. Retrieved from https://arxiv.org/abs/1811.00720

[24] Wonje Choi, Woo Kyung Kim, Minjong Yoo, and Honguk Woo. 2024. Embodied CoT distillation from llm to off-the-shelf agents. In *Forty-first International Conference on Machine Learning*. 8702–8721.

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

[26] Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V. Le, Sergey Levine, and Yi Ma. 2025. SFT memorizes, RL generalizes: A comparative study of foundation model post-training. arXiv:2501.17161. Retrieved from https://arxiv.org/abs/2501.17161

[27] Zheng Chu, Jingchang Chen, Qianglong Chen, Weijiang Yu, Tao He, Haotian Wang, Weihua Peng, Ming Liu, Bing Qin, and Ting Liu. 2024. 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)*. 1173–1203.

[28] Zheng Chu, Jingchang Chen, Qianglong Chen, Weijiang Yu, Tao He, Haotian Wang, Weihua Peng, Ming Liu, Bing Qin, and Ting Liu. 2024. A survey of chain of thought reasoning: Advances, frontiers and futurearXiv:2309.15402. Retrieved from https://arxiv.org/abs/2309.15402

[29] Yu-Neng Chuang, Guanchu Wang, Chia-Yuan Chang, Ruixiang Tang, Fan Yang, Mengnan Du, Xuanting Cai, and Xia Hu. 2024. FaithLM: Towards faithful explanations for large language models. arXiv:2402.04678. Retrieved from https://arxiv.org/abs/2402.04678

[30] Nathan Cloos, Meagan Jens, Michelangelo Naim, Yen-Ling Kuo, Ignacio Cases, Andrei Barbu, and Christopher J. Cueva. 2024. Baba is AI: Break the rules to beat the benchmark. arXiv:2407.13729. Retrieved from https://arxiv.org/abs/2407.13729

[31] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. arXiv:2110.14168. Retrieved from https://arxiv.org/abs/2110.14168

[32] MAA Codeforces. 2024. American Invitational Mathematics Examination-AIME 2024. https://maa.org

[33] Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine Learning* 20 (1995), 273–297.

[34] Zhuyun Dai, Vincent Y. Zhao, Ji Ma, Yi Luan, Jianmo Ni, Jing Lu, Anton Bakalov, Kelvin Guu, Keith B. Hall, and Ming-Wei Chang. 2023. Promptagator: Few-shot dense retrieval from 8 examples. arXiv:2209.11755. Retrieved from https://arxiv.org/abs/2209.11755

[35] Can Demircan, Tankred Saanum, Akshay K. Jagadish, Marcel Binz, and Eric Schulz. 2024. Sparse autoencoders reveal temporal difference learning in large language models. arXiv:2410.01280. Retrieved from https://arxiv.org/abs/2410.01280

[36] Aniket Didolkar, Anirudh Goyal, Nan Rosemary Ke, Siyuan Guo, Michal Valko, Timothy Lillicrap, Danilo Rezende, Yoshua Bengio, Michael Mozer, and Sanjeev Arora. 2024. Metacognitive capabilities of LLMs: An exploration in mathematical problem solving. arXiv:2405.12205. Retrieved from https://arxiv.org/abs/2405.12205

[37] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. 2023. A survey on in-context learning. arXiv:2301.00234. Retrieved from https://arxiv.org/abs/2301.00234

[38] Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. 2025. Kimi k1.5: Scaling reinforcement learning with LLMs. arXiv:2501.12599. Retrieved from https://arxiv.org/abs/2501.12599

[39] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. 2019. Diversity is all you need: Learning skills without a reward function. arXiv:1802.06070. Retrieved from https://arxiv.org/abs/1802.06070

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

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

[42] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the International Conference on Machine Learning*. PMLR, 1126–1135.

[43] Peter Flach. 2012. *Machine Learning: The Art and Science of Algorithms that Make Sense of Data*. Cambridge University Press.

[44] Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. 2022. Complexity-based prompting for multi-step reasoning. In *Proceedings of the 11th International Conference on Learning Representations*.

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

[46] Peizhong Gao, Ao Xie, Shaoguang Mao, Wenshan Wu, Yan Xia, Haipeng Mi, and Furu Wei. 2024. Meta reasoning for large language models. arXiv:2406.11698. Retrieved from https://arxiv.org/abs/2406.11698

[47] Louie Giray. 2023. Prompt engineering with ChatGPT: A guide for academic writers. *Annals of Biomedical Engineering* 51, 12 (2023), 2629–2633.

