Exploring the Computational Necessity of Dual Processes for Intelligence

MSc Thesis *(Afstudeerscriptie)*

written by

Clara Zoe Riedmiller
(born January 12th, 2000 in Karlsruhe, Germany)

under the supervision of **Dr Erman Acar** and **Dr Giorgio Sbardolini**, and submitted to the Examinations Board in partial fulfillment of the requirements for the degree of

MSc in Logic

at the *Universiteit van Amsterdam.*

Date of the public defense:	**Members of the Thesis Committee:**
August 28, 2025	Dr Maria Aloni
	Dr Erman Acar
	Dr Giorgio Sbardolini
	Dr Karolina Krzyzanowska
	Dr Martha Lewis

INSTITUTE FOR LOGIC, LANGUAGE AND COMPUTATION

Abstract

The recent success of deep-learning based *Large Language Models (LLMs)* (Grattafiori et al. 2024; T. Brown et al. 2020) across a wide range of tasks has established them as the most promising candidate for achieving human-level *Artificial Intelligence (AI)*. However, their simultaneous systematic failures on *formal reasoning tasks* (Xu et al. 2025) demonstrate their struggle with *robust generalization* (Marcus 2020). This raises questions about the nature and limitations of their intelligence, which is poorly understood on a theoretical basis (van Rooij et al. 2024; Bender et al. 2021). This thesis investigates how understanding different AI models through the lens of their inference mechanisms can shed light on this issue, while drawing on insights from Dual Process Theory (Evans and Stanovich 2013). This framework from Cognitive Science proposes that cognition consists of two types of thinking: System 1, which makes fast, approximate and implicit inferences and System 2, which reasons slowly, accurately and explicitly. We demonstrate that neural network architectures fundamentally align with the characterization of System 1, while symbolic systems align with System 2. To formalize these claims, we apply a complexity-theoretic analysis that allows us to understand AI under real-world resource constraints while assessing intelligence in relation to the cost-accuracy tradeoff (Johnson and Payne 1985). This effort will provide insights into LLMs' struggle with formal reasoning problems from a theoretical standpoint by framing them as System 1 (Kambhampati et al. 2024). Further, it will argue for the necessity of interaction with a System 2 component to achieve intelligent behavior. These examinations provide evidence for the computational necessity of dual processes for intelligence.

Acknowledgments

I want to thank a few of the people that have supported me throughout the writing of this Thesis and entire degree. First of all, the many students from my year, with whom I shared long homework-solving sessions and (admittedly also long) lunch breaks - to name a few: Emma, Orestis, Spyros, Sid, Stefano, Kira, Lucrezia, Francijn, Amity, Mees, Bas. A special mention to Lorenz, who has been my closest ally from our first course (MPML) to our last (CLT). I also want to shout out my friends and family at home, who always have an open ear for Logic- or Life-related issues (though these were almost synonymous at points). Further, thank you to the ILLC staff for creating an exceptionally supportive learning environment, and for always making time to answer questions. Lastly, and most importantly, I want to thank my supervisors. Both of you allowed me a lot of freedom in the execution of my idea and thereby extended a lot of trust in my conceptual intuitions. Thank you, Erman, for handing me over the initial idea of exploring Formal Reasoning in LLMs from a Complexity Theoretic standpoint. Our meetings always left me with many interesting impulses, as well as great quotes ("Scaling up an LLM in the hope that this leads to reasoning capabilities is like praying for the rain with more people."). And thank you, Giorgio, for two things: For one, providing a very critical eye that helped me identify conceptual challenges in my theory. Second, for your genuine excitement and commitment to help me solve these problems - from optimizing Pareto-Optima to unpacking PAC-Learning.

Contents

Chapter 0

Introduction

Imagine yourself on a late Sunday afternoon in the year 2025. You are an ambitious planner and your household is well-organized - so you decide to get a head-start on the week and revise your meal expenses for the past month. Every single day, you have religiously collected the receipts for lunches and dinners with the goal of keeping your daily budget below 20€ with a couple of exceptions for fancy evenings out. You begin by writing down the sums of pairs of lunch and dinner totals per day until you have a long list of thirty addition problems, that looks like the following:

$$[13 + 5 =?, 6 + 14 =?, 3 + 16 =?, 5 + 8 =?, 0 + 15 =?, 6 + 58 =?, 12 + 7 =?, ...]$$

While you are rigorous, you are also not unreasonable - when there are *Artificial Intelligence (AI)* assistants that can take this list of problems and input the solution without further instruction, who would sit down and type them into a calculator one by one or even solve them manually? You get back the following list:

$$[13 + 5 = 18, 6 + 14 = 20, 3 + 16 = 19, 5 + 8 = 13, 0 + 15 = 15, 6 + 58 = 66, 12 + 7 = 19, ...]$$

Once you ensure your daily budget was never exceeded, you tally up the total - and notice that it is a couple of euros off from what was deducted from your bank account. So you go back and check the AI's work - while most of the sums are calculated correctly, there seem to be some issues. $'6 + 58 = 66'$ for instance was 2 off - not a big deviation from what it should be. Yet, of course still wrong - and for tasks like budgeting, minor mistakes like this defeat the whole purpose. You wonder what went wrong and discover that there seems to be a general issue with summing up bigger numbers. The solutions are never far off from the correct answer and thus certainly not *un*intelligent. The AI seemingly had no problem solving small sums, yet failed to do anything more than just 'something like addition' when the numbers got bigger and the problems harder.

This anecdote illustrates three big problems arising from contemporary, deep-learning based AI implementations:

- **Robust Generalization.** Deep-Learning based AI excels at many tasks but fails to generalize this ability to similar problems (Marcus 2020). This can look like it did in our example, where the model could solve addition problems with smaller numbers but fails to generalize the same solution method to harder problem instances. This lack of consistency across similar tasks is often referred to as *brittle* behavior, as opposed to *robustness* (Marcus 2020). A lack of robust generalization leads to undesirable and often unpredictable drops in performance across problem instances.

- **Explainability.** Not only do these systems deliver inaccurate solutions, but we also do not know how they arrive at them. In our example, the AI gave us back the completed list without specifying what it did to solve the problems. And even if it told us it performed *'addition'*, due to the fundamental opaqueness of large, complex neural architectures and vast amounts of training data of deep-learning based reasoners (Marcus 2020), we cannot ensure that this reliably tracks how we intend the +-operator to work. We do not know, for instance how the model solved $6 + 14 =?$ - maybe it understood addition but only for smaller numbers, maybe it has seen this problem before and merely memorized the solution. This opaqueness limits us to treating the AI models like *'animals in the wild'* (Subbarao Kambhampati 2024). Understanding what they can and cannot do is limited to behaviorist investigations and probing them with more tasks until they fail - or do not.

- **Lack of Theory.** Both above points make apparent that the behavior of current AI systems are not very well understood. They behave unpredictably and we do not know *how* they arrive at this behavior - and resultantly even less so *why*. For one, this leads to concerns about problematic societal implications and environmental resource consumption (Bender et al. 2021). Further, this lack of theory has direct consequences for AI research itself. These follow from two main objectives of the field: *Assessing* the intelligence of AI models, while at them same time *building* these models. The implications for assessment have already been outlined above - being restricted to behaviorist probing means we need to ask the right questions to understand these models limitations. If we fail to do so, we might wrongly attribute them with intelligence they do not possess and miss crucial weaknesses. Second, without a lack of theory, once we identify limitations of current AI models, there is no clear path forward to resolve them. If we do not know why they make mistakes, we do not know what changes would effect them stopping. Lastly, both of these facets contribute to a general uncertainty about whether expanding on current limited approaches can genuinely lead to generally intelligent systems. As Mitchell (2021) point out, there is no guarantee that narrow and general intelligence lie on one continuum.

Understanding all of these factors and their interplay is central in obtaining a grasp on current AI discourse. The absence of a theory is deeply intertwined with the opaqueness of large neural networks. This leads to a lack of understanding of contemporary AI systems, which can easily be overshadowed by an enthusiasm about their performance (van Rooij et al. 2024). Historically, surges of great optimism about a new AI model's performance have always been followed by periods of skepticism as soon as its limitations move into the focus of the debate (Mitchell 2021). This aligns with the current discourse about deep-learning based AI in general, and *Large Language Models (LLMs)* specifically. Upon their introduction, initial breakthroughs in tasks like natural language processing and creative problem solving quickly rendered LLMs the most promising candidate for human-level AI (Grattafiori et al. 2024; T. Brown et al. 2020; Devlin et al. 2019). Over the last year however, more rigorous analysis revealed of their systematic struggles with formal reasoning problems (Xu et al. 2025; Mirzadeh et al. 2024; Burnell et al. 2023). These types of problems include mathematical and planning problems[1], on which the LLMs display brittle performance: They can often solve some task instances, while this skill does not generalize to similar ones (Marcus 2020). This limitation puzzles the AI research community and leads many to hypothesize about the nature of the models' intelligence. After all, it would be strange to observe the same behavior in a human, who can add up numbers up to 100 but fails to do so reliably with bigger numbers.

Some have proposed that the LLMs' curious behavior is an effect of them being able to perform one type of thinking, while entirely lacking another that humans possess additionally. Specifically, that modern AI can only think by extracting pattern-based heuristics from their training data instead of obtaining a structural understanding of the problem. In the cases where the pattern fails, the models are incapable of thinking more deeply about the task to come up with a better answer. Conversely, it is reasonable to assume that a human might also miscalculate $'385 + 493 = 892'$ when making a quick inference. However, when we inform the person of their mistake, they can decide to take more time and evaluate their strategy more thoroughly the second time around, eventually arriving at the correct answer. Human Cognitive Science captures this concept of two distinct types of thinking in *Dual Process Theory (DPT)*. Commonly, these types are characterized as System 1, that is fast, automatic and based on implicit heuristics and System 2 that is slow, deliberate and based on explicit logic (Kahneman 2011; Evans, Jonathan St B T and Stanovich, Keith E 2013).

Relying on this distinction, some researchers propose a conceptual overlap between the System 1 thinking and the abilities and limitations of current AI models. Kambhampati et al. (2024), for instance, suggest that, in the pursuit of AI, we have built a *'giant pseudo System 1'*. Though this intuition seems to be shared by many, as evident in many high-level conceptual debates (such as the *AAAI-20 Fireside Chat with Daniel Kahneman* (2020)) and 'Discussion' sections of research papers (Xu et al. 2025)), there has not been a thorough theoretical exploration of this mapping up until now (Xu et al. 2025). In this work, therefore, we want to ground this intuition in theory. Thereby, we will approach question such as: "How well do the conceptual uses of DPT in the fields of human Cognitive Science and Artificial Intelligence overlap?", "How can insights from one field inform the other?" and "How can this exploration form the basis for a better theoretical understanding of AI?".

Foreshadowing the answer to some of these queries, we briefly outline our strategy and results. This work develops a theoretical scaffolding for the theoretical investigation of AI, which has a computational characterization of System 1 and System 2 at its core. In doing so, we will attack the notions in the title of this work in reverse order: We begin by defining why and how intelligence should be assessed based on theoretical principles in chapter 1, followed by an examination of Dual Process Theory in chapter 2. After deriving an information-processing based characterization of System 1 and System 2,

1. In this work, we will focus our attention on mathematical and planning problems, since these are the easiest to stake out. Other problems include logic, coding and commonsense reasoning tasks. For a taxonomy of reasoning benchmarks, see Xu et al. (2025).

we map these types of thinking to distinct AI implementations in chapter 3. On this basis, chapter 4 will show strong evidence that dual processes are a necessary prerequisite for any kind of intelligence.

Chapter 1

Assessing Intelligence

Getting an LLM to reason is like trying to mow the lawn with your car.

Moshe Vardi
World logic day webinar, January 2025

In this chapter, we will first find a tangible definition of intelligence, then investigate what consequences this definition has for the assessment of artificial intelligence. In doing so, we will first look at the architectures that are commonly considered AI at the moment and outline some of their fundamental properties. Next, we will explore Benchmarking as the predominant method of evaluation and show that it fails to capture important metrics of intelligent behavior. Specifically, we will see how an exclusively behaviorist assessment of a model contributes to the lack of understanding about their task-solving strategies, and even blurs error traces that could help us to understand them in the future. We will see that this lack of understanding can only be mitigated by a theoretical approach to artificial intelligence and propose what that could look like.

1.1 Terminology

The goal of this section is to find a working definition of Intelligence. To assess the Intelligence of a system, we need to clearly define what it measures. We also want to keep in mind that we are trying to arrive at a general truth about human and artificial intelligence, so our second objective will be to make these two manifestations comparable. In doing so, we will establish a clear language that allows us to demystify central concepts and rid them of anthropocentric connotations.

1.1.1 Why is this Necessary?

The concept of *intelligence* is notoriously hard to pin down. Even when it only describes human behavior, the terminological vagueness effects disagreements in the theoretical and practical assessments (Burgoyne et al. 2023). However, there is no way around tackling this concept when we want to assess the success of artificial intelligence - which, trivially, should be measured by an artificial system's ability to behave intelligently. For the scope of this work, this prompts us to define a working definition that results in clear requirements for intelligent behavior. This addresses two main issues when talking about *artificial* intelligence specifically.

First, by setting clear goal posts, we can avoid falling victim to the *AI effect*, where intelligent behavior is defined as whatever current models cannot do (e.g. Nadin (2023)). Second, ensuring this definition is void of human connotations allows for an unbiased assessment of both human and artificial systems. Otherwise, our terminology might act as *wishful mnemonics*, where labeling an artificial agent's behavior as *intelligent*, talking about what it *understands*, *thinks* or even *learns* could misguide us into falsely concluding parallels to human cognitive abilities and mechanisms (McDermott 1976; Mitchell 2021, 2024).

Our goal in finding this working definition will not be to introduce even more fine-grained definitions of these terms, but rather break them down to their conceptional core. In doing so, we will strip them of their anthropocentric origins and define intelligent cognition solely in terms of information-theoretic problem solving.

1.1.2 Mind over Matter

The first step of making these different systems comparable is removing hardware from the discussion. We want to have a notion of intelligence independent of the physical realization of the intelligent system. Thus, we adapt the following two positions from the philosophy of mind:

- **Functionalism.** Functionalism is the view that what matters about an agents behavior is what function they implement (what they can do) rather than their physical implementation (what they are made of) (Levin 2023). This allows us to remove the distinction between carbon-and silicon based lifeforms; what matters to us, in assessing intelligence, is simply the agent's ability to think (definition follows below).

- **Computationalism.** Further, we will also take a particular stance on how thinking works in human agents. While it is clear that any artificial intelligence is physically realized on a computer and thus can be viewed as a form of computation, we will assume the same about the human mind (Rescorla 2024). Whenever we talk about human thinking (details below), we are assessing a computation. Defining it as such allows us to track its operational constituents (input, algorithm, output) as well as resources (time, space, data) and accuracy metrics.

Starting from these assumptions, we will now construct a working definition of intelligence. A flavor of circularity in these definitions is unavoidable, yet we will outline the necessary terms and dependencies from the ground up as best as possible.

1.1.3 Thinking as Information Processing

In the most basic terms, intelligence measures how something behaves. Thus, we will begin by describing what this something is and what behavior we are concerned with. We will start of with tackling the latter: Our assumption of Computationalism (section 1.1.2) allows us to view human cognition in its entirety as information processing. This means *thinking*, at its core, can be understood as the processing steps a system takes to produce an output from an input. In most cases, we will be framing these in terms of a problem and its solution. Of course, in real life it would be reasonable to require that, the other way around, not all computation should be viewed as thinking. A cash register, for instance, is not commonly assumed to think. However, since thinking is only an auxiliary notion for us and we ultimately only care about *intelligent* thought, this distinction is not important to us.

Definition: Thinking

We define *thinking* as the information processing steps between input and output.

Equivalent terms: reasoning, (information) processing, cognition, problem solving.

Important to note here is that setting *thinking* equivalent to *reasoning* is not common practice. Yet, we do not want *reasoning* to allude to any superior mystical capabilities that specific type of information processing may or may not have based on this notion. The challenge with the term *reasoning* stems from its varied usage. For instance, what specific characteristic, if anything would constitutes reasoning tho render it different from information processing? And what does AI literature mean by *reasoning* tasks as a consequence? We disambiguate this term by distinguishing:

- **Reasoning Practice.** We view thinking and reasoning as equivalent terms denoting information processing. Instead, a crucial differentiation lies between *exact* and *approximate* reasoning.

- **Reasoning Tasks.** According to our definition, problem solving is a computation, all computations are thinking and all thinking is reasoning. Thus, all problems are reasoning tasks. To denominate what specific task types we address when saying something along the lines of "LLMs struggle with reasoning problems", we will refer to these tasks as *formal* reasoning problems (a characterization follows in section 1.4.2).

Last, let us also pre-emptively define some more concepts related to *problems.* We will use this term to refer to a set of problem *instances* that require a shared solution strategy. For instance, the problem of *addition* requires applying the rule $'s = x + y'$ to problem instances like $'385 + 493 =?'$. Singular problems can be placed in a bigger hierarchy of problem types, for instance: *addition* $\subsetneq$ *arithmetic* $\subsetneq$ *mathematical problems* $\subsetneq$ *planning problems.*

Our second auxiliary notion describes the entity that performs *thinking*, which we will call *agent.* By our assumption of Functionalism (section 1.1.2), the physical realization of this agent does not concern us. To again keep our conceptual baggage minimal, *system* or *model*, as well as *learner* or *reasoner* will be used interchangeably with *agent.* The only further requirement is self-containment; Thinking needs to happen independently of any other system's involvement.

Definition: Agent

We define *agent* as an entity capable of thinking independently.

Equivalent terms: system, model, reasoner, learner.

1.1.4 Understanding as Internal Representation

Lastly, we will define *understanding*, which will only become central in later chapters. For the sake of completeness, and to also rid this term of human connotations, we include it here. Understanding for us denotes the possessing the internal representation of a concept. For instance, an agent can be said to understand the concept of *addition* its internal representation gives it the capacity to correctly reason about all problem instances of addition.

Definition: Understanding

We define *understanding* a concept as possessing an internal representation that enables correct reasoning based on it.

1.1.5 Intelligence as Optimality

After setting up these auxiliary notions, we can now define intelligence. In broad strokes, intelligence measures whether a system *thinks well*. We have already defined thinking and system, now we are only lacking what *well* means. For this, we will begin by looking at how the terminology and research objective developed over the years in measuring it, which will allow us to derive a tangible and measurable definition of intelligence. In doing so, we will touch upon slightly varying objectives and concepts that this metric has traversed through the years and discuss the notions of *rationality* and *bounded rationality*. For us, it is not important to distinguish between (bounded) rationality and intelligence and we will use these terms interchangeably following our final definition. In outlining their historical differences, we primarily seek to understand the different factors we should consider when assessing whether an agent thinks well. Ultimately, we will submit to convention and aim to define *intelligence* for the remainder of this work. We stick to this notion mostly out of convention, since we are talking about artificial *intelligence* (c.f. Buckner (2024)).

Let us now delve into historical approaches to these assessments. Since intelligence and rationality define properties of a system, we can best understand these concepts by seeing *how* they are measured (Burgoyne et al. 2023). Historically, intelligence has been the predominant measure of good thinking, with IQ (intelligence quotient) test mostly assessing an agent's ability to give a normatively correct answer to puzzles, mathematical problems or analogy making defined by logical validity (K. E. Stanovich 2009). However, these test seem to miss an important aspect of human cognition, that can dynamically adapt to nuanced environmental factors beyond a singular correct reasoning schema (K. E. Stanovich 2009). Consider for instance the following scenario, where a friend is visiting your house and asks "Do you have salt?" Given that you run a well-organized household, the logically correct answer is "Yes.". However, human behavior is much more complex in practice, where you would probably answer "It is in the cupboard." - this reply, technically speaking, does not address the question at all. However, it responds to the spatio-temporal and social context to give the answer the person was probably looking for - thereby optimizing the informational content instead of the objective correctness (Grice 1975). This framing of optimizing one's response *adaptively* relative to the environmental context instead of the pure *formal correctness* is central to the concept of *rationality* (K. E. Stanovich 2009). Lastly, the field of *bounded rationality* considers optimality in relation to resource factors specifically(H. A. Simon (1997), Gigerenzer (2020)). A central insight for the field of bounded rationality is that any real-life system always operates under resource constraints and thus has to consider the optimality of their thinking in relation to how much its execution costs. This constraint leads to tradeoffs, where minimizing the cost of problem solving often comes at the sacrifice of other factors. In the following, we restrict our attention to one of the most prominent ones: The *cost-accuracy tradeoff* (originally the effort-accuracy tradeoff (Johnson and Payne 1985)). As we will see, this concept will be a guiding principle when assessing intelligence, allowing us to understand fundamental limitations in reasoning performance of any real-life, and thus resource-bounded, system. Intelligence becomes a question of optimality instead of perfect behavior, relating a reasoner's performance to the cost of its strategy.

To further illustrate the implications of this behavioral lens, consider the following example (inspired by Thorstad (2025)). Imagine you are very hungry, so you go to the grocery store to find a candy bar. You make it your goal to find the one with the most amount of calories, since you know it will best combat your hunger. You devise different strategies to identify this bar among a shelf with 5 rows and 20 bars per row. Each strategy will lead to you picking a specific bar, so we can respectively determine the *cost* of this choice measured in amount of time it takes you to pick out this bar and *accuracy*, where 100% represents choosing the bar with is the highest calorie count and 0% the one with the lowest. In practice, measuring this accuracy depends many factors, which do not add to our argument at this point. Thus, we give estimates of the strategies' average accuracy. You devise the following four strategies:

- **Option (A).** You go through all 100 bars one-by-one, starting from the top left and ending at the bottom right. You pick up each one and retain in your hand whichever candy bar has the most calories so far. *Cost:* Let's say picking up a bar, reading the label and assessing whether it is higher or lower than the previous record takes 3 seconds. In total, this strategy will take you 5 minutes. *Accuracy:* 100% - this strategy guarantees you the optimal solution.

- **Option (B).** You scan the shelf and intuitively choose the 10 bars that, based on their packaging, look like they have the most calories. Then apply strategy A to this subset. *Cost:* Scanning and picking takes you one minute, and applying A another 30 seconds, resulting in 90 second total. *Accuracy:* Assuming a decent correlation between packaging and calories, we assume this results in 80% accuracy.

- **Option (C).** You intuitively choose the bar that looks like it has the most calories from the whole shelf. *Cost* 20 seconds. *Accuracy:* Again, assuming a decent correlation between packaging and calories, we estimate 60% accuracy.

- **Option (D).** You simply take the bar on the top left corner. *Cost:* Let's say this takes you 2 seconds. *Accuracy:* We estimate that this gives you 50% accuracy.

Because we see these strategies as computations, we can now give estimates on how successful these strategies were with respect to cost and accuracy. We can see that for this example, like many others, there is a positively monotone correlation between cost and accuracy. The more resources we invest, the more accurate our solution will be. Importantly, the optimal answer is determined by how we weight these dimensions. In alignment with a conventional use of *'intelligence'*, in most cases, it could be considered intelligent for a very hurried person to opt for Option B. At the same time, a more hungry person with slightly more time could behave intelligently by choosing option A. This example thus highlights that the optimal (intelligent) behavior is not determined by the task itself but also immediately dictated by the task environment (how much we weigh accuracy and cost).

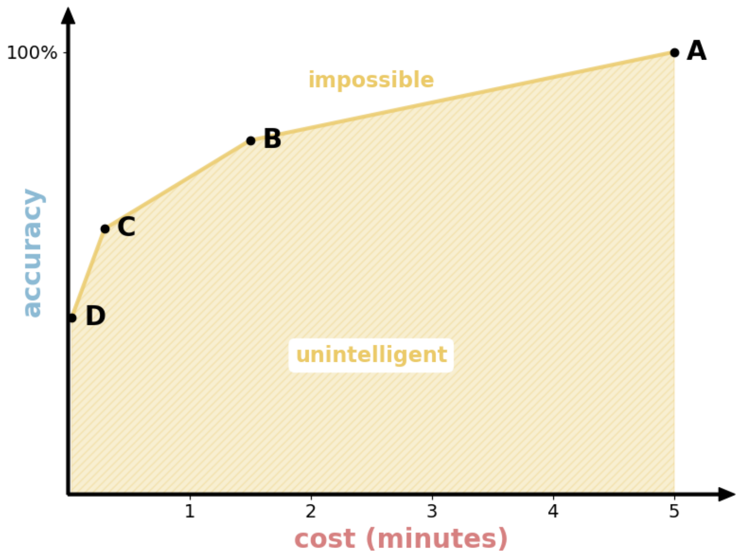


Figure 1.1. Cost-accuracy tradeoff between different strategies based on our estimations.

What this also exemplifies: The best strategy does not depend on the task itself - these are all optimal strategies in bounded rationality. Instead, which one we should choose depends on factors of the environment. If a person is so hungry that they will soon pass out, the intelligent strategy would be to opt for D. If a person is moderately hungry but has some time, they should go for one of $\{B, C, D\}$. And if a person goes to the same supermarket every day and wants to settle the highest calorie count once and for all, they might go with A. Which option they chose should depend on how the environmental factors weight cost and accuracy against each other. All these strategies are optimal with respect to the cost-and accuracy tradeoff: There is no strategy as cheap as D that is more accurate and no strategy that is as accurate as A but cheaper. These options form the **pareto-frontier** of the cost-accuracy tradeoff for this task. This means they are all optimal in some way - when one of the parameters is fixed, a strategy on the pareto-frontier guarantees that the other one is as good as it can be.

So which option should an *intelligent* reasoner choose then? It seems like we might be done with the definitions now, we have already established that all of the strategies are optimal. But actually, there is an additional requirement we will fix before we arrive at a criterion for intelligence. Namely, if a person were about to pass out from hunger, it would not be intelligent to opt for anything but D, and conversely, if the situation is less urgent, it would not be intelligent to opt for D. This highlights the importance of being able to consider environmental factors of the task context when choosing among different pareto-optimal strategies and, crucially, being capable of executing all of them. An agent that can perform B to D but not A is surely not stupid - but at the same time not fully intelligent. This allows us to conclude with a refined definition of intelligence:

Definition: Intelligence

We define *intelligence* as the competence to execute a pareto-optimal problem solving strategy with respect to the cost-accuracy tradeoff that optimizes environmental demands for these parameters.

Equivalent terms: (Bounded) Rationality.

Lastly, we can extend this definition from the capacity to solve one problem intelligently to a more general type of intelligence. In AI, the common goal for intelligent systems is often referred to as *human-level* intelligence. Notably, the interpretations of this benchmark differs slightly depending on whether human-level refers to the intellectual ability of a singular human mind or all of humanity (Mitchell 2024). For us, the weaker threshold suffices, as intelligent behavior on formal reasoning problems differentiates AI from individual human abilities (LeGris et al. 2024; Stevenson et al. 2024).

1.2 From Definition to Assessment

Now, that we have a definition of intelligence, the most straight-forward way to test whether a model is intelligent would be to see how it performs on tasks respective to the cost-accuracy tradeoff. However, as we will see in the following, an assessment purely based on performance does not bring us closer to understanding this intelligence. For instance, Zellinger and Thomson (2025) highlights that first, we need to fix the semantics of *'cost'*, after which we can observe that different models display varying intelligence depending on this interpretation, the specific task and environmental conditions. If we want to assess the *general* intelligence of an AI, as well as *understanding* its behavior, we thus need to examine these factors. To begin our investigations, we will first identify what we mean when we talk about contemporary AI (section 1.3), and investigate how these models perform on different tasks (section 1.4). In doing so, we will restrict our attention on the model's performance *accuracy*. Focusing on this one dimension will allow us to get a clear picture of their real-life abilities first. In the later chapter 4, we will then reintroduce the cost dimension and formally show how the kind of thinking an AI engages in, combined with real-world resource constraints, determines how its strategies interact with the cost-accuracy tradeoff. Our explorations will make demonstrate that exclusively performance-based assessments are fundamentally limited in their explanatory potential, necessitating a better theoretical framework. Following this observation, we will introduce a principled methodology for a more scientific investigation of AI (section 1.7).

1.3 LLMs as Contemporary AI

In this section, we will examine which computational models contemporary discourse commonly refers to when talking about *AI*. Mostly, this term refers to large neural network-based architectures trained with deep learning (section 3.3). Their empirical success has crystallized them as the most promising candidate for achieving human-level intelligence (Buckner 2024; LeCun, Bengio, and Hinton 2015). At the center of this development lie prominent breakthroughs in tasks like computer vision (Krizhevsky, Sutskever, and Hinton 2012) and language and speech processing. Especially the latter has proved a promising application of these architectures, almost rendering the term *'LLM'* conventionally synonymous to *'AI'*. Their empirical success on a wide range of tasks (Grattafiori et al. 2024; T. Brown et al. 2020; Devlin et al. 2019; Xu et al. 2025) is commonly attributed to two main factors: The choice of language as domain (Rothschild 2025), and the underlying *transformer* architecture (Vaswani et al. 2017). These factors in combination allows the models to capture and learn from long-range contextual dependencies in information-dense data, in turn enabling them to display *emergent* problem solving abilities for a wide range of applications (Wei, Tay, et al. 2022).

As motivated in the introduction, the initial optimism about these abilities was quickly met by skepticism about the nature of this AI following the observation that they struggle to display robust performance, especially on formal reasoning tasks Xu et al. (2025). This has prompted a shift in focus, with current LLM development largely centered around improving the reasoning abilities of these models and thereby turn them into *Large Reasoning Models (LRMs)*. While some empirical success in this direction has been achieved, their behavior remains brittle and lacks a theoretical basis (Shojaee et al. 2025a; OpenAI 2024b; Xu et al. 2025). Thus, For the sake of clarity, we will focus our attention on LLMs in the following, while the insights we gain likely generalize to LRMs.

1.4 On Benchmarking and Behaviorism

As motivated above, contemporary AI models are poorly understood, with many asking: "[H]ow can these systems be so smart, yet also seem so limited?" (Browning and LeCun, Yann 2022). In the following, we will first discuss how the performance (smartness) is typically assessed by performance metrics (section 1.4.1) and how this method established the limitations of an agents, such as our initial motivation that LLMs struggle with formal reasoning problems (section 1.4.2).

1.4.1 The Imitation Game

The oldest and most famous test of AI as proposed by Turing (1950) is the *Imitation Game* (a.k.a *Turing Test*. In this setup, a human tester communicates via writing with two hidden agents, one of them a computer and one another human. The goal is to tell which is which - once the tester can no longer tell them apart, the computer is attributed human-level intelligence.

Contemporary AI, specifically LLMs are now widely considered to have passed this threshold (Jones and Bergen 2025). Yet, as motivated above, their behavior remains uneven and, in systematic evaluations, we can still detect their curiously inconsistent performance, as with formal reasoning tasks. If the tester thus prompted the agents strategically with tasks of this nature, they could likely still differentiate AI from human, indicating that it falls short of human-level intelligence (LeGris et al. 2024; Stevenson et al. 2024). So why are these limitations not more apparent?

One possible explanation posits that most every-day conversations do not probe deeply. AsBrowning and LeCun, Yann (2022) highlight: "[I]n many cases, the surface is enough; few of us really apply the Turing test to other people, aggressively querying the depth of their understanding and forcing them to do multidigit multiplication problems. Most talk is small talk." And this small-talk is what LLMs have really mastered. If we applied the Turing Test thoroughly, we would include questions requiring varying levels of structured problem-solving, like "What do you want for lunch?", "What is the capital of France?" and "What is the sum of four-hundred-fifty-six and six-thousand-three-hundred-eighty?", which could increasingly highlight the inability of the AI to reliably answer prompts like the latter.

The human agent could probably solve all these tasks, with the last one being the most difficult. When testing the intelligence of a human, we rely on our assumption of hierarchical difficulty: If a person can solve the math problem, they could also solve all other problems, as they are easier. Thus, for assessing human intelligence, testing these abilities suffices, as apparent from the focus on logical problem solving in IQ test (K. E. Stanovich 2009)). A meaningful behaviorist assessment this requires a good understanding of which tasks are difficult for the system under investigation. To understand why implicitly assumed that this hierarchy is reversed for computers compared to humans, we need to take a brief historical detour. In the beginning of AI, symbolic systems, which we can think of as big calculators for now (section 3.2), were the predominant method for implementing computational intelligence. For these systems, *Moravec's paradox* states the reversal of difficulty ordering, claiming that *'hard things are easy and easy things are hard'* [1]. However, contemporary AI operates on entirely different, neural implementations (section 3.3) - an LLM fundamentally works unlike a calculator. From their struggle with formal reasoning tasks, LLMs subjective problem difficulty seems more aligned with that of a human again. Without a theoretically founded assumption, performance-based testing can thus easily lead us astray, ultimately obscuring our grasp on cognition (van Rooij et al. 2024).

To regain clarity, we must first understand *how* AIs struggle with formal reasoning tasks, followed by an investigation of *why*.

1. Originally, Moravec stated the following: "It is comparatively easy to make computers exhibit adult level performance on intelligence tests or playing checkers, and difficult or impossible to give them the skills of a one-year-old when it comes to perception and mobility" (Moravec (1988), cited from Mitchell (2021)).

1.4.2 Formal Reasoning Problems as the Limitation of Imitation

Before we dissect *how* LLMs struggle with formal reasoning problems in section 1.4, we will first characterize them here. This will allow us to identify *where* the failures manifest in their performance and what fundamental intellectual inability this indicates. For this characterization, we will draw on descriptions of formal reasoning benchmarks from the literature (Xu et al. 2025; Patil and Jadon 2025)[2]. These tasks types include commonsense, programming and logic problems, but we will focus our attention on planning and math problems in the following for simplicity. We expect the results we discuss to translate to these other problems as well, due to their common structure:

- **Step-by-step problem solving**. Obtaining a solution to these tasks requires logically consistent multi-step reasoning.
- **Exactness.** A correct solution has to be exact instead of approximate, which also requires the exact execution of intermediate steps.
- **Infinite Domains.** There are infinitely many problem instances and variations of a task[3].