[48] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.

[49] Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2023. MiniLLM: Knowledge distillation of large language models. In *Proceedings of the 12th International Conference on Learning Representations*.

[50] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. arXiv:2501.12948. Retrieved from https://arxiv.org/abs/2501.12948

[51] Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, et al. 2022. Folio: Natural language reasoning with first-order logic. arXiv:2209.00840. Retrieved from https://arxiv.org/abs/2209.00840

[52] Nikolaus Hansen and Raymond Ros. 2010. 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*. 1519–1526.

[53] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. arXiv:2103.03874. Retrieved from https://arxiv.org/abs/2103.03874

[54] Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B. Brown, Prafulla Dhariwal, Scott Gray, et al. 2020. Scaling laws for autoregressive generative modeling. arXiv:2010.14701. Retrieved from https://arxiv.org/abs/2010.14701

[55] 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. 2022. Training compute-optimal large language models. arXiv:2203.15556. Retrieved from https://arxiv.org/abs/2203.15556

[56] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. 2021. Meta-learning in neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 9 (2021), 5149–5169.

[57] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. arXiv:2106.09685. Retrieved from https://arxiv.org/abs/2106.09685

[58] Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards reasoning in large language models: A survey. . arXiv:2212.10403. Retrieved from https://arxiv.org/abs/2212.10403

[59] Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2023. Large language models can self-improve. arXiv:2210.11610. Retrieved from https://arxiv.org/abs/2210.11610

[60] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, et al. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transaction on Information Systems* 43, 2 (2025), 1–55.

[61] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. 2022. Inner monologue: Embodied reasoning through planning with language models. arXiv:2207.05608. Retrieved from https://arxiv.org/abs/2207.05608

[62] Zhen Huang, Haoyang Zou, Xuefeng Li, Yixiu Liu, Yuxiang Zheng, Ethan Chern, Shijie Xia, Yiwei Qin, Weizhe Yuan, and Pengfei Liu. 2024. O1 replication journey–part 2: Surpassing O1-preview through simple distillation, big progress or bitter lesson? arXiv:2411.16489. Retrieved from https://arxiv.org/abs/2411.16489

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

[64] Shima Imani, L. Du, and H. Shrivastava. 2023. Mathprompter: Mathematical reasoning using large language models. arXiv:2303.05398. Retrieved from https://arxiv.org/abs/2303.05398

[65] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. 2018. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2704–2713.

[66] Mingyu Jin, Qinkai Yu, Dong Shu, Haiyan Zhao,Wenyue Hua, Yanda Meng, Yongfeng Zhang, and Mengnan Du. 2024. The impact of reasoning step length on large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, 1830–1842. https://aclanthology.org/2024.findings-acl.108/

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

[68] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. 1996. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research* 4 (1996), 237–285.

[69] Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Kaya Stechly, Mudit Verma, Siddhant Bhambri, Lucas Saldyt, and Anil Murthy. 2024. LLMs can't plan, but can help planning in LLM-modulo frameworks. arXiv:2402.01817. Retrieved from https://arxiv.org/abs/2402.01817

[70] Ryo Kamoi, Yusen Zhang, Nan Zhang, Jiawei Han, and Rui Zhang. 2024. When can LLMs actually correct their own mistakes? A critical survey of self-correction of LLMs. *Transactions of the Association for Computational Linguistics* 12 (2024), 1417–1440.

[71] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. arXiv:2001.08361. Retrieved from https://arxiv.org/abs/2001.08361

[72] Benjamin Kempinski, Ian Gemp, Kate Larson, Marc Lanctot, Yoram Bachrach, and Tal Kachman. 2025. Game of thoughts: Iterative reasoning in game-theoretic domains with large language models. In *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems*. 1088–1097.

[73] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. 2024. Openvla: An open-source vision-language-action model. arXiv:2406.09246. Retrieved from https://arxiv.org/abs/2406.09246

[74] Tom Kocmi, Rachel Bawden, Ondřej Bojar, Anton Dvorkovich, Christian Federmann, Mark Fishel, Thamme Gowda, Yvette Graham, Roman Grundkiewicz, Barry Haddow, et al. 2022. Findings of the 2022 conference on machine translation (WMT22). In *Proceedings of the 7th Conference on Machine Translation (WMT)*. 1–45.