Consider, for instance, the *Towers of Hanoi* game (Shojaee et al. 2025a). Obtaining a solution requires moving one disk at a time and following an exact strategy. Further, the game could be played with an arbitrary amount of disks and we can create variations for instance by turning the hollows disks into stackable blocks. In the next section, we will start exploring *how* LLMs struggle with these problems.

1.5 The Expectation of Robust Generalization

We have previously established that LLMs do not fail to solve formal reasoning problems generally - they can often solve some problem instances but not similar ones. These models instead fail to perform *robust generalization*, which we have identified in the introduction as one of the biggest questions in

2. See Xu et al. (2025) page 22 for a comprehensive taxonomy of reasoning benchmarks.
3. Formalizing this more technically: Each concrete problem instance can have an *arbitrarily large* size and we can form *arbitrarily many* variations of a problem. This fact results in a theoretically infinite space of possible problem instances.

contemporary AI research (Marcus (2020), also called *transfer learning*, as in Mitchell (2021)). In the following, we will first precise what types of generalization we would expect from a reasoner and how LLMs fail at these (section 1.5.1), followed by exploring the role of robustness (section 1.5.2).

1.5.1 Types of Generalization

We will outline three types of generalization we expect from a reasoner that understands the concepts that are relevant to solving the task. To illustrate this, let us again consider the example of addition. For this example, let us assume that the data we want the learner to generalize from are samples of integer addition up to 10. If the agent understands addition based on these problem instances with their respective solutions, we would expect it to be able to generalize this knowledge to the following related problem instances (see fig. 1.2): **(1) Within the same problem and the restricted domain**, and correctly solve problems instances of small integer addition it has not seen before, **(2) Within the same problem and an extended domain**, like addition with bigger integers and **(3) To structurally related tasks**, like subtraction. These types are not mutually exclusive and only serve as conceptual aids. For instance, we could technically see the addition of three small integers are either type (2) or type (3).

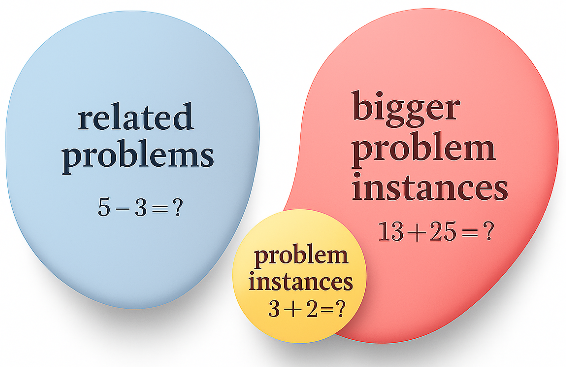


Figure 1.2. Related problem types for which we would expect robust generalization. This example shows the concept of addition, where the original problem instances the learner is exposed to is the pairwise addition of small integers (yellow), the pairwise addition of bigger integers (red) and the structurally related problem of subtraction. Graphic generated with Sora (Open AI 2024a)

.

We will proceed with the investigation of LLM generalization on formal reasoning problems, relative to our types.

(1) Generalization within the same problem and the restricted domain.. Staying with our example of addition, recent work shows that even state-of-the art LLMs have trouble grasping elementary addition (Yan et al. 2025). Their failures seems to stem from a fundamental lack of understanding the *structure* of the domain, such as the compositionality of addition. This effects that they can sometimes produce the correct solution for $x + y$ but not $y + x$. Instead, it seems like they rely on superficial *patterns* in the training data (section 2.4.4).

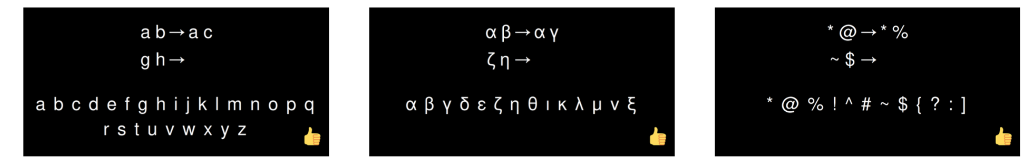
Another canonical example shows an analogous problem in the visual domain. Ribeiro, Singh, and Guestrin (2016) trained a neural network to classify images as depicting huskies or wolves, achieving great accuracy on the training set. However, it learned to output *'wolf'* based on the amount of white in the picture - reflecting a pattern in the training data where wolves were increasingly photographed in snowy environments. Based on this, the classifier performed poorly on general images of the animals.

(2) Generalization within the same problem and an extended domain. It is well-known that LLMs struggle with generalizing to increasingly hard problem instances (D. Zhou et al. 2023; Keysers et al. 2020; Lake and Baroni 2018). Shojaee et al. (2025b), for instance, demonstrate this for the *Towers of Hanoi* game, where the LLMs solve instances with few disks reliably but experience *'complete collapse'* when a certain problem size is exceeded.

(3) Generalization to structurally related tasks. This last type of generalization is also the most abstract; It requires transferring conceptual understanding to structurally related domains. As we exemplified, this could deducing from $'2+3=5'$ to $'5-3=2'$ but also to problems like $'\square\square\square + \square\square = ?'$. This type of generalization is commonly referred to as *analogical reasoning.*

Stevenson et al. (2024) test this ability in LLMs based on *letter-string analogies* (fig. 1.3). While humans display no problem transferring their strategies between different alphabets, LLMs' performances decline when transitioning the task from the Latin to the Greek alphabet (close-domain) and even more upon transfer to a Symbol alphabet (far domain). Lewis and Mitchell (2024) provide similar evidence for further task types, thereby critically assessing previous claims of emergent zero-shot[4] analogical reasoning in LLMs (Webb, Holyoak, and Lu 2023).

4. In *x-shot* reasoning, x denotes the quantity of reasoning examples that were provided to the model as part of the prompt. Zero-shot therefore signifies the absence of such examples.


a b → a c
g h →
a b c d e f g h i j k l m n o p q r s t u v w x y z
α β → α γ
ζ η →
α β γ δ ε ζ η θ ι κ λ μ ν ξ
* @ → * %
~ $ →
* @ % ! ^ # ~ $ { ? :]

(a) Latin (b) Greek (c) Symbol

Figure 1.3. Different domains (ordered lists of symbols) and examples of letter string analogy tasks to be performed based on these domains. Latin (a) is the original domain, Greek (b) a close domain and the Symbol alphabet (c) the far domain. Experiment setup and graphic from Stevenson et al. (2024).

1.5.2 Not Robust? Let's Readjust!

Analyzing LLM's failure to generalize has demonstrated to us that, in many cases, our expectation of knowledge transfer from one problem instance to another hinges upon understanding *structural* similarities between them. This type of bridging between similar problems is especially important when handling formal reasoning problems, due to to their infinite space of problem instances (section 1.4.2, we will establish what this means more formally in section 4.4). Without structural understanding, high accuracy on a a subset of problem instances does not guarantee good performance on new data, as we saw for instance in the *husky-wolf classifier* (section 1.5.1). This highlights the role of the former composite of *robust generalization* - it is not only crucial to generalize, but to do so *reliably* and *consistently*, constituting *robust* behavior.

While we will revisit *patterns* and *structures* at a later point (section 2.4.4), we want to establish another crucial fact here: While it seems like AI can solve problems on the surface, this behavior is often brittle - without *explainability of* and **theory about** these models, instead relying on purely behaviorist methods of assessment, superficial abilities often lead to preemptive attribution of intelligence. This in a convention to the effect of "intelligent until proven guilty", forcing the creation of new benchmarks for every claimed competence. This dialectic effects the repeated creation of new benchmarks, new AI models and newer benchmarks like a dog chasing its own tail, contributing to *benchmark inflation* (Haimes et al. 2024). Since there are infinitely many formal reasoning problems alone (section 1.4.2), we can never perform exhaustive tests. Thus, the cycle can only be broken when the research community agrees that AI's behavior is robust enough to trust that these systems rely on a sufficient understanding of the world.

Before we move on to discuss alterative to behaviorist assessments, we will briefly outline a last additional issue with it that applies specifically to the case of AI in the next section.

1.6 Patch the Case or Fix the Base

A last aspect that highlights the problems with an exclusively performance-based assessment of AI, we discuss the issue of *data contamination* (Cheng, Chang, and Wu 2025). This aspect is uniquely challenging for the examination of artificial compared to human intelligence. To grasp this issue, we need to understand a key difference: We are not only assessing but also building and training these systems simultaneously. This might not only lack a theoretical basis but actively hinder us from obtaining it. To illustrate this point, let us consider the following example.

Suppose you are a quality control officer in a small tech company, where your job is to test a new AI based calculator. You devise a small benchmark, consisting of the following problems:

$$['4+2=?','6+8=?','9+1=?','5+7=?','4+6=?']$$

The system subsequently produces the correct answers for four of the problems but returns $'9+1=11'$. You report this to a software engineer, who retrains the AI, after which the bug is gone. The system then gets released to the public, after which problem reports start flooding in with users reporting that it output $'3+9=13'$ and $'9+9=20'$. In hindsight, the bug was not an isolated issue but indicated a broader systematic failure of the system to handle the number 9. The method the programmer applied was not a *fix* but a mere *patch* of faulty behavior. This method solved the problem temporarily, while also removing error traces - with every patch, it becomes harder to understand the errors systematically.

An analogous problem holds for LLMs. As soon as a benchmark is publicly available, in addition to examples of correct and faulty solution to its problems, this data can appear in an LLM's training. This allows the model to memorize what (not) to answer to the problems, thereby potentially patching its incapabilities instead of correcting them. This issue is known as *data contamination* (Cheng, Chang, and Wu 2025). The *bucket-of-water* anecdote illustrates this dynamic: In an introductory lecture on AI[5], Patrick Winston illustrated AI's failure to engage in common sense reasoning by arguing that it cannot answer the question "What happens if I run down the street with a full bucket of water?". Since this information has likely not been recorded in any corpus of training data, he predicts that it would not be able to give the correct answer along the lines of "Water sloshes around and your leg gets wet". This lecture was recorded in 2010, and, since then, the field has progressed a lot. When a contemporary LLM is prompted the same question[6], it succeeds in producing the answer. Yet, upon further investigation, the transcript of this very lecture had been transcribed on the internet, making it possible that the LLM had memorized the solution.

5. https://www.youtube.com/playlist?list=PLUl4u3cNGP63gFHB6xb-kVBiQHYe_4hSi
6. https://www.youtube.com/watch?v=_8xhqXKtTZs

1.7 Better Query with Meta Theory

In the previous sections, we outlined how behaviorist, purely performance-based approach of assessing AI falls short on multiple fronts: First, we saw in section 1.4.1 that a thorough investigation relies on a good prior understanding of what is difficult and easy for these models. Next, we outlined how AI fails to perform robust generalization (section 1.5), which becomes evident when testing it on formal reasoning problems (section 1.4.2). Last, we saw how excessive testing not only fails to really improve our explanations of these models, but actively hinders a theoretical understanding.

Taken together, these limitations indicate how we can neither know whether AI's performance relies on a superficial and thus brittle understanding of the domain, nor reliably draw consequences for its further development. This leaves us with the challenge of proposing an alternative, theory-based approach to AI. We will derive this by first examining what challenges of modern AI make theory so difficult, then suggest a structured, computational perspective that can drive explanatory progress.

1.7.1 The Bar Set by Marr

To place and understand where theoretical ambitions fail, we will be using a framework from Cognitive Science called *Marr's Levels* (Marr 2010). This posits that any computational system can be understood at different levels of explanation. Since we are viewing both human and artificial intelligent agents as systems performing computations (section 1.1.2), we can investigate them using these levels. The traditional framework consists of three levels (see fig. 1.4). First is the Computational Level, that answers the question: *'What problem is being solved?'*, then comes the Algorithmic Level, that asks: *'How is the problem being solved?'*, while considering both the system's internal representation of the problem, as well as the rules (algorithms) that it applies to them to produce a solution (output) to a problem (input). Finally, the Implementational Level answers: *'How is the algorithm physically implemented?'*. While these levels do not dictate a hierarchical research agenda, the levels of explanation from top to bottom require an increasingly detailed description of the system.

We can now begin to understand the different approaches to AI based on this framework. The methodology of AI implementation in the last couple of years heavily starts from the Implementational Level; the basis for the neural-network approach to AI is based on an implementational principle (more detail will follow in section 3.3). This dynamic is depicted in fig. 1.5. The choice of neural architecture

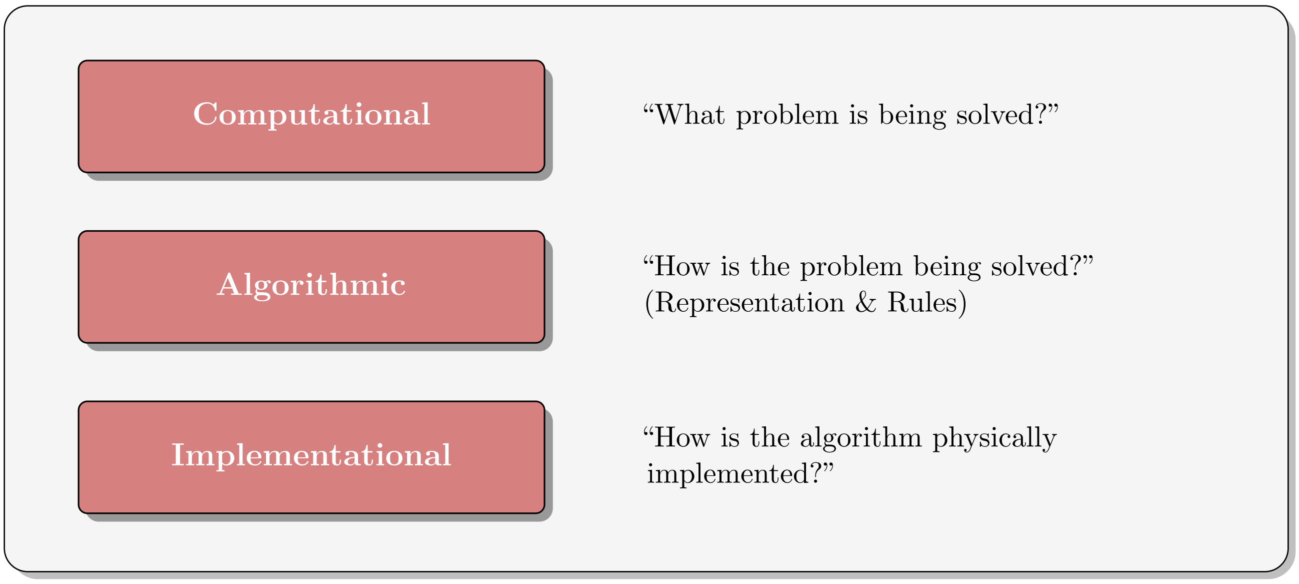

Computational
"What problem is being solved?"
Algorithmic
"How is the problem being solved?" (Representation & Rules)
Implementational
"How is the algorithm physically implemented?"

Figure 1.4. Marr's Levels: The basic framework (Marr 2010).

directly *impacts* the internal representations of domain and reasoning mechanisms of the system on the Algorithmic Level (section 3.3). However, this mapping is not trivial. As we outlined in previous sections, the resultant models are inherently opaque and explanations of how their behavior maps to the real world are for the most part only possible post-hoc (Merullo, Eickhoff, and Pavlick 2024). The interpretability only returns on the Computational Level, where a behaviorist assessment can indicate to us what kind of problems the model solves - and which it cannot. Due to the disconnect between the Computational and the other Levels however, this error signal has no direct consequences - since we do not know what thinking caused the system to err on certain tasks, we do not know how it needs to think differently about them.

This roadblock leads us to the fundamental challenge of finding out how we can connect the levels in a way where we can not only understand all of them but also how they relate. To address this challenge, we will go back to the idea that inspired our application of Marr's Levels to AI research: An interview with Noam Chomsky by Katz (2012). In it, Chomsky posits that contemporary artificial and human intelligence research alike is heavily influenced by the implementational implications of associationism. As we outlined argued above, this leads to a neglect of the Algorithmic Level and ultimately to exclusively behaviorist interpretations of the systems. While he emphasizes that this approach is not useless, he also proposed that it fails to live up to the scientific spirit. He compares the performance-based assessment of intelligence to Physics and states that it is like if you were to "take endless numbers of videotapes [...] and feed them into the biggest and fastest computer, gigabytes of data, and do complex statistical analysis [...] and you'll get some kind of prediction about what's gonna happen outside the window next. In fact, you get a much better prediction than the physics department will ever give. [...] But you won't get the kind of understanding that the sciences have always been aimed at—what you'll get at is an approximation to what's happening."

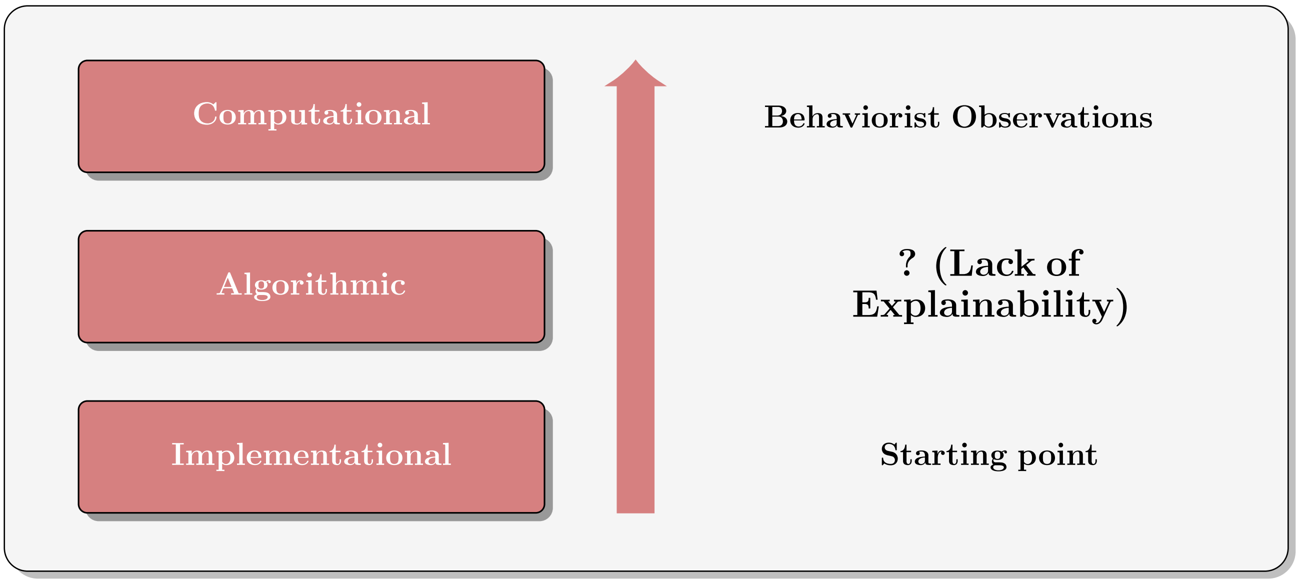


Figure 1.5. Marr's Levels: A performance-based approach to AI. This method starts with an implementational idea, like selecting a neural network architecture. The Algorithmic Level remains opaque. Interpretability only returns on the Computational Level, where we make observations about the performance of the agent on certain tasks, such as "LLMs struggle with formal reasoning tasks".

Thus, he would probably deem much of current AI research unscientific, rendering the first method we outlined insufficient to gain an understanding about an agent's intelligence. Chomsky continues by outlining what a theoretical investigation of computational systems could instead look like, mentioning how Gallistel and Gibbon (2002) apply this principle to understanding neuro-biology. They construct a theory-based understanding of animal brains by addressing Marr's Level hierarchically starting from the Computational Level, as depicted in fig. 1.6: At first one understands the problem the system solves, then the strategy it applies, and lastly, how this strategy is implemented.

While this method provides a good theoretical basis in *theory*, it does not scale to complex computations. Marr (2010) for instance restricted his attention to understanding the comparatively well-defined domains of animal vision. If we want to understand intelligence as a whole, we face another challenge: Devising or proposing algorithms for lots of complex tasks is often infeasible. We can observe this effect in the history of symbolic AI, that, in its most basic form, relies on the manual specification of facts and rules about the problem. Most real-world problems are so intricate and nuanced that this is infeasible[7], which we will argue in more detail in section 3.2. In summary, there is no straightforward mapping from the Computational Level to the Algorithmic Level when the computational system under investigation solves a wide range of complex tasks.

7. For an anecdotal demonstration, see the example of making a Peanut butter and Jelly Sandwich (https://www.youtube.com/watch?v=cDA3_5982h8).

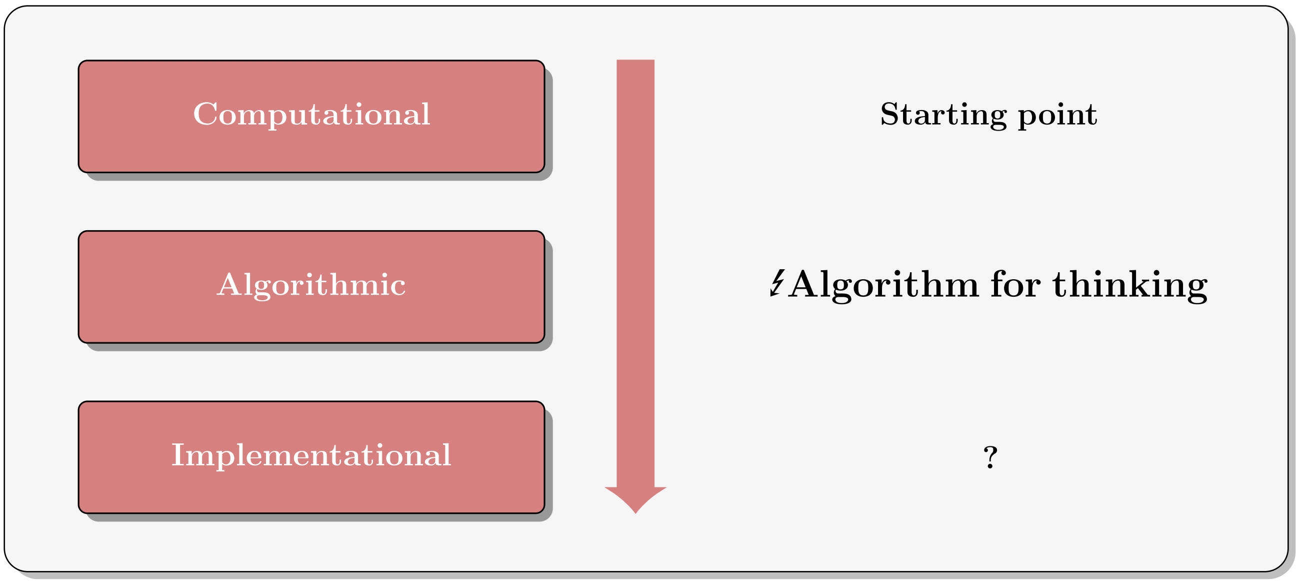


Figure 1.6. Marr's Levels: A theory-based approach to AI. To understand what Cognition is necessary to solve a task, we start from the task itself. The problem we run into: Even for simple problems, describing algorithms that solve them is difficult to impossible.

Now, it seems like the Algorithmic Level forms an insurmountable obstacle, no matter which side we approach it from. Either, we start from the Implementational level and are left with behaviorist observations on the Computational Level, which likely will not lead to real scientific insights. Or, we start from the Computational Level and are overwhelmed by our inability to define how a generally intelligent system should problem-solve. So, do we need to rid ourselves of our scientific ambition altogether and give up on understanding these systems?

Rather than conceding defeat, we instead propose modification of Marr's Levels that tackles the Algorithms challenge by replacing the Algorithmic Level of explanation with a *Meta-Algorithmic Level* (see fig. 1.7). This Level does not describe *specific* algorithms that solve a task but rather describes what *types* of algorithms could solve it. The corresponding question to this Level is: "What characterizes the algorithms that can solve this task?". Our investigation can thus proceed with leveraging the *computational* insight that current AI cannot solve formal reasoning tasks, understand what types of *algorithms* are necessary to solve these problems, and assess the abilities of an AI architecture to *implement* these strategies.

Conclusion and Outlook

In this chapter, we outlined the limitations of behaviorist assessments of AI, outlined what makes obtaining an alternative theoretical understanding so challenging and proposed a novel framework of modified Marr's Levels. In the subsequent chapters, we will propose how AI can be understood on with this scaffolding by employing different theoretical insights. First, chapter 2 introduces Dual Process Theory from Cognitive Science. This will help us connect the Meta-Algorithmic Level to the Computational Level by outlining how different information processing strategies lead to different problem-solving behavior. Next, chapter 3 connects the Meta-Algorithmic to the Implementational Level by demonstrating how these solution strategies are naturally implemented by different AI architectures. Last, chapter 4 will tie all previously established concepts together, which will allow us to understand the whole picture from implementation over meta-algorithms to computation, and how the system's intelligence are governed by the cost-accuracy tradeoff.

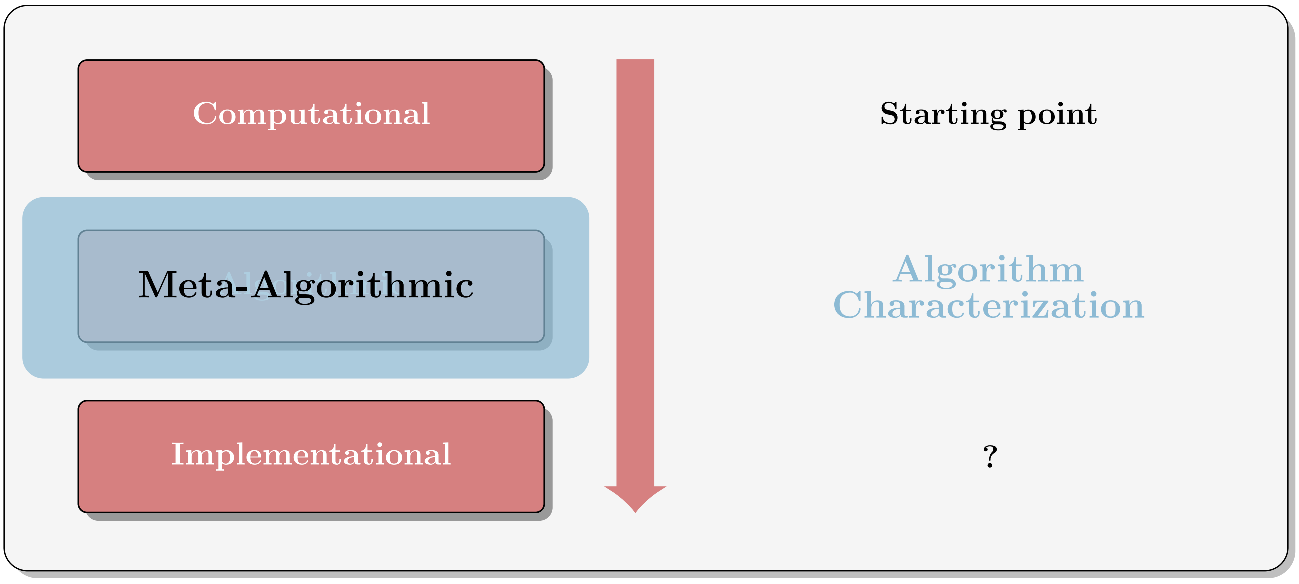


Figure 1.7. Marr's Levels: A theoretical approach to investigating AI with the added 'Meta-algorithmic' level. Given the difficulty of describing precise algorithms that solve a task (see fig. 1.6), we instead characterize the Algorithms that could potentially solve it. This approach leverages insights from the Computational Level, while also establishing more solid criteria for the Implementational Level.

Chapter 2

Dual Processes

There are two kinds of people in the world, those who believe there are two kinds of people in the world and those who don't.

Robert Benchley

As we have seen in the previous chapter, computational systems that are commonly considered AI still struggle when solving certain task types, particularly formal reasoning problems. However, curiously, they seldom behave in a way that is entirely stupid - just not consistently intelligent. We discussed the limitations of a purely behaviorist assessment to identify these types of limitations and proposed investigating artificial intelligence from a theoretical, meta-algorithmic level instead. This allows us to see whether they engage in the correct types of task-solving strategies without having to fully specify or understand the employed strategy itself. In this chapter, we unpack a framework that is repeatedly mentioned in discourse surrounding AI that aims to understand the systems' abilities and limitations from this standpoint. Dual Process Theory (DPT) from Cognitive Science posits that there are two types of strategies that humans rely on in their problem solving - one type offers fast and approximate solutions, the other slow and exact ones.

2.1 AI and DPT - An Intuition about a Theory

To motivate DPT when we have talked about AI so far, we will refer to some mentions of DPT in the AI literature. As mentioned in the introduction, this is mainly the case in high level conceptual debates or the discussion section of AI research papers. For instance, the following:

- “LLMs are best seen as a giant pseudo System 1” (Kambhampati et al. 2024)
- “[T]he human cognitive science of “System 1 + System 2” has been repeatedly mentioned, but the ideas on how to implement it based on large models have been constantly updated, mainly still staying at the stage of drawing on the concept of slow thinking [...] There has been no truly significant and representative work on the theoretical analysis of slow-thinking of LLMs up to now.” (Xu et al. 2025)

System 1 and System 2 thus seem to be capture a salient intuition about the abilities and limitations of current approaches to AI. However, besides the increasing mentions of Dual Process Theory, there does not seem to be solid theory underlying these claims, rendering them a convenient talking point instead. One blog succinctly captures the effect of this dynamic, stating “System 2 is everything AI cannot do yet.” (Paritosh, Wong, and Bollacker 2023). This certainly not a satisfactory conceptualization, so let us try to delve into the Dual Process Theory to derive a more solid definition.

2.2 The Foundations of DPT

First, I will describe the general idea of DPT. Namely, that there are two different kinds of thinking humans can engage in. There are multiple renditions of it that characterize the types differently, so we will again have to find a working definition that applies to our context. In doing so, we will utilize the collection of attributes for both types from Evans and Stanovich (2013) (see original in fig. 5.2). In it, they collected various dichotomies that have been used in iterations of DPT over the years. In modifying it, I condensed the dichotomies as much as possible and will justify my decisions in the text below. Our goal is to pin down the systems to the best of our abilities. A lot of attempts have been made in DPT to condense this list to its conceptual core, but the aim of this work is not to further the theoretical basis of DPT. Instead, we want to see what intuition lies behind its uses in AI. Thus, we will collect the descriptions for now so that we may refer back to the most fitting one whenever we interpret mentions of *System 1* and *System 2* at later points. For now, I have grouped the most relevant attributes into 5 sections thematically. While these sections are not strict and, as we will see, closely related, iterating over them sequentially can help us understand different perspectives on DPT.

Importantly, pinning down a clear definition of the two Systems addresses one of the most common criticisms of DPT itself, namely that it is conceptually vaguely defined. This makes it difficult to pin down precisely what characteristics need to appear in combination to clearly characterize a type of thinking as System 1 or System 2 - does one attribute suffice, for instance, or do all of them need to be present at the same time? Evans and Stanovich (2013) call this the *Clustering Problem.* This fuzzyness makes DPT susceptible to criticism, with many deeming it a messy reinterpretation of empirical data (Keren and Schul 2009; Kruglanski and Gigerenzer 2011). To tackle this issue, Evans and Stanovich (2013) distinguish between *defining features* and *correlated features* and choose thereby choose to put two main dimensions in the center of their discourse. As we will see in the following, other Dual Process Theorists, such as Kahneman (2011) focus on different aspects, which we will also discuss. Further, we will identify that, with AI as our application of DPT and our framing of thinking as computational problem solving, the most relevant angle to us will be the description of the Systems' underlying information processing. We will proceed by discussing the narrowed down, reordered, and modified version of the attribute list in fig. 2.1.

System 1	**System 2**
Evans & Stanovich	
No working memory required	Working memory required
Autonomous	Cognitive decoupling
Kahneman	
Biased responses	Normative responses
Fast	Slow
Resources	
Effortless	Effortful
Parallel	Serial
High capacity	Limited capacity
Processing	
Intuitive	Reflective
Associative	Analytical (rule-based)
Implicit	Explicit
Transduction	Induction

Figure 2.1. Dichotomous attributes used to characterize and distinguish between System 1 and System 2 thinking. Modified (reduced, reordered, rephrased and extended) from Evans and Stanovich 2013 (for the original table, see fig. 5.2). Each section corresponds to a set of characterizations, separated by the main focal points of authors or types. In blue, we highlighted attributes relating to accuracy and cost respectively, in red, we highlighted the central distinction between the Systems in our application.

2.3 Characterizing the Systems

2.3.1 *Evans & Stanovich*

We will start with the examination of the two characteristics Evans and Stanovich (2013) accentuated as their defining attributes. As outlined above, the objective of this paper was to clearly identify the defining characteristics of the Systems, and render all other ones correlated: Statistically, these often appear in conjunction but they do not have to. Again, this paper only dealt with *human* intelligence, so we need to keep in mind that the distinction and prioritization of the attributes might not be the same for *artificial* intelligence.

Working Memory. This is especially important to keep in mind for the first distinction of **Working Memory Use**. A prominent factor that makes Dual Process Theory hard to prove is the non-triviality of clearly attributing externalized human behavior to either System. As Dual Process Theory states, there are two types of thinking that a singular reasoner can employ. Since a lot of the features are only correlated and not defining (as discussed above, System 2 can be faster in some instances), we cannot distinguish which System caused which response from observing behavior alone. This simultaneous presence of both Systems poses a fundamental obstacle for empirical investigation of Dual Process Theory in human subjects. Our only means of investigating each System by itself is **empirical isolation**: By creating an experimental setup so that they can only use on System to respond, we can clearly attribute the behavior to this type of thinking and establish a refined distinction. One of the most reliable ways to clearly separate the Systems is through working memory manipulations: When subjecting the human participant to a high working memory load, we can observe that their System 1 responses are consistent with normal circumstances, while their System 2 abilities are impaired (e.g De Neys and Glumicic (2008)). This indicates that working memory resources are needed for the execution of System 2 strategies but not for System 1.