[75] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in Neural Information Processing Systems* 35 (2022), 22199–22213.

[76] Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. MAWPS: A math word problem repository. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 1152–1157.

[77] Richard E. Korf. 1999. Artificial intelligence search algorithms. In *Algorithms and Theory of Computation Handbook, 22-17*. Citeseer.

[78] Akshay Krishnamurthy, Keegan Harris, Dylan J. Foster, Cyril Zhang, and Aleksandrs Slivkins. 2024. Can large language models explore in-context? arXiv:2403.15371. Retrieved from https://arxiv.org/abs/2403.15371

[79] Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. 2024. Tulu 3: Pushing frontiers in open language model post-training. arXiv:2411.15124. Retrieved from https://arxiv.org/abs/2411.15124

[80] Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, et al. 2023. Measuring faithfulness in chain-of-thought reasoning. arXiv:2307.13702. Retrieved from https://arxiv.org/abs/2307.13702

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

[82] Jonathan Lee, Annie Xie, Aldo Pacchiano, Yash Chandak, Chelsea Finn, Ofir Nachum, and Emma Brunskill. 2023. Supervised pretraining can learn in-context reinforcement learning. *Advances in Neural Information Processing Systems* 36 (2023), 43057–43083.

[83] Martha Lewis and Melanie Mitchell. 2024. Evaluating the robustness of analogical reasoning in large language models. arXiv:2411.14215. Retrieved from https://arxiv.org/abs/2411.14215

[84] Jindong Li, Yali Fu, Li Fan, Jiahong Liu, Yao Shu, Chengwei Qin, Menglin Yang, Irwin King, and Rex Ying. 2025. Implicit reasoning in large language models: A comprehensive survey. arXiv:2509.02350. Retrieved from https://arxiv.org/abs/2509.02350

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

[86] Shiyang Li, Jianshu Chen, Yelong Shen, Zhiyu Chen, Xinlu Zhang, Zekun Li, Hong Wang, Jing Qian, Baolin Peng, Yi Mao, et al. 2024. Explanations from large language models make small reasoners better. arXiv:2210.06726. Retrieved from https://arxiv.org/abs/2210.06726

[87] Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. 2023. Making language models better reasoners with step-aware verifier. arXiv:2206.02336. Retrieved from https://arxiv.org/abs/2206.02336

[88] Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. arXiv:1705.04146. Retrieved from https://arxiv.org/abs/1705.04146

[89] Michael Lederman Littman. 1996. *Algorithms for Sequential Decision-Making*. Brown University.

[90] Fei Liu, Tong Xialiang, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. 2024. Evolution of heuristics: Towards efficient automatic algorithm design using large language model. In *Proceedings of the 41st International Conference on Machine Learning*.

[91] Hanmeng Liu, Ruoxi Ning, Zhiyang Teng, Jian Liu, Qiji Zhou, and Yue Zhang. 2023. Evaluating the logical reasoning ability of ChatGPT and GPT-4. arXiv:2304.03439. Retrieved from https://arxiv.org/abs/2304.03439