Control. Second, they outline the differences in **control** the agent has over the execution of the strategies. System 1 is triggered *automatically* by stimuli in the agent's environment and executed *autonomously* (K. Stanovich 2011). Conversely, System 2 is activated *deliberately* under control by the agent and can *abstract* away from the stimulus-response cycle; which Evans and Stanovich (2013) call *cognitive decoupling.* They deem this System 2 ability as crucial in order to engage in *mental simulation* and playing out *hypotheticals*: "In order to reason hypothetically, we must be able to prevent our representations of the real world from becoming confused with representations of imaginary situations." (Evans and Stanovich 2013). The authors deem this decoupling a crucial prerequisite for counterfactual reasoning, which involves the mental manipulation of abstract symbols to simulate scenarios in our mind. Judea Pearl, a prominent voice in the reasoning literature, has long argued that engaging in these counterfactual simulations is central to human intelligence (e.g. Pearl (2009)). His model of causal hierarchy delineates three levels of understanding the world: On the lowest level, *associations* allow us to understand that two events are related when we conjointly observe them. To achieve better understanding, we need abstraction to mentally intervene and simulate deviations from the observed scenarios and ask *'What if?'* questions, which allow us to grasp *causal dependencies.* Once we have mastered Causality, we can engage in retrospection and imagination, enabling us to understand *'Why?'* (Pearl 2019)[1].

The automatic response of System 1 and deliberate interjection of System 2 raises another key question: How does an agent decide when to *switch*? Evans and Stanovich (2013) answer this question with the idea of *default-interventionism.* According to this account, the switch to System 2 is triggered by factors in the environment (section 2.5) that indicate System 1 responses might not be accurate enough to solve the problem at hand.

2.3.2 *Kahneman*

Accuracy. Daniel Kahneman is another central figure in the field of Dual Process Theory. A lot of his work was focused around using System 1 and System 2 to explain why humans give faulty answers to reasoning questions. This literature on *heuristics and biases* explains how people rely on strategies that provide approximate or 'good enough' answers in some circumstances but lead to erroneous responses in others. While determining what a 'correct' answer is in reasoning problems is often not trivial, the reasoning literature often refers to **normative** answers instead (this terminology was also used by Evans and Stanovich (2013) in their table). Giving a normative answer to the problems often relies on employing logical reasoning dependent on the *structure* of the problem (like the famous Linda Problem (Tversky and Kahneman 1983)), while a biased answer results from a heuristic dependent on the *content* of the task. Importantly, providing an answer by employing a heuristic does not necessarily lead to a faulty response - by definition, a heuristic produces approximately accurate results. However, its success can deteriorate depending on the task context. This effect is best exemplified by an experimental setup by (Frederick 2005), who introduced the *Cognitive Reflection Test* (CRT), that contains questions that have intuitive but false answers (which System 1 falls victim to), that require more reflective thinking to solve correctly (System 2). Kahneman and Frederick (2002) interpret these results as an effect of *attribute substitution*, where humans, when confronted with a difficult task, often answer a related, easier task which System 1 answers. Now, one might wonder why humans would

1. Causality and Counterfactuals are a big research area in the field of AI at the moment. While this debate is incredibly interesting and worth looking into relatedly and independently of what is discussed here, engaging further with it would explode the scope of this paper. We will only remark that the necessary abilities to gain this level of understanding are System 2 abilities. This has interesting implications for Deep-Learning based AI, which we have previously seen compared to System 1.

engage in such an error-prone strategy if they are capable of executing a better one using their System 2. The answer is simple: Heuristics are cheap (and humans are lazy). Some of the earliest work on DPT had the cost-accuracy tradeoff at its very core and explained how humans employ heuristics to give approximate but cheap answers (Tversky and Kahneman 1974).

Speed. The cost-perspective of DPT became central after the release of Kahneman's popular science book named after it: *'Thinking, fast and slow'* Kahneman (2011). Whenever DPT is mentioned in Artificial Intelligence research papers, this book is the most frequently (and almost exclusively) cited reference. Consequently, the most salient and well-known attributes of these systems are their processing speeds, as reflected in the quote from Xu et al. (2025) (section 2.1). The authors highlight the lack of theory that maps DPT's to AI, effecting a lack of clear vision following from this intuition. It is plausible that this lack of actionable steps comes from not looking beyond the surface of the fast-and-slow characterization: While fast and slow thinking do not have direct consequences for implementation of artificial intelligence, we should aim to understand *why* these processes are distinguishable by their processing speed (and, in doing so, demonstrate our human counterfactual thinking abilities).

2.3.3 *Resources*

To understand how resource metrics relate to the characteristics we have discussed so far, we will now explore two more concepts that are central to Kahneman's conception of Dual Process Theory: Attention and Effort (Kahneman 1973).

Effort. According to Kahneman, two main factors make System 2 operations more *effortful* than System 1 operations. These are first the *load on working memory* that System 2 requires, which we have outlined previously: System 2 operations require cognitive decoupling - we abstract away from our immediate experience and instead perform operations on mental items that we hold in our working memory. Not only is this abstraction by itself effortful but also the operations we apply to it. That is, applying a heuristic requires one step only - the answer comes to us immediately from perceiving the task. On the other hand, reflection can take multiple reasoning steps.

Attention. Remember from before that System 2 requires deliberative control, whereas System 1 is autonomous. This means System 2 processes require conscious *attention.* Since attention is singular, computations must be performed serially. On the other hand, System 1 can execute tasks on multiple channels at the same time in parallel.

Capacity. As a result of the high effort and limited channel of System 2, its capacity is much lower than that of System 1. Consequently, a reasoner with a limited processing rate (i.e. any real world reasoner), is *fast* when executing System 1 strategies and *slow* for System 2 respectively.

Still, these terms of *attention*, *effort* and *speed* might seem vaguely defined - so let us examine some empirical results to see how they manifest in behavior and thus become measurable. Examining the Systems empirically requires their empirical isolation. That is, if we want to attribute a behavior to System 1 for instance, we must make sure System 2 could not have interfered. One method to achieve this is to inhibit System 2 by exploiting its serial and singular nature (Evans and Stanovich 2013): If it is busy with one task already, any other task that is executed in parallel must be solved using System 1. Second, we can measure pupil dilation relative to the the task. It is well-known that pupil dilation correlates heavily with *cognitive load* and can thus be a good metric of effort. We can measure that with increasing number size or amount and therefore demands on System 2 computation, the effort increases (Kahneman 2011). Lastly, introducing time-pressure when solving tasks that require System 2 also significantly increases System 1 behavior, indicating the limited capacity and effortful nature of System 2 processes, effecting slow solution speed (Evans and Stanovich 2013).

2.3.4 *Processing*

The previous characterization of the Systems relied on many references to human cognition in particular. Yet, we set out to derive functionalist definitions of intelligence and its correlates, and have previously outlined how this requires breaking down terminology to its information-theoretic core section 1.1.3. Consequently, understanding what kind of Information Processing the Systems partake in is crucial to us. And as we will see in the subsequent Chapters, taking the following dichotomies as our *defining distinction* will naturally bring forth all other correlated attributes. In the following, we will move through distinctions between these processing styles on different levels of technicality.

Intuition. When referring to System 1 operations, we will often be calling them *intuitions*, as opposed to *reflections* performed by System 2. These merely serve as tools for the (human?) reader to easily understand on a superficial level how the Systems work and should not be taken literally. Otherwise, we risk attributing any agent that performs these operations with anthropocentric features - something we set out to avoid in the beginning. We will thus only explore why this dichotomy describes the Systems.

Besides the previously outlined differences between the Systems that align well with how we think of intuition and reflection, there is also another interesting conjecture of these descriptions. Namely, that intuition need not be a mere gut-feeling that is shared equally across humans as a result of an evolutionary adaptation, for instance be the natural instinct to run away upon the sight of a Tiger. Instead, intuition is *learned*, which results in interpersonal differences. For one, intuition can be trained by mere exposure: Someone who buys candy bars more often and looks at the nutritional labels from time to time will develop a better intuition for which candy bar has the most calories just from looking at the packaging. Alternatively, another vast part of System 1 mechanisms rely on habituations - processes that were at first deliberate and effortful (executed by System 2) but over time became autonomous (automatically executed by System 1). Kahneman (2011) refers to the example of of driving a car: When we first learn to drive, every move requires conscious effort and attention. After a while, the driving itself becomes second nature and we can easily use our limited System 2 capacity to converse with the passenger instead. Consequently, an agent can form an 'educated intuition' based on frequent exposure to a task. What also becomes apparent is that this intuition might not be reliably formed without (temporary) access to a System 2.

Processing Types. Next, forming and executing an intuition is purely *associative*: Through repeated exposure to two events, the reasoner learns that these are related. On the other hand, *analytical* thought requires a deeper understanding, which, as we outlined above, can only be formed by counterfactual thinking, which in turn requires cognitive decoupling.

In the original table, Evans and Stanovich (2013) called the System 2 processing *rule-based*, while in the same breath dissecting why this terminology is misleading, thus we supplemented this description with *analytical*. It is, however, useful to delve into why *rule-based* is misleading - and, when misunderstood, one of the biggest points of critique against DPT. Kruglanski and Gigerenzer (2011) for instance deliberate whether: If System 2 is characterized as rule-based to contrast it with System 1, does this imply System 1 is not rule-based? If this were the case, System 1 could not be implemented on a computational system (which is fundamentally rule-based). Consequently, this interpretation of Dual Process Theory would directly oppose Computationalism.

However, Evans (2006) clears up that this is not the intended interpretation of rule-based and states that "[r]ules can be concrete as well as abstract and any automatic cognitive system that can be modeled computationally can in some sense be described as following rules.". This further distinction between *'concrete'* and *'abstract'* rules is still a bit ominous. Fortunately, Evans elaborates in follow-up work: "Associative processing can, of course, be modeled by neural networks that are implemented using rules. However, these 'rules' are not what people generally mean when they refer to Type 2 rule-based processing." (Evans and Stanovich 2013). Combining this with the tools of Marr's Levels (fig. 1.4), we can see how both argue past each other: Dual Process Theory characterizes types Algorithms (hence its positioning on the Meta-Algorithmic Level), while Kruglanski and Gigerenzer (2011) argue about rules on the **Implementational Level** - both parties agree on Computationalism.

Representations. As promised, we will now try to understand what Evans (2006) meant by *concrete* and *abstract* rules. We will preserve the intended interpretation but instead use a different terminology for clarification, namely *implicit* and *explicit* processing. Kahneman later stated in the *AAAI-20 Fireside Chat with Daniel Kahneman* (2020) that he believes this distinction to be the most relevant one when applying the Dual Process framework to Artificial Intelligence.

Before we pin this notion down even further, it is worth mentioning that there is also a big debate about Knowledge Representation in AI, that deals with implicit (continuous/ distributed) and explicit (discrete) methods of internal representations. The former underlies a Neural Network's representation, while the latter underlies a Symbolic System's Representation, which we will revisit in chapter 3. Dual Process Theory also has the distinction between implicit and explicit knowledge (see fig. 5.2). However, the debate about underlying representations in again incredibly extensive, in both AI (Neurons vs Symbols) and Cognitive Science (e.g. Language of Thought) and a detailed discussion goes beyond the scope of this Thesis. This exploration should proceed in future work.

While it might seem natural to group explicit/implicit processing and representation together as one big distinction, renowned AI researcher Yoshua Bengio discusses how *processing* symbolic content is different from *relying* on symbolic representations (*AAAI-20 Fireside Chat with Daniel Kahneman* 2020). In other words, reasoning *about* symbols is different from reasoning *with* symbols. He gives the example of image processing and generation: A neural network, that does not have discrete internal representations, can learn how to process or generate images that are made of discrete pixels. While the discussion does not reach a consensus, all parties agreed that at least it *seems* like System 2 does *something like* symbol manipulation based on the type of operations it performs (c.f. Garcez and Lamb (2020)). The question of symbol processing is not exclusive to AI and has a solid place in the philosophy of human cognition, where researchers have long disputed whether the mind operates based on (something like) symbols (Fodor and Pylyshyn 1988; Marcus 2022). We will not delve into this debate further and merely direct our attention towards implicit and explicit information processing.

After delving into the characterization of the two Systems of DPT, we approach the central motive of our complexity analysis (chapter 4). We have System 1, that takes little effort but may provide less accurate results, while System 2 requires more resources to provide more accurate results. Further, we we already got a first intuition about how these characteristics emerge from the fundamental type of information processing that underlies the Systems. This complementary nature of the Systems with respect to the cost-accuracy tradeoff will play a central role in understanding why only their combination enables an agent to dynamically respond to environments that require different pareto-optimal strategies - rendering Dual Process interaction a necessity for intelligent behavior.

2.4 Patterns and Structures as Modes of Generalization

We have now identified that the information processing types of System 1 and System 2 are the central characterization of Dual Process Theory for this work. This means that System 1 and System 2 are *defined* by these processing types. System 1 performs *implicit processing*, while System 2 performs *explicit* processing. The remainder of the attributes that describe are deemed *correlated* features, in line with the goal of disambiguating the definition of DPT set by Evans and Stanovich (2013). In chapter 3, we will see how these attributes naturally emerge from their respective processing types, providing further empirical evidence for DPT by computational modeling.

To understand explicit and implicit processing, we will formalize these descriptions with terminology from Computer Science. Explicit and implicit processing describe different kinds of **inferences**; The methods by which a system obtains a new result by relying on prior insights. In the following, we will discuss two types of inference *induction* and *transduction*, and outline how they capture System 1 and System 2 processing.

2.4.1 Explicit Processing is Induction

Induction described the process of extracting *general rules* from *specific instances*. An inductive reasoner seeks to capture regular properties of the domain based on data it observes and formulate an *explicit* explanation for it that allows *generalization* to potentially unseen data points. This method thus aligns with our characterization of System 2 thinking as explicit processing. Induction's counterpart is *deduction*, that this takes the known *universal* rules and arrives at *particular* data. Induction and deduction capture explicit, rule-based reasoning, which will become important in chapter 3.

2.4.2 Implicit Processing is Transduction

Transduction follows the same objective as induction, in that it generalizes from specific data to a general representation. However, this representation is implicit and not captured by a rule. This concept is slightly less intuitive but will become clearer throughout the course of this work; mainly when we talk about concrete implementations of this strategy in section 3.3. The concept stems from the field of Computational Learning Theory, where Vapnik (1995) defined it as "estimating the *values* of the function at particular points of interest given the observations", as opposed to induction, that "requires one to find a *function* given particular data". For now, we highlight how this captures reasoning based off intuitions (from a more phenomenological standpoint), which aligns with the implicit processing of System 1.

2.4.3 Illustrating Induction and Transduction

In the following, we will illustrate this difference between induction and transduction by going back to the example of addition from the introduction. Suppose the reasoner is presented with the following inference task: Given the data $\{((1,1),2)$, $((1,2),3)$, $((9,5),14)$, $((8,4),12)\}$, what solve $((385,493),?)$. A transductive model might spit out 878 directly, but it could also answer 892 without further elaboration. Inductive reasoning would instead produce a rule to explain the input, like $s = x + y$. Just by looking at this rule, we can verify that it understood the problem by generalizing correctly - and if not, provide counterexamples to the rule to prompt a revision. Further, this method is perfectly explainable and we know that this behavior will scales robustly to any other addition problem that we throw at the system.

From these examination of accuracy and explainability alone, it seems like having an inductive learner would always be preferable. So why would we ever *choose* a transductive generalization instead? The answer is simple - it is cheaper. In line with the effortless nature of System 1 processes, reasoning based on an intuition is *less hard* than analytical reasoning. We will discuss this more formally in chapter 4 - for now, we refer to an imperative devised by the creator of transduction:

> "When solving a problem of interest, do not solve a more general problem as an intermediate step. Try to get the answer that you really need but not a more general one. According to this imperative [...] Do not estimate a function if you only need to estimate its values at given points. (Try to perform direct inference rather than induction.)"
> - Vapnik (2006), p. 477

Summing up, in a complex world, finding and understanding a general rule of that explains how things work is much harder than finding a model that merely makes statistically good predictions. If a good approximation suffices to solve a task, Vapnik (2006) thus instructs us to opt for this instead of an exact explanation. In the next section, we will outline how the explicit and implicit representations of induction and transduction respectively impact what the model understands about the domain.

2.4.4 Transduction Learns Patterns, Induction Learns Structures

We have now defined System 1 and System 2 by induction and transduction as their processing types. Now, we discuss what kinds of generalized representations of the domain these methods support. Induction learns an explicit rule from logical regularities in the data and thus aims to capture the domain's underlying *structure.* On the other hand, transduction identifies regularities in the form of statistical correlations in the data and aims to make a good prediction for similar instances using patterns. This method's accuracy is only assessed by its empirical performance - it generalizes based on increasingly accurate *pattern-approximation.* Induction and transduction thus differ in the ambition: When learning a pattern, it suffices to pick up on local regularities in the observed data. A pattern counts as *accurate*, when it generalizes well within a given context; Capturing the spirit of a *heuristic.* A structure on the other hand aims to identify rules that govern the *whole* domain, independent of context (also referred to as *out-of-distribution generalization*).

Relating this back to System 1 and System 2, we can now see how this type of reasoning naturally leads to the correlated attributes of *biased* and *normative* responses. These do not occur because System 1 performs *faulty* reasoning - instead, it is an effect of a non-robust approximation, that is accurate enough to give a right answer in some context but fails in others (Evans and Stanovich 2013). In the next section, we will outline the factors that determine what these contexts are - when does a good pattern-based approximation suffice and when is structure necessary? DPT provides an explanation based on the distinction based on characteristics of the environment.

2.5 The Environmental Impact on System 1 Accuracy

In the previous section, we explained how transductive reasoning based on patterns can present an accurate method of generalizing from particular data instances. This implies that System 1 reasoning *is* reliable in some tasks environments and not others. At first, it might seem strange that not the type of processing or the task itself determines the accuracy of the reasoner, but that the environment instead plays such a crucial role. However, this factor plays a crucial role for the quality of the data a reasoner observes, which influences the cost-accuracy tradeoff of generalization (we will analyze this more formally in chapter 4). As Herbert Simon, one of the founding figures of bounded rationality put it:

"If we wish to know what form gelatin will take when it solidifies, we do not study the gelatin; we study the shape of the mold in which we are going to pour it."
- A. Simon (1990)

A central concept in DPT is the distinction between two types of environments: A *benign* environment is one in which System 1 can learn and reason with high accuracy, while a *hostile* environment lacks the necessary preconditions: the presence of stable feedback cues and the opportunity to repeatedly sample them (Kahneman and Klein 2009).[2]. A benign environment allows System 1 to develop increasingly accurate *expert intuitions* over time, as in the case of chess. Conversely, a hostile environment, like stock trading, lacks this stable feedback signal and thus make it impossible to obtain a good intuition (Kahneman and Klein 2009).

Let us briefly illustrate this distinction with our candy-bar example from the beginning (section 1.1.5). Strategy A compared nutrition labels, while strategy C relied on the statistical relation of the calories and the packaging. A reasoner relying on the latter could have internalized a pattern that relies on the amount of red in the packaging, which is correlated with the calorie count for this selection of candy bars. When this strategy is applied in a different supermarket, A will still achieve 100% accuracy, while the success of C can decline arbitrarily if the first supermarket was a hostile environment, where the feedback cues failed to represent the general domain. Interestingly, environmental hostility impacts the accuracy of an intuition but not the confidence of the answer (Kahneman (2011), Ch. 20). System 1 seems to lack the meta-knowledge about its reasoning - knowing what we do not know or cannot know is a System 2 ability (section 2.3.1).

Conclusion and Outlook

This chapter equipped us with a first Meta-Algorithmic perspective in the form of DPT, that distinguishes two types of thinking. We argued that these types of thinking are characterized by their

2. This conceptual divide appears under different names in the reasoning literature: While Evans and Stanovich (2013) distinguish between *benign/hostile*, Kahneman and Klein (2009) call the environments *low-validity/high-validity*, while Hogarth (2001) refers to them as them *kind/wicked*.

implicit (System 1) and explicit (System 2) processing types. Further, System 1 captures approximate and cheap reasoning, while System 2 problem solving is exact but costly. We also gained insights about the impact of the environment on the accuracy of the processing types. In the next chapter, we will show how these processing types are naturally implemented by two dominant AI paradigms: Neural networks and symbolic systems. This exploration will demonstrate how the remaining attributes of the Systems correlate with their processing types. Further, ground the Meta-Algorithmic in the Implementational Level and qualify AI as computational models for the investigation of Cognitive Science theory, aligned with the call of action from van Rooij et al. (2024).

Chapter 3

Implementations of Inferences

Logic is the beginning of wisdom, not the end.

Mr. Spock
Star Trek VI: The Undiscovered Country

In the previous chapter, we introduced Dual Process Theory as a framework that distinguishes between two kinds of thinking (chapter 2). We identified these processing types as the transductive System and the inductive System 2. In this Chapter, we will connect this framework to AI. The resulting bridge will connect the Meta-Algorithmic to the Implementational Level (fig. 1.7) and lay the basis for chapter 4, which relies the mapping of neural networks and System 1, as well as symbolic systems and System 2.

3.1 Learning and Reasoning

In chapter 2, we introduced transduction and induction as the different types of inferences System 1 and System 2 use to obtain a general understanding from particular data. While our exploration of DPT mainly focused on their *reasoning* behavior, we have also encountered *learning*, for instance when discussing how intuitions are formed section 2.5. We have not yet made conceptually clear what differentiates these, which will now follow. Fittingly, Valiant (2003) identifies "the ability to learn from experience, and the ability to reason from what has been learned" as the "two most fundamental aspects of intelligent cognitive behavior".

3.1.1 Inference Breakdowns

This opens up a new dimension of thinking and leaves us with four distinct types, visualized in fig. 3.1: The first dimension of System 1 and System 2 describes implicit processing and explicit processing. The second dimension differentiates learning and reasoning. We can now distinguish four qualitatively different types of thinking: α describes acquiring understanding by internalizing patterns in the data, γ making heuristic predictions on this basis, β consists of learning a rule that describes the data and δ applies the rule during to new data.

	System 1	**System 2**
Learning (Acquisition)	α	β
Reasoning (Application)	γ	δ

Figure 3.1. Conceptual Matrix distinguishing four types of thinking. The first dimension of System 1 and System 2 distinguishes between implicit processing (transduction) and explicit processing (induction and deduction). The second dimension differentiates learning (acquiring understanding) and reasoning (applying understanding).

On this basis, let us now pin down what *understanding* means in this context.

3.1.2 Learning and Reasoning as Acquiring and Applying Understanding

In our terminology, we defined *understanding* of a concept as possessing an internal representation that enables correct reasoning based on it (section 1.1.4). This means that, in order to understand a concept, the agent has to learn an explanation for the data that allows it to make accurate predictions about data it encounters in the future. Valiant (2003) refers to this explanation as "what has been learned" in the quote above - defining understanding captures this "what" more tangibly (c.f. Mitchell (2023)). This improved grasp on what lies between learning and reasoning also allows us to better understand the System 1 and System 2 characterization of implicit and explicit representations (fig. 2.1): The transductive understanding of System 1 is implicitly internally represented and in a way embodied by the system itself (we will demystify this in section 3.3), while the inductive learning of System 2 leads to an internal representation in the form of an explicit rule.

A potential conceptual confusion might arise at this point: Since reasoning equates to thinking (section 1.1.3) and learning and reasoning are differentiable, is learning not thinking? This is not the case, and unfortunately, another terminological victim of the *'reasoning'* polysemy. When we refer to the *learning and reasoning*, we are strictly speaking differentiation between reasoning *to* an explanation and reasoning *from* an explanation, a distinction from Rationality literature. Hence, learning is a form of reasoning and consequently also a type of thinking. We will additionally spell out reasoning *from* an explanation subsequently whenever this clarification is necessary.

Having established this, let us now proceed with the exploration of different kinds of AI implementations.

3.2 Symbolic Systems

3.2.1 Principles and Examples

Symbolic AI was founded based on a direct conjecture from Computationalism (section 1.1.2): If human thinking can, in its entirety be viewed as a from of computation, a computer should be able to

α	$\boxed{\beta}$
γ	$\boxed{\delta}$

implement human-level intelligence if we encode human knowledge into its system. Following our previous outline of internal representations and types of thinking, this approach represents providing a system with all the explicit understanding it needs to reason about a domain, after which it performs deductive reasoning to solve problems. This assumption of explicitness is captured by the *Physical System Hypothesis (PSSH)*: “A physical symbol system has the necessary and sufficient means for general intelligent action.” (Newell and Simon 1976). This approach was central for the Dartmouth Workshop, which is commonly referred to as the birthplace of AI. In their initial proposal for this event, four of the most influential information scientist of their time (John McCarthy, Marvin L. Minsky, Nathaniel Rochester, Claude E. Shannon) posit that “every aspect [...] of intelligence can in principle be so precisely described that a machine can be made to simulate it.” (McCarthy, J. et al. 1955).

In practice, symbolic AI thus relied on the manual encoding of *facts* and *rules* that explained a domain into symbolic representations. For instance, objects into the real world could be represented by a symbol (like *apple*, *Newton*, *tree*), along with facts and rules of inference. Reasoning from this explanation thus looks like the following: From *apple - falls from - tree* and *Newton - is under - tree*, the reasoner infers *apple - falls on - Newton.*

Examples of early AI based on these implementational principles include:

(1) Automated Reasoning (1950s). Among the first forms of symbolic AI were *automated reasoning systems* (Crevier 1995). The *Logic Theorist* for instance, independently found proofs for Mathematical Theorems based only on basic mathematical expressions (Allen Newell and Herbert A. Simon 1956).

(2) Expert Systems (1970s). The jump less abstract application was made with the introduction of *expert systems*, that deduced from hand-crafted rules describing specific domains (Buchanan and Shortliffe 1984). An example of this is *MYCIN*, a system that predicted medical diagnoses based on 600 *if-then rules* encoded by human experts.

(3) Games (1990s). Games like chess frequently appear in as AI benchmarks and lend themselves naturally to symbolic reasoning due to their fully and explicitly specified domain and combinatorial logic. The chess computer *Deep Blue* leveraged large amounts of computational power and rule-based heuristics from expert players to assess around 200 million board positions per second, famously defeating then world champion Garry Kasparov in a 1997 match (Campbell, Hoane, and Hsu 2002).

3.2.2 Strengths and Limitations

From the above it becomes clear that these symbolic AI systems have an explicit internal representation of their domains of application and derive new conclusions through the application of logical rules (induction and deduction). This nature emerges their greatest strengths: They are robust, and every reasoning step is is exact and fully explainable - all the desiderata we missed when discussing the limitations of contemporary AI in previous chapters. However, this stability comes at a cost: Deduction is computationally expensive. The strategy of *Deep Blue* for instance comes close to brute-forcing through a combinatorially huge search space (Campbell, Hoane, and Hsu 2002) and even just extending this strategy to more complex games like *Go* would require an infeasible amount of computational power (we will formalize this in section 4.3).

Further, as we saw in the *expert systems*, the understanding of (these basic) symbolic systems relies on manual explicit encoding of human knowledge, the system knows precisely what it is told. This leads to the *Knowledge Engineering Bottleneck*: Exhaustively specifying real-world problems at scale is infeasible. The lack of generalization abilities stems from a fundamental problem with of symbols: The *Symbol Grounding Problem* (Harnad 1990) describes how symbols have no intrinsic meaning, thus there is no straightforward way to know what real-world concept they capture or, resultantly, whether other concepts are similarity to this one. This limitation with learning also effects the immobility of their knowledge. Humans, for instance can dynamically adjust their understanding on the fly. For instance, upon learning that *'penguins are birds'*, we can refine our previous conviction that *'all birds can fly'*.

3.2.3 Symbolic Systems as the System 2 of AI

In summary, symbolic systems can reason reliably and accurately but are expensive. Further, they fundamentally limited learners and fail to dynamically generalize their knowledge across related domains. This exploration demonstrates how, starting from a symbolic implementation follows an explicit analytical processing style (induction and deduction), which emerges System 2 reasoning characteristics (fig. 2.1). Further, it shed light on the difficulties of learning this System's internal representations.

3.3 Neural Networks

3.3.1 Principles and Examples

$\boxed{\alpha}$	β
$\boxed{\gamma}$	δ

We have seen above how symbolic systems started from the idea of encoding explicit knowledge into a computer. The neural network based approach followed a contrasting route. Based on the biological insights that human brains consists of large clusters of neurons that adapt their connections flexibly, suggesting that knowledge emerges from mere exposure to data. This idea is grounded in Hebb's principle: "Neurons that fire together, wire together" (Hebb 1949). This principle of *Associationism*, or *Connectionism* in AI applications, bypasses explicit encodings of structures and instead learns patterns that minimize the statistical error of their predictions - the essence of transduction (Vapnik 1995). Based on every new data point that they observe, Neural networks adjust the weight of the neural connections to minimize the *average error* of their predictions compared to the correct solution.

The internal representation of this knowledge, as a result of this architecture and learning type, is also implicit and not explainable. Concepts like *apple* are captured by high-dimensional vectors that are shaped by their *context*, such as the statistical correlation with other concepts like *red* and *stem*. To understand this implementation better, let us again consider some representative historical applications:

(1) Machine Learning (1950s). The first and most basic implementation of a neural-network-like structure was the *Perceptron* (Rosenblatt 1958). With only a single layer of neurons, it could capture concepts like *AND* and *OR* from data, allowing a first meaningful step towards *Machine Learning*. However, the initial enthusiasm about this discovery was quickly met with results about fundamental limitations, such as the inability to capture other simple functions like *XOR* (Minsky and Papert 1969).

(2) Training methods (1980s). The development of new training methods like *backpropagation* offered a way to train deeper networks, thereby surpassing some of the previous limitations (Rumelhart, Hintont, and Williams 1986).

(3) Deep Learning (2010s). The true potential of this approach revealed itself with increasing computational resources, that allowed scaling up the neural architectures to new dimensions. *Deep Learning* established itself as the dominant AI paradigm following unmatched performance on text and image processing (Krizhevsky, Sutskever, and Hinton 2012; LeCun, Bengio, and Hinton 2015)).

(4) Large Language Models (2020s). This trajectory culminates with the release of Large Language Models, where the introduction of the transformer architecture (Vaswani et al. 2017) combined with training on enormous language data corpora set new AI performance records (Devlin et al. 2019; T. B. Brown et al. 2020).

3.3.2 Strengths and Limitations

This empirical success of neural-network based approaches confirms the promise of *emergent abilities* (Wei, Tay, et al. 2022): By mere exposure to vast amounts of data, these architectures are able to internalize an understanding of the world that achieves great performance in some domains of application. Their distributed internal representations enable a more dynamic and contextual representation of the world. Consequently, these models do not share the challenges symbolic systems face in learning.

Yet, the nature of these abilities and underlying knowledge should be investigated through a skeptical lens (Schaeffer, Miranda, and Koyejo 2023). Their remarkable performance restricts itself to *'fuzzy'* tasks, that rely more on approximation than precise solutions and remains brittle on others, like formal reasoning problems (section 1.5). Further, their implicit processing styles and representations are not fundamentally explainable - there is no way to directly infer what these models know or how they think.

3.3.3 Neural Networks as the System 1 of AI

Having outlined the mechanisms behind neural network based AI implementations, which are fundamentally transductive reasoners, we can now understand why they are often compared to System 1 (Kambhampati et al. 2024). Their method of understanding the world and making predictions based on this aligns with a human intuition, that forms over time and provides quick, heuristic-based judgments - that sometime provide good approximations and sometimes miss the mark fig. 2.1. This parallel extends beyond superficial characteristics, and we can identify research topics in contemporary AI that congruently fit our previous description of patterns and structures (section 2.4.4), as well as hostile environments (section 2.5):

- **Shortcut Learning.** Similar to how we characterized System 1 as relying on patterns rather than structural insights (section 2.4.4), neural networks often pick up on superficial patterns in their training data, like we saw in the section 1.5.1 classifier (Ribeiro, Singh, and Guestrin 2016). This phenomenon is referred to as *shortcut learning* (Geirhos et al. 2020; Lapuschkin et al. 2019), also evident in state-of-the-art LLMs (Yuan et al. 2024; Song et al. 2024).
- **Adversarial susceptibility.** Neural networks are highly susceptible to *adversarial attacks*, that

exploit their reliance on patterns (Moosavi-Dezfooli et al. 2017). This is similar to how settings like the *Cognitive Reflection Test*, that intentionally tricks System 1 with intuitive but false answers (Frederick 2005).

- **Hallucinations.** Like System 1's *expert intuition*, that is confident independent of its accuracy (Kahneman and Klein 2009), LLMs *hallucinate* - they state wrong information as fact and cannot differentiate between justified and ungrounded knowledge (Perković, Drobnjak, and Botički 2024).

Conclusion: DPT Grounded in AI Implementation

In this chapter, we linked System 1 and System 2 to the distinct AI implementations of neural networks and symbolic systems based on their information processing types. This demonstrated how other System characteristics emerge from the implementation directly. Neural networks internalize implicit patterns to achieve fast and approximate solutions, while lacking explainability and robustness section 3.3.3. Symbolic Systems are explicit reasoners that display robust and explainable performance but fail to learn and generalize independently (section 3.2.3). This demonstrates a new dimension to these Systems; They not only complement each other in their fast-and-approximate and slow-and-exact reasoning *from* an explanation but also in generalization and robustness of reasoning *to* and explanation section 3.1. Using our matrix of thinking fig. 3.1, we can summarize this new dynamic:

$$\text{Strengths: } \begin{array}{|c|c|}\hline \boxed{\alpha} & \beta \\ \hline \gamma & \boxed{\delta} \\ \hline \end{array} \qquad \text{Challenges: } \begin{array}{|c|c|}\hline \alpha & \boxed{\beta} \\ \hline \boxed{\gamma} & \delta \\ \hline \end{array}$$

Aligned with these findings, Garcez and Lamb (2020) state that "[i]t is now accepted that **learning** takes place on a **continuous** search space of (sub)differentiable functions; **reasoning** takes place in general on a **discrete** space as in the case of goal-directed theorem proving." (emphasis added). In the next chapter, we will leverage these insights and examine how System 1 and System 2 can be combined towards true AI, while grounding our arguments in computational theory.

Chapter 4

Computational Necessity

Not only is the Universe stranger than we think, it is stranger than we can think.