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

[93] Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2024. Reasoning on graphs: Faithful and interpretable large language model reasoning. arXiv:2310.01061. Retrieved from https://arxiv.org/abs/2310.01061
[94] Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. 2023. Faithful chain-of-thought reasoning. arXiv:2301.13379. Retrieved from https://arxiv.org/abs/2301.13379
[95] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems* 36 (2023).
[96] Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. 2023. Teaching small language models to reason. arXiv:2212.08410. Retrieved from https://arxiv.org/abs/2212.08410
[97] Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. 2020. A diverse corpus for evaluating and developing english math word problem solvers. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. Retrieved from https://api.semanticscholar.org/CorpusID:220047831
[98] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. 2018. Mixed precision training. arXiv:1710.03740. Retrieved from https://arxiv.org/abs/1710.03740
[99] Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. 2024. Large language models: A survey. arXiv:2402.06196. Retrieved from https://arxiv.org/abs/2402.06196
[100] Melanie Mitchell and Douglas R. Hofstadter. 1990. The emergence of understanding in a computer model of concepts and analogy-making. *Physica D: Nonlinear Phenomena* 42, 1–3 (1990), 322–334.
[101] Amir Moeini, Jiuqi Wang, Jacob Beck, Ethan Blaser, Shimon Whiteson, Rohan Chandra, and Shangtong Zhang. 2025. A survey of in-context reinforcement learning. arXiv:2502.07978. Retrieved from https://arxiv.org/abs/2502.07978
[102] Philipp Mondorf and Barbara Plank. 2024. Beyond accuracy: Evaluating the reasoning behavior of large language models–A survey. arXiv:2404.01869. Retrieved from https://arxiv.org/abs/2404.01869
[103] Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. arXiv:2501.19393. Retrieved from https://arxiv.org/abs/2501.19393
[104] Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. arXiv:1808.08745. Retrieved from https://arxiv.org/abs/1808.08745
[105] Allen Newell and Herbert A. Simon. 1961. Computer simulation of human thinking: A theory of problem solving expressed as a computer program permits simulation of thinking processes. *Science* 134, 3495 (1961), 2011–2017.
[106] Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, et al. 2022. Show your work: Scratchpads for intermediate computation with language models. arXiv:2112.00114. Retrieved from https://arxiv.org/abs/2112.00114
[107] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems* 35 (2022), 27730–27744.
[108] Davide Paglieri, Bartłomiej Cupiał, Samuel Coward, Ulyana Piterbarg, Maciej Wolczyk, Akbir Khan, Eduardo Pignatelli, Łukasz Kuciński, Lerrel Pinto, Rob Fergus, et al. 2024. BALROG: Benchmarking agentic LLM and VLM reasoning on games. arXiv:2411.13543. Retrieved from https://arxiv.org/abs/2411.13543
[109] Liangming Pan, Alon Albalak, Xinyi Wang, and William Yang Wang. 2023. Logic-LM: Empowering large language models with symbolic solvers for faithful logical reasoning. arXiv:2305.12295. Retrieved from https://arxiv.org/abs/2305.12295
[110] Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. arXiv:1606.06031. Retrieved from https://arxiv.org/abs/1606.06031
[111] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. 311–318.
[112] Arkil Patel, S. Bhattamishra, and N. Goyal. 2021. Are NLP models really able to solve simple math word problems?. arXiv:2103.07191. Retrieved from https://arxiv.org/abs/2103.07191
[113] Debjit Paul, Mete Ismayilzada, Maxime Peyrard, Beatriz Borges, Antoine Bosselut, Robert West, and Boi Faltings. 2024. Refiner: Reasoning feedback on intermediate representations. arXiv:2304.01904. Retrieved from https://arxiv.org/abs/2304.01904
[114] Debjit Paul, Robert West, Antoine Bosselut, and Boi Faltings. 2024. Making reasoning matter: Measuring and improving faithfulness of chain-of-thought reasoning. arXiv:2402.13950. Retrieved from https://arxiv.org/abs/2402.13950
[115] Aske Plaat. 1996. *Research, Re: Search & Re-search*. Ph. D. Dissertation. Erasmus University Rotterdam.