Werner Heisenberg
Across the Frontiers

In the preceding chapters, we built a layered framework to understand AI as a computational system based on our augmented Marr's Levels (fig. 1.7). Introducing DPT in chapter 2 connected types of processing on the Meta-Algorithmic Level to characteristic behavior on the Computational Level, while chapter 3 grounded them in implementation. With this complete theory of two enclosed computational systems, we will now tackle our final question: Why are both types of processing needed for intelligence?

We will answer this question by going back to our definition of intelligence in terms of optimality with respect to the cost-accuracy tradeoff (section 1.1.5) and finally shift our focus to include both parameters. In doing so, we will begin by describing how the cost of a computation can be measured using the resource metrics of TIME, SPACE and DATA. This will allow us to examine reasoning under resource constraints. From there, we outline our investigation methodology.

4.1 Measuring Complexity

Before we start with our explorations, we will establish the ground methodology for this chapter. First, let us briefly outline how we can combine our previous definition of intelligence, that framed it as an optimality requirement relative to the cost-accuracy tradeoff, with a measure of problem hardness. Here, the most straightforward answer is the correct one: the harder the problem, the steeper the cost-accuracy tradeoff. Formally, we can measure problem hardness with tools from *Computational Complexity* and *Computational Learning Theory (CLT)*, that allow us to put numerical bounds on how much it costs to achieve a certain average or guaranteed level of accuracy for a problem. This cost is measured in worst-case resource requirements, which can measure either SPACE, TIME or DATA. We will see that formal reasoning problems have intrinsic properties that can make them *objectively* harder to solve than other problem types. As per our definition, the intelligence of a reasoner is contingent upon its ability to behave optimally for any kind of problem - and maybe especially for these hard ones. Consequently, we will assess the intelligence of different AI architectures by exploring whether they can implement the pareto-optimal strategy that can respond to the unique hardness formal reasoning tasks present. In doing so, we will outline for each criterion, if and why a neural System 1 or symbolic System 2 alone fails to fulfill it, and how the combination of both mitigates these issues. Demonstrating the superiority of these solutions under complexity challenges provides strong evidence for the computational necessity of dual processes. Each section will thus consist of a *[R]equirement* that for an intelligent agent following our definition of intelligence (section 1.1.5), the *[C]omputational Background* providing the necessary formal concepts to grasp the complexity of formal reasoning problems, *[A]rchitectural Examples* that solve these problems and a *[D]ual Process Conclusion*, that deduces roles for System 1 and System 2, which exclusively enables a reasoner that possesses to behave intelligently.

To summarize how this chapter ties all previous results together in one sentence: With the meta-algorithmic tools of Dual Process Theory and the newly introduced Complexity Theory, we can analyze and understand what kinds of information processing are necessary in both learning and reasoning to implement pareto-optimal solution strategies for computationally hard formal reasoning problems.

4.2 Unbounded Reasoning: Working Memory SPACE and the Capacity for Deep Thought

In this section, we will investigate a reasoners ability to perform optimally with unbounded resources. We will see that, when we do not limit time and working memory access, the ability to think deeply about complex problems hinges upon the reasoner's ability to reliably execute step-by-step computations, which requires interaction with (something like) a working memory. This is why the computational resource SPACE will play a central role in this assessment.

4.2.1 [R] Perfect Reasoning Accuracy with Unbounded Costs

Establishing the positively monotone relationship between the cost and accuracy of pareto-optimal solution strategies demonstrated to us that the most accurate intelligent solution strategy for a problem is also the most computationally expensive. While we will delve into the precise cost metrics in the next section (section 4.2), we already know intuitively that solving some problems are so difficult to solve that they require an absurdly large amount of time to solve. Consider for instance this example from popular culture: The supercomputer *Deep Thought* in the book *The Hitchhiker's Guide to the Galaxy* takes 7.5 million years to compute the answer to the *Ultimate Question of Life, the Universe, and Everything* (Adams 1980). Even though finding a solution to this question took an almost unreasonable amount of time, we would at least want an AI to be *capable* of expending an arbitrary amount of resources if this is necessitated by the complexity of the problem.

Thus, the first computational requirement for an intelligent reasoner is the following: If a problem is solvable and the reasoner has access to unbounded resources, it should be able to give an answer with perfect accuracy. We can informally think of this as the ability to engage in *arbitrarily deep thought*. Formally, this requirement is captured by *Turing Completeness*, which we will define in the following. An intelligent reasoner has to be Turing Complete, thus assessing whether System 1 or System 2 alone qualify determines whether they are by themselves, valid candidates for AI.

4.2.2 [C] Turing Completeness

In defining Turin Completeness, we first have to briefly delve into the notion of Turing Machines. A **Turing Machine** (TM) is a theoretical construct introduced by Turing (1936) is a finite-state machine that operates on ordered sequences of symbols. These lists are provided in the form of tapes. The first finite length tape represents the problem input and the second unbounded tape is initially blank and acts as the internal working memory of the machine. The machine has one attention head, that can perform operations on one cell at a time. That is, it can read cells on either tape or writing on the memory tape. Lastly, a program deterministically controls the actions of the attention head. TMs are the most basic form of any computer and we know that any computable function can be computed by a TM (De Mol 2025).

Since we have been talking about computations for the whole duration of this thesis, defining it now in this level of detail might seem a bit over-the-top. However, it helps us with the formalization of different notions we previously discussed informally, or track the *cost* of a computation. Using the concept of TMs, we can now quantify precisely:

- the problem size, which is given by the length of the input tape,
- the amount of time-steps a computation, measured by the number of operations the attention head performs (TIME complexity)
- the amount of working memory used, measured by the number of cells of the memory tape that the attention head inscribes (SPACE complexity).

Following this definition, a *Universal Turing Machine (UTM)* is a TM that can simulate any other TM. That is, if there exists any other TM that can calculate the solution to a problem, the UTM can (theoretically) calculate the solution to the problem. Finally, a system is *Turing Complete* if it can simulate a Universal Turing Machine. Summing all of this up: If a system is Turing Complete, it has the ability to solve any computable problem when given access to an unbounded working memory and set no time constraints. An important caveat for us is that this is a *theoretical* ability, practically contingent upon the knowledge about a correct solution strategy for the problem. Further, is is important to note that this scenario so far is unbounded - we have not placed any constraints on time or space resource usage for the computation. What we want to test by considering Turing Completeness is something about the power of a system, measured by its ability to perform reliable step-by-step computations, which in turn require interaction with a working memory.

4.2.3 [A] Working Memory Interaction

Arbitrarily deep thus requires the reasoner to be able to execute a computation with arbitrarily many steps. This, in turn requires the reasoner to keep track of the subproblems that need to be solved, as well as their incremental solutions, which necessitates something like a working memory. The capacity for deep thought, and formally Turing Completeness, of a reasoner is contingent upon this ability - and we will see that the reliable interaction with spacial resources crucially determine a system's Turing Completeness.

Let us start off by establishing the Turing Completeness of System 2, which turns out to be a short pursuit. It is a well-known fact that most basic programming languages, like Python, are Turing Complete. Thus, if a symbolic system reasons based off python programs (assuming enough knowledge about the domain is specified), and has access to unbounded time and space, this forms a Turing Complete reasoner.

On the other hand, neural networks do not fundamentally operate with working memory - which consequently also applies to LLMs. As we outlined previously, their thinking consists of one step - feeding the input into the system and obtaining the output. Consequently, neural networks in the most basic form cannot think longer about more complex problems - entirely lacking the ability for deeper thought of any degree, let alone arbitrary thinking depth. Understanding the precise LLM architecture can give us some clues as to how embedding neural modules in a recursion unlocks additional power. LLMs are designed to iteratively predict the next token for a sequence of previous tokens[1]. Obviously, in most cases the LLM is expected to give an answer longer than one token and should instead output a long, *cohesive* text - which requires more than one step. This ability is achieved by a nuance in the LLM architecture posed by the *context window* ((Vaswani et al. 2017), Zhang et al. (2024)). Before the LLM generates any output, the only context it considers for its answers comes from the prompt itself. Then, as it is generating each new token, this already generated sequence gets added to the previous context and thus enables the LLM to generate long, cohesive sequences. In this recursive loop, the LLM considers the context like a *sliding window* that dynamically readjusts to include the output of one iterative step as the input of the next. To sum up; Neural reasoners by themselves do not use a working memory as part of their cognition, which disables them from engaging in mutli-step computations. However, architectures like LLMs embeds their neural network in a recursion, which allows them to iteratively repeat their one-step process and thereby achieve something similar.

So now the question: Can a System 1 alone be Turing complete? The straightforward answer is no, as a neural network by itself cannot perform multi-step computations. However, from the above observations about LLMs, two slight tweaks could get us there, both follow pretty directly and rely on embedding the one-step neural reasoner in a recursive structure. The first option is to give an LLM access to an external working memory, that they can utilize to record subtasks and incremental

1. Tokens are a chunks of language, which we can think of as the atomic units LLMs reason about. Instead of single letters, they generate one token at a time, which could be anything from a single letter to a whole word.

solutions, much like a symbolic system would. Schuurmans (2023) show that this architecture is Turing complete (c.f. Neural Turing Machines, Graves, Wayne, and Danihelka (2014)). The second method relies on expanding on the natural recursion that is already part of the LLM architecture due to the sliding context window. *Chain-of-Thought (CoT)* Prompting, introduced by Wei, Wang, et al. 2022, describes the strategy of appending the phrase *'Think step-by-step.'* to the problem description prompt. This technique at first seems almost too simple, yet it has shown to be incredibly effective in improving reasoning performance of LLMs and has become standard practice (Xu et al. 2025). Further, there is a theoretical explanation for this phenomenon that goes beyond the mysterious emergence of reasoning properties: This technique can be compared to a human taking out a physical scratchpad to tackle complex problems (e.g. Hagendorff, Fabi, and Kosinski (2023)). This method allows us to note down our intermediate solutions and strategies, then look at what is written to support our further reasoning. We can see how this method achieves something very similar to using internal working memory (c.f. Huang et al. (2025)). Recent work has shown that LLMs with CoT prompting, too, are Turing Complete (Qiu et al. 2025).

We have so far established the following: While LLMs, as representatives of System 1, are not Turing Complete by themselves, they can be made Turing Complete with slight modifications. These are, of course, highly theoretical results. As a last step, we will investigate whether this translates into practice. First, making LLMs interact with a traditional working memory has not established itself as a popular method to increase reasoning abilities, and there are only few examples of implementations (e.g. Schick et al. (2023)). At this current point, we thus cannot say much about this methods efficacy, though the unpopularity of this approach might indicate it not being very promising. On the other hand, CoT Prompting, as we outlined above, is a standard practice when improving the formal reasoning capabilities of LLMs, or, as has become common terminology, improving them towards *Large Reasoning Models.* However, the theoretical Turing Completeness of this technique hinges not only upon the fact *that* this technique empirically works but also that it works *how* we think it does. Recent work by Barez et al. (2025) indicates that this is not the case and demonstrate how LLMs often produce a faulty reasoning chain, while still outputting the correct solution. This more so speaks to a superficial emulation of step-by-step thinking, where the meta-reasoning is performed by the human prompter, who by CoT prompting tells the model to *'use System 2'.* However, what follows is a System 1 response that only acts like System 2 by replicating its training data. If this is the case and the LLM consequently fails to *use* the intermediate results, they do not and maybe cannot reason step-by-step, meaning their theoretical abilities do not translate into practice.

Lastly, this potential limitation of LLMs is also supported by a recent empirical result, that analyzes the deep thinking abilities of LLMs, as well as LRMs. Shojaee et al. (2025b) examine whether LLMs and LRMs are able to solve formal reasoning problems that require an increasing amount of steps to solve. They show that both *'experience complete collapse'* beyond a certain complexity, despite having the necessary resources to execute the strategy. Notably, this is even the case if the correct *solution strategy* is provided as part of the prompt, indicating that the problem lies not with the understanding of the problem but with the execution of the strategy. These results show that System-1-like architectures, even before the requirement of *arbitrarily* deep thinking, already face severe limitations in their ability to consistently reason step-by-step when a large but *limited* depth is required - indicating much more *fundamental* inability to execute long solution strategies (Shojaee et al. 2025b).

4.2.4 [D] Conclusion

In this section, we have formulated our first requirement for an intelligent reasoner as the ability to engage in arbitrarily deep thought in the unbounded case, formally fulfilling Turing Completeness. We saw how this capacity is contingent upon reliable interaction with (something like) a working memory, which many symbolic systems do by default, it is not naturally part of neural processing. This dynamic aligns congruently with the neural and symbolic DPT counterparts; Evans and Stanovich (2013) determined the (non-)reliance on working memory as defining characteristics for the Systems (fig. 2.1). Last, while theoretical results show that modified LLMs, as neural System 1 architectures, are Turing Complete, it is unclear whether these results translate to practical application. A definitive conclusion hinges upon improved theoretical understanding of these systems.

4.3 Bounded Reasoning: So Many Options, yet so Little TIME

We now move on to the bounded case and assess a reasoner's abilities to achieve the best possible solution within resource constraints. In doing so, we will focus primarily on the resource TIME, since this metric might be easier to intuitively grasp[2].

2. The time and space resources a computation requires are often dependent on each other in non-trivial ways. Sometimes, different algorithms for the same task show a time-space tradeoff, like in the case of sorting algorithms (see section 5.3). Due to this intertwinement of these metrics and our assessment of overall *cost*, the analysis of one of them is sufficient to make our point

4.3.1 [R] Maximal Reasoning Accuracy for Bounded Resources

In the above section, we argued that an intelligent reasoner should be able to produce the best possible solution to a problem given unbounded space and time resources. Now, we shift our attention to the bounded case. We will focus on time as a bounded resource to explore the case where executing strategies that lead to an exact solution require more resources than available. An intelligent reasoner should thus aim for the strategy with the highest accuracy given the resources available. Since the relation between cost and accuracy for pareto-optimal strategies is positively monotone, the agent should aim to expend the most costly strategy that is still within resource bound requirements.

For our investigation, we will first explore the complexity-theoretic underpinnings of how we can put numerical bounds on the number of time-steps solving a problem takes. This will allow us to get a grasp on what kinds of problem-solving strategies, and under which circumstances, require more resources than are plausibly available to a reasoner. Then, we will look at AI implementations that solve these kinds of problems and assess how they deal with these limitations and conclude with implications for Systems.

4.3.2 [C] Computational Complexity

As intelligent agents problem-solve all day, we know that not all problems are the same level of difficult. We have already discussed, for instance, how addition of bigger integers is harder to execute than addition of smaller integers. This is an intrinsic hardness of the problem that affects all types of reasoners the same - while a calculator is quicker at them than a human in their head, bigger calculations will also take them longer. This increase in difficulty is a direct effect of growing *problem size*, also referred to as *input length.* But we can also look at a different measure, namely the **intrinsic complexity** between problems. Consider for instance the difference between addition and multiplication. If given two numbers to add together, most people would use a pen and paper to perform a written addition, where both numbers are placed on top of each other and their digits are added up sequentially while 'carrying the one'. Long multiplication is a similar method to obtain the product of two numbers, but the amount of intermediate steps grows with the number of minimum number of digits of the factors. There are necessarily more steps involved in the multiplication than their addition - and this difference becomes more pronounced the bigger we make the integers.

Yet, neither addition nor multiplication are close to the hardest problems one can think of. The difference in amount of time steps it takes is marginal between them, and in both cases it scales linearly with the size of the input. These problem are easy to solve compared to other problems; It is unlikely that the effort a simple addition or multiplication of any plausible size would bring a reasoner to its computational limits.

To find out where these are, let us consider different variations of our candy bar example. The problem size/ input length here corresponds to the number of candy bars we are evaluating. In our original scenario, even the most costly strategy of checking all the nutritional labels and comparing them to the current calorie high-score takes linear time relative to the input. As the first variation, let's assume the task was not only to find the candy bar with the highest calorie count but to sort the whole shelf of candy bars according to their calorie count. This problem is already a lot harder - it is no longer sufficient to go over the shelf once; most strategies would require iterating over the candy bars multiple times. The amount of time-steps required to solve sorting problems grows more than linearly with the input size [3]. Both these problems belong to the complexity class P, meaning the amount of time-steps required to solve a problem instance grows at worst polynomially with the problem size. While polynomial growth does not sound great initially, problems in this complexity class are still considered to be *efficiently solvable*. Of course, this does not mean solving them never requires a large amount of resources or that this amount of resources is always available, just that the necessary resources scale often scale at a manageable rate in practice.

To identify a crucial complexity threshold with detrimental impacts on time complexity, let us consider a final variation of our candy bar scenario. You are fed up by having to choose between these calorie-maxing and search strategies all of the time and would rather settle the dispute once and for all - you want to find the candy bar that has the most calories out of all the candy bars in the whole country. So you devise a plan to collect one of every available candy bar to compare them in your room. There is only one problem: No store sells all of them. This leaves you with the additional challenge of figuring out which stores to visit. Since you do not want to waste time, your goal is to visit the least possible amount of stores while still obtaining all bars. Finally, this problem is provably more difficult than the previous two. It is formally known as SET-COVER (see section 5.2.1) and is one of Richard Karp's 21 NP-complete problems (Karp 1972). Problems that belong in the complexity class NP are significantly harder than the problems in P under the standard complexity assumption that $P \neq NP$. According to a second standard assumption called the *Exponential Time Hypothesis* (ETH), there does not exist an algorithm that provides a solution to most of these problems in sub-exponential time (Cygan et al. 2015). An even stricter version of this hypothesis, the Strong ETH, states that for some of these problems, brute-forcing through the exponential search space is essentially optimal (Cygan et al. 2015).