[116] Aske Plaat. 2020. *Learning to Play: Reinforcement Learning and Games*. Springer Nature.
[117] Aske Plaat. 2022. *Deep Reinforcement Learning*. Springer, Singapore.
[118] Aske Plaat, Max van Duijn, Niki van Stein, Mike Preuss, Peter van der Putten, and Kees Joost Batenburg. 2025. Agentic large language models, a survey. *Journal of Artificial Intelligence Research, arXiv preprint arXiv:2503.23037* (2025).
[119] Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. arXiv:2210.03350. Retrieved from https://arxiv.org/abs/2210.03350
[120] Moschoula Pternea, Prerna Singh, Abir Chakraborty, Yagna Oruganti, Mirco Milletari, Sayli Bapat, and Kebei Jiang. 2024. The RL/LLM taxonomy tree: Reviewing synergies between reinforcement learning and large language models. *Journal of Artificial Intelligence Research, 80; arXiv preprint arXiv:2402.01874* (2024).
[121] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.
[122] Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. 2021. Scaling language models: Methods, analysis & insights from training gopher. arXiv:2112.11446. Retrieved from https://arxiv.org/abs/2112.11446
[123] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems* 36 (2024).
[124] Daking Rai, Yilun Zhou, Shi Feng, Abulhair Saparov, and Ziyu Yao. 2024. A practical review of mechanistic interpretability for transformer-based language models. arXiv:2407.02646. Retrieved from https://arxiv.org/abs/2407.02646
[125] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. arXiv:1606.05250. Retrieved from https://arxiv.org/abs/1606.05250
[126] Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, et al. 2024. Mathematical discoveries from program search with large language models. *Nature* 625, 7995 (2024), 468–475.
[127] Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. arXiv:1608.01413. Retrieved from https://arxiv.org/abs/1608.01413
[128] Baptiste Roziere, Marie-Anne Lachaux, Lowik Chanussot, and Guillaume Lample. 2020. Unsupervised translation of programming languages. *Advances in Neural Information Processing Systems* 33 (2020), 20601–20611.
[129] Anian Ruoss, Fabio Pardo, Harris Chan, Bonnie Li, Volodymyr Mnih, and Tim Genewein. 2024. LMAct: A benchmark for in-context imitation learning with long multimodal demonstrations. arXiv:2412.01441. Retrieved from https://arxiv.org/abs/2412.01441
[130] Omer Sagi and Lior Rokach. 2018. Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 8, 4 (2018), e1249.
[131] Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. 2024. A systematic survey of prompt engineering in large language models: Techniques and applications. arXiv:2402.07927. Retrieved from https://arxiv.org/abs/2402.07927
[132] Abulhair Saparov and He He. 2023. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. arXiv:2210.01240. Retrieved from https://arxiv.org/abs/2210.01240
[133] John Schultz, Jakub Adamek, Matej Jusup, Marc Lanctot, Michael Kaisers, Sarah Perrin, Daniel Hennes, Jeremy Shar, Cannada Lewis, Anian Ruoss, et al. 2024. Mastering board games by external and internal planning with language models. arXiv:2412.12119. Retrieved from https://arxiv.org/abs/2412.12119
[134] Marwin Segler, M. Preuss, and M. Waller. 2018. Planning chemical syntheses with deep neural networks and symbolic AI. *Nature* 555, 7698 (2018), 604–610.
[135] Bilgehan Sel, Ahmad Al-Tawaha, Vanshaj Khattar, Ruoxi Jia, and Ming Jin. 2023. Algorithm of thoughts: Enhancing exploration of ideas in large language models. arXiv:2308.10379. Retrieved from https://arxiv.org/abs/2308.10379
[136] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. arXiv:1511.06709. Retrieved from https://arxiv.org/abs/1511.06709
[137] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, et al. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. arXiv:2402.03300. Retrieved from https://arxiv.org/abs/2402.03300
[138] Jianhao Shen, Yichun Yin, Lin Li, Lifeng Shang, Xin Jiang, Ming Zhang, and Qun Liu. 2021. Generate & rank: A multi-task framework for math word problems. arXiv:2109.03034. Retrieved from https://arxiv.org/abs/2109.03034
[139] Zhengliang Shi, Shen Gao, Xiuyi Chen, Yue Feng, Lingyong Yan, Haibo Shi, Dawei Yin, Zhumin Chen, Suzan Verberne, and Zhaochun Ren. 2024. Chain of tools: Large language model is an automatic multi-tool learner. arXiv:2405.16533. Retrieved from https://arxiv.org/abs/2405.16533

[140] Zhengliang Shi, Shen Gao, Xiuyi Chen, Lingyong Yan, Haibo Shi, Dawei Yin, Zhumin Chen, Pengjie Ren, Suzan Verberne, and Zhaochun Ren. 2024. Learning to use tools via cooperative and interactive agents. arXiv:2403.03031. Retrieved from https://arxiv.org/abs/2403.03031
[141] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2024. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems* 36 (2024).
[142] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. 2022. Cliport: What and where pathways for robotic manipulation. In *Proceedings of the Conference on Robot Learning*. PMLR, 894–906.
[143] Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2021. ALFWorld: Aligning text and embodied environments for interactive learning. arXiv:2010.03768. Retrieved from https://arxiv.org/abs/2010.03768
[144] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. 2017. Mastering the game of go without human knowledge. *Nature* 550, 7676 (2017), 354–359.
[145] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. arXiv:2206.04615. Retrieved from https://arxiv.org/abs/2206.04615
[146] Richard S. Sutton. 1988. Learning to predict by the methods of temporal differences. *Machine Learning* 3 (1988), 9–44.
[147] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction*. MIT Press.
[148] Oyvind Tafjord, Bhavana Dalvi Mishra, and Peter Clark. 2020. ProofWriter: Generating implications, proofs, and abductive statements over natural language. arXiv:2012.13048. Retrieved from https://arxiv.org/abs/2012.13048
[149] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. arXiv:1811.00937. Retrieved from https://arxiv.org/abs/1811.00937
[150] Yiming Tan, Dehai Min, Yu Li, Wenbo Li, Nan Hu, Yongrui Chen, and Guilin Qi. 2023. Can ChatGPT replace traditional KBQA models? An in-depth analysis of the question answering performance of the GPT LLM family. In *Proceedings of the International Semantic Web Conference*. Springer, 348–367.
[151] Yi Tay, Mostafa Dehghani, Vinh Q. Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Siamak Shakeri, Dara Bahri, Tal Schuster, et al. 2023. Ul2: Unifying language learning paradigms. arXiv:2205.05131. Retrieved from https://arxiv.org/abs/2205.05131
[152] Gerald Tesauro. 1995. Temporal difference learning and TD-Gammon. *Communications of the ACM* 38, 3 (1995), 58–68.
[153] Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. 2022. Lamda: Language models for dialog applications. arXiv:2201.08239. Retrieved from https://arxiv.org/abs/2201.08239
[154] Jason Toy, Josh MacAdam, and Phil Tabor. 2024. Metacognition is all you need? Using introspection in generative agents to improve goal-directed behavior. arXiv:2401.10910. Retrieved from https://arxiv.org/abs/2401.10910
[155] Miles Turpin, Julian Michael, Ethan Perez, and Samuel Bowman. 2024. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. *Advances in Neural Information Processing Systems* 36 (2024).
[156] Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. 2023. On the planning abilities of large language models-a critical investigation. *Advances in Neural Information Processing Systems* 36 (2023), 75993–76005.
[157] Frank Van Harmelen, Vladimir Lifschitz, and Bruce Porter. 2008. *Handbook of Knowledge Representation*. Elsevier.
[158] Niki van Stein and Thomas Bäck. 2024. LLaMEA: A large language model evolutionary algorithm for automatically generating metaheuristics. *IEEE Transactions on Evolutionary Computation; arXiv preprint arXiv:2405.20132* (2024).
[159] Fien van Wetten, Aske Plaat, and Max van Duijn. 2025. Baba is LLM: Reasoning in a game with dynamic rules. arXiv:2506.19095. Retrieved from https://arxiv.org/abs/2506.19095
[160] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems* 30 (2017).
[161] Marcel V. J. Veenman, Bernadette HAM Van Hout-Wolters, and Peter Afflerbach. 2006. Metacognition and learning: Conceptual and methodological considerations. *Metacognition and Learning* 1 (2006), 3–14.
[162] Suzan Verberne. 2024. *Is the Search Engine of the Future a Chatbot?* Inaugural lecture, Leiden University.
[163] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in Neural Information Processing Systems* 32 (2019).
[164] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. arXiv:1804.07461. Retrieved from https://arxiv.org/abs/1804.07461

[165] Boshi Wang, Sewon Min, Xiang Deng, Jiaming Shen, You Wu, Luke Zettlemoyer, and Huan Sun. 2023. Towards understanding chain-of-thought prompting: An empirical study of what matters. arXiv:2212.10001. Retrieved from https://arxiv.org/abs/2212.10001

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

[167] Hui Wang, Michael Emmerich, Mike Preuss, and Aske Plaat. 2023. Analysis of hyper-parameters for alphazero-like deep reinforcement learning. *International Journal of Information Technology & Decision Making* 22, 02 (2023), 829–853.

[168] Huaijie Wang, Shibo Hao, Hanze Dong, Shenao Zhang, Yilin Bao, Ziran Yang, and Yi Wu. 2024. Offline reinforcement learning for LLM multi-step reasoning. arXiv:2412.16145. Retrieved from https://arxiv.org/abs/2412.16145

[169] Jiuqi Wang, Ethan Blaser, Hadi Daneshmand, and Shangtong Zhang. 2024. Transformers learn temporal difference methods for in-context reinforcement learning. arXiv:2405.13861. Retrieved from https://arxiv.org/abs/2405.13861

[170] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. arXiv:2203.11171. Retrieved from https://arxiv.org/abs/2203.11171

[171] Yuqing Wang and Yun Zhao. 2024. Metacognitive prompting improves understanding in large language models. arXiv:2308.05342. Retrieved from https://arxiv.org/abs/2308.05342

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

[173] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems* 35 (2022), 24824–24837.

[174] Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin Choi. 2023. Generating sequences by learning to self-correct. arXiv:2211.00053. Retrieved from https://arxiv.org/abs/2211.00053

[175] Pengcheng Wen, Jiaming Ji, Chi-Min Chan, Juntao Dai, Donghai Hong, Yaodong Yang, Sirui Han, and Yike Guo. 2025. Thinkpatterns-21k: A systematic study on the impact of thinking patterns in LLMs. arXiv:2503.12918. Retrieved from https://arxiv.org/abs/2503.12918

[176] Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. 2023. Large language models are better reasoners with self-verification. arXiv:2212.09561. Retrieved from https://arxiv.org/abs/2212.09561

[177] Siwei Wu, Zhongyuan Peng, Xinrun Du, Tuney Zheng, Minghao Liu, Jialong Wu, Jiachen Ma, Yizhi Li, Jian Yang, Wangchunshu Zhou, et al. 2024. A comparative study on reasoning patterns of OpenAI's o1 model. arXiv:2410.13639. Retrieved from https://arxiv.org/abs/2410.13639

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

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

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

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

[182] Fengli Xu, Qianyue Hao, Zefang Zong, Jingwei Wang, Yunke Zhang, Jingyi Wang, Xiaochong Lan, Jiahui Gong, Tianjian Ouyang, Fanjin Meng, et al. 2025. Towards large reasoning models: A survey on scaling LLM reasoning capabilities. arXiv:2501.09686. Retrieved from https://arxiv.org/abs/2501.09686

[183] Jundong Xu, Hao Fei, Liangming Pan, Qian Liu, Mong-Li Lee, and Wynne Hsu. 2024. Faithful logical reasoning via symbolic chain-of-thought. arXiv:2405.18357. Retrieved from https://arxiv.org/abs/2405.18357

[184] Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. 2024. A survey on knowledge distillation of large language models. arXiv:2402.13116. Retrieved from https://arxiv.org/abs/2402.13116

[185] Ling Yang, Zhaochen Yu, Tianjun Zhang, Shiyi Cao, Minkai Xu, Wentao Zhang, Joseph E. Gonzalez, and Bin Cui. 2024. Buffer of thoughts: Thought-augmented reasoning with large language models. arXiv:2406.04271. Retrieved from https://arxiv.org/abs/2406.04271

[186] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems* 35 (2022), 20744–20757.
[187] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems* 36 (2024).
[188] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. arXiv:2210.03629. Retrieved from https://arxiv.org/abs/2210.03629
[189] Fei Yu, Hongbo Zhang, Prayag Tiwari, and Benyou Wang. 2023. Natural language reasoning, a survey. *Comput. Surveys* (2023).
[190] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, et al. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. arXiv:1809.08887. Retrieved from https://arxiv.org/abs/1809.08887
[191] Eric Zelikman, Jesse Mu, Noah D. Goodman, and Yuhuai Tony Wu. 2022. Star: Self-taught reasoner bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems (NeurIPS)* (2022).
[192] Yizhuo Zhang, Heng Wang, Shangbin Feng, Zhaoxuan Tan, Xiaochuang Han, Tianxing He, and Yulia Tsvetkov. 2024. Can LLM graph reasoning generalize beyond pattern memorization?arXiv:2406.15992. Retrieved from https://arxiv.org/abs/2406.15992
[193] Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2023. Automatic chain of thought prompting in large language models. arXiv:2210.03493. Retrieved from https://arxiv.org/abs/2210.03493
[194] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. arXiv:2303.18223. Retrieved from https://arxiv.org/abs/2303.18223
[195] Chuanyang Zheng, Zhengying Liu, Enze Xie, Zhenguo Li, and Yu Li. 2023. Progressive-hint prompting improves reasoning in large language models. arXiv:2304.09797. Retrieved from https://arxiv.org/abs/2304.09797
[196] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2024. Judging LLM-as-a-judge with MT-bench and chatbot arena. *Advances in Neural Information Processing Systems* 36 (2024).
[197] Wanjun Zhong, Siyuan Wang, Duyu Tang, Zenan Xu, Daya Guo, Yining Chen, Jiahai Wang, Jian Yin, Ming Zhou, and Nan Duan. 2022. Analytical reasoning of text. In *Findings of the Association for Computational Linguistics: NAACL 2022*. 2306–2319.
[198] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. 2023. Least-to-most prompting enables complex reasoning in large language models. arXiv:2205.10625. Retrieved from https://arxiv.org/abs/2205.10625

Received 17 July 2024; revised 25 October 2025; accepted 2 November 2025