Based on these assumptions, any exact algorithm for NP-hard problems grows at least exponentially in the amount of worst-case time-steps in relation to the problem size. This poses a severe challenge for a reasoner wanting to find the exact solution to these problems. They are *intractable* or *not efficiently-solvable*: Responding to large problem instances can quickly exhaust more resources than are available in the universe (van Rooij et al. 2024).

3. Between $\mathcal{O}(n \cdot log(n))$ and $\mathcal{O}(n^2)$, depending on the space complexity (see section 5.3).

Heuristics. This combinatorial explosion makes it impossible to execute exact algorithms under practical resource constraints, necessitating the application of *heuristics*. In complexity theory, this term mostly refers to strategies that guarantee a lower bound on the accuracy of a solution, that can be formally proven based on fundamental properties of the problem. For our problem to accumulate all candy bars in the least time consuming way, there exists a heuristic in P, providing an efficient strategy to find an approximate solution. This strategy is the *greedy* algorithm for SET-COVER and consists of iteratively visiting the store that, at that point, has the most candy bars that have not yet been collected (see section 5.2.2). We can now explore how the costs of these strategies relate and formally define the cost-accuracy tradeoff between them: Assuming a reasonable amount of stores (see section 5.2.4), the exact algorithm takes time $\mathcal{O}(2^n)$ the greedy algorithm takes time $\mathcal{O}(n^2)$. We also know that the greedy algorithm guarantees an approximation factor of $1 + ln(n)$ (Chvatal 1979). This factor quantifies how much larger the heuristic solution is compared to the optimal - in our case we would visit $1 + ln(n)$ more stores than optimally needed. For 5, 10 and 100 candy bars, this gives us the cost and approximations depicted in table 4.1. For as little as 100 candy bars, the disparity is grave: $10,000$ and $\approx 1.310^{30}$. Assuming a reasoner performs at the rate of one computational time-step per second, this is the difference between under three hours and around $4 \cdot 10^{22}$ years.

n	Greedy (n^2)	Exact (2^n)	Approx. factor
5	25	32	≈ 2.61
10	100	1024	≈ 3.30
100	10,000	$\approx 1.3 \times 10^{30}$	≈ 5.61

Table 4.1. Time steps and approximation factors for greedy vs. exact Set Cover algorithms under our runtime assumptions

The goal of heuristics is often described as mirroring something like human intuition, that allows making quick and clever shortcuts in an otherwise insurmountably big search space. The *rule-based* heuristic above, however details another step-by-step strategy and, while being faster than the exact algorithm, still requires a considerable amount of time. More importantly, finding such heuristics is a computationally complex problem in itself (Pendurkar et al. 2022). This highlights the need for more immediate and dynamic *pattern-based* approximations, more in line with System 1 intuitions, that better align with human intuition.

This excursion can give us an idea about the true intrinsic difficulty of problems in NP - obtaining an exact solution to them is not only much harder than for lower-complexity problems but significantly so, making an exact solution infeasible in many cases. Critically, many formal reasoning problems are known to be at least NP-hard (for instance, planning problems (Bylander 1994) or finding mathematical proofs(Dean and Naibo 2024)). Consequently, formal reasoning problems often present the agent with the challenge of finding the best possible approximation instead of the exact solution, while staying within feasible time constraints.

4.3.3 [A] Neuro-Symbolic AI

For this, we will first have a look at AI architectures that combined Neural and Symbolic components. These approaches are often mentioned when System 1 and System 2 in AI are discussed. All of them adhere to our division where System 1 has a neural implementation and System 2 a symbolic one, but they combine them in different ways.

α	β
$\boxed{\gamma}$	$\boxed{\delta}$

Combining Induction and Transduction. The first architecture we will consider consisted of two modules that both tried to solve the same task. Li et al. (2024) built this model to tackle tasks in the *Abstraction and Reasoning Corpus (ARC)*. This corpus contained a large, hand-crafted set of problems that test a reasoner's abilities to perform few-shot learning and broad generalization (Chollet 2019). Every problem contains of a test set of small number of input, output tuples that depict transformations of one colored grid into another, while these transformations all follow the same underlying principle. The task is to take a new test input grid and predict its transformed version according to the same rule (see fig. 4.1). As we outlined above, the tasks were designed to require two skills. First, few-shot learning, which is notoriously hard for neural systems (Chollet 2019). As we have discussed in detail before, their type of learning requires picking up on patterns in the data. Why this is difficult with only a few examples is easiest to see when we think back to the concept of hostile environments (section 2.5): System 1 can not form good intuitions when the environment is void of valid feedback cues that can be repeatedly sampled, which is only the case with abundant amounts of training data. Second, the ARC problems require broad generalization; the grid transformations require the recognition and application of abstract concepts. This requirement arises from the intrinsic difficulty of the underlying planning, which we are often NP-hard (Bylander 1994; Dean and Naibo 2024)). Exhaustively exploring all options of grid manipulation is computationally infeasible due to the combinatorial explosion of possible options, making it necessary to rely on pattern-based intuition for some tasks.

Li et al. (2024) constructed an architecture that could perform both induction and transduction on the tasks to tackle both these challenges. As we discussed in chapter 2, induction corresponds to System 2 reasoning while transduction corresponds to System 1 reasoning. In chapter 3, we outlined how these methods are natural to symbolic and neural systems respectively. The authors set up the following processing pipeline: First, an LLM proposed the code for variations of different programs that transformed the input grid to the output grid. A python compiler would then run these on the training examples and verify whether they predicted the correct transformation. If a program that fit all the test examples could be found, it was applied to the test input grid to obtain the output. If the inductive method did not succeed within a predetermined computational limit, the LLM would instead guess an output directly and thereby utilize a transductive approach. The found that the inductive and transductive approaches had similar solution rates on the problems with little problem overlap between them. In combination, they were thus able to yield better overall performance than either approach individually. Further, there were clear problem characteristics predicting their respective success:

While Induction performed better on tasks requiring precise computations like counting (fig. 4.1a), transduction did better on tasks requiring fuzzy object recognition, like completing patterns (fig. 4.1b).

Interpreting this in the Dual Process Framework, we can see parallels to how they Systems are thought to interact in human problem solving, although the intervention happens the other way around. Default-interventionism (section 2.3.1) proposes that System 1 automatically responds unless meta-reasoning processes flag that it would not yield accurate results given the task environment. Li et al. (2024) chose the reverse route: System 2 responds automatically until a reasonable amount of computational cost is exhausted, after which System 1 takes over. Further, the abilities of the inductive System 2 and transductive System 1 approach empirically support our predictions. If a reliable pattern is present that can be easily picked up and verified based on few examples, System 1 excels. On the other hand, if careful step-by-step exploration is required, System 2 solves the task.

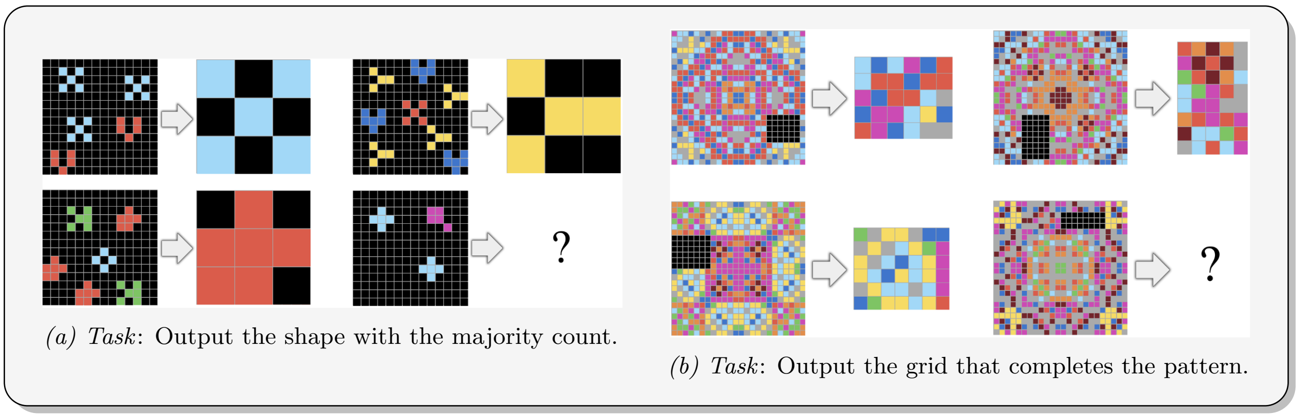
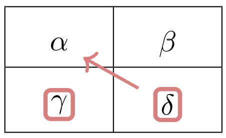

?
?
(a) *Task*: Output the shape with the majority count.
(b) *Task*: Output the grid that completes the pattern.

Figure 4.1. Two example problem instances from the *Abstraction and Reasoning Corpus (ARC)*. Graphic from Chollet (2019).

AlphaGo. Another neuro-symbolic application that is often mentioned when System 1 and System 2 come up in AI discourse is deep mind's *AlphaGo* (Silver et al. 2016). Go is a game that is very intrinsically hard[4] and has a big input size (measured in the size of the playing field). There are around 10^{170} legal game states (Walraet and Tromp 2016) for a 19x19 board - much more than atoms in the universe, of which there are only around 10^{80} (Borowiec 2016). Because of this combinatorial explosion of possible strategies, Go was always considered one of the biggest challenges for AI. Without a good intuition, not even an incredibly powerful reasoner with bounded resources would be able to come up with good strategies.



AlphaGo was a program designed to tackle this challenge by combining the underlying symbolic understanding of the game's discrete components and rules with reinforcement learning (Silver et al. 2016). Specifically, they used a method called *Monte Carlo tree search*. At every point in the game, a neural

4. Precisely, Go is PSPACE-hard (Lichtenstein and Sipser 1980).

network could guess promising directions the strategy should be taken in, after which the symbolic search space could be unrolled from this standpoint up until a certain, predetermined, depth. Prior to application, the deep neural network was trained using reinforcement learning, which allowed it to pick up an intuition about which options should be explored depending on the game state.

This approach allowed *AlphaGo* the agent to cleverly navigate the search space, while using its trained intuition to avoid having to explore a computationally infeasible amount of possible strategy options. Importantly, System 1 learned while having access to and underlying System 2 understanding of the problem and traversed an existent symbolic search space of legal moves as its domain. This is similar to Kahneman's example of learning to drive a car: One does not just get in and guess how that might work based on intuition. Rather, the System 2 understanding of road rules and car mechanics underlies the slow and iterative construction of this intuition (Kahneman 2011). The interaction of the Systems in this architecture does not stop at learning though - even during execution, System 1 does not determine the agent's actions but instead tells System 2 what options it should systematically explore. This approach represents an interaction of System 1 and System 2 in both learning and reasoning that combats the intrinsic difficulty of an exponential search space and thus provides a nice basis for understanding the Systems' interaction throughout an entire problem-solving process.

4.3.4 [D] Conclusion

This exploration of neuro-symbolic AI architecture provides a solid foundation for examining how System 1 and System 2 strategies and internal representations can interact to tackle the intrinsic complexity of solving formal reasoning problems. We outlined how their respective strengths and weaknesses in performance stem from problem complexity and cost-accuracy tradeoffs based on problem characteristics. Further, we already got a first glimpse of how the presence of System 2 understanding can aid in better System 1 learning; potentially constraining the formation of an educated intuition in crucial ways that make it more reliable than it would be otherwise. The lack of this principled symbolic understanding in ordinary training of LLMs for instance, might be why Kambhampati et al. (2024) deemed them a "giant *pseudo* System 1" (emphasis added) - the formation of a *real*, more reliable System 1 might require the symbolic scaffolding a System 2 provides.

4.4 Bounded Learning: Infinite Domains and Limited DATA

Lastly, we will address the complexity of learning and assess intelligent learning. Part of this will discuss the resource demands of of reasoning *to* an explanation, which can be analogously to reasoning *from* an explanation (section 3.1). Further, we will introduce the last resource metric, analyzing how much DATA a reasoner needs to observe to obtain a good understanding of the domain. Centering our debate around this metric will allow us to grasp the complexity that arise from the infinite domain sizes of formal reasoning problems (section 1.4.2).

4.4.1 [R] Understanding Infinite Domains with Finite Data

Our last criterion for intelligence will require the agent to obtain the most accurate understanding of the domain with the amount of data available to it. As we have argued before, formal reasoning problems often have infinite domain sizes (section 1.4.2), which makes it impossible to observe all problem instances and variations. Thus, a resource-bounded agent needs to be able to understand concepts from observing a finite subset of the domain, which will show that robust generalization (section 1.5) is not only a behavioral expectation but rather computationally necessary to reliably understand formal reasoning problems.

4.4.2 [C] Computational Learning Theory

The complexity of learning is analyzed in the field of *Computational Learning Theory (CLT)*. Aligned with our identification of learning as a type of reasoning (reasoning *to* an explanation, section 3.1), we can assess the computational hardness of obtaining an explanation analogously to reasoning *from* an explanation. That is, by using tools from Computational Complexity as above (section 4.3.2) to track time and space resource demands. Importantly, we will place our focus on the remaining metric that we have not yet addressed: The amount of DATA required to obtain an *accurate* understanding of the domain (this relation is called the *sample complexity*). To understand it, we need to define some core concepts of CLT first, which we will illustrate using the example problem of addition:

- Each data point represents a problem instance, called an **example**. In addition, this would look like $(4, 5)$.

- All the examples form a **domain**. For addition of integers up to 10, this could be visualized as a 10 by 10 coordinate system that contains all possible touples.
- The solution for a problem instance is referred to as a **label**. The intended label for $(4, 5)$ would be 9, the labeled example $((4, 5), 9)$.
- A **hypothesis** is the internal representation/explanation of the domain that the learner obtains. That means, for induction, the function that is learned to describe the examples from the domain seen during training and for transduction the function the learner itself implements. The correct hypothesis for addition would be $'s = x + y'$.
- The **hypothesis space** is the set of all possible hypotheses. When learning, the ultimate goal of the agent is obtaining identify a hypothesis from this set that fits the training data accurately by predicting the correct labels for the examples. This can look like $H = \{'s_1 = 9',' s_2 = x - y + 10',' s_3 = x + y'\}$.

An inductive learner will approach learning by establishing a hypothesis space, consisting of explicit rules that might describe the domain. Its goal is to shrink the hypothesis space by seeing more data and eliminating invalid hypotheses based on it. This captures how learning, in the case of symbolic systems, can also contain step-by-step reasoning *to* an explanation. Based on the labeled example $((4, 5), 9)$, the learner creates H. If it encounters $((6, 3), 9)$ next, s_2 can be eliminated. If it knows also knows that the labeling is not constant, s_1 is eliminated, leaving it with the correct understanding of s_1. Note that the assumption of a non-constant function did not come from a data point - it is an *inductive bias* instead, which we will explore in section 4.4.2, after briefly outlining how transductive learning differs from inductive learning.

Transductive learners like neural networks interact with training data differently than inductive learners. As we described in section 3.3, they internalize a pattern that captures the data directly, without relying on explicit rules. This method still generalizes from the data - however, it approaches an understanding statistically rather than logically. With every labeled example they observe, they adapt the weight of their neural connections to minimize the average prediction error based on the discrepancy between the predicted labeling and the actual label.

Inductive biases and infinite domains. The concept of *inductive biases* captures a central notion we discussed more informally above: Using what an agent already knows about the domain to inform its subsequent learning. Aligned with our framing above, a good inductive bias can shrink the hypothesis space by making assumptions about the constraints a valid hypothesis has to adhere to. Thus, it limits the *expressivity* of the hypothesis space. We can intuitively grasp this as enforcing the dependence of the labeling of one example and the labeling of another example. This dependence founds the spectrum between *memorization* and *generalization*: When memorizing data, we assume no structural dependence from one data point to another. All examples can have different labels - to predict all of them correctly, an agents have to observe all examples during training. Generalization occurs when we apply inductive biases and determine a relation between the data points. Finally, on the other end of the spectrum lies *robust* generalization, where an inductive bias entirely captures the structural dependence between data points.

To see why robust generalization is crucial for formal reasoning problems, we revisit our three types of generalization (section 1.5), view it through the lens of CLT and illustrate how the infinite domain size of formal reasoning problems (section 1.4.2) necessitate robust generalization. We assume that the learner observes, again, labeled examples of pairwise the addition of integers smaller than 10, for instance $((4, 5)9)$. The three types of generalization are visualized via graphical depiction of their domain shift in fig. 4.2 and outlined in the following:

- **(1) Generalization within the same problem and the restricted domain.** This domain includes around 100 labeled examples - memorization is costly but computationally feasible.
- **(2) Generalization within the same problem and an extended domain.** We distinguish two cases: In (2a), the addition can include integers up to 100. Analogously to the above, this domain includes around $10,000$ examples. (2b) depicts shifting from binary to ternary addition, giving us $1,000$ examples. For these domains, memorization significantly more expensive.

- **(3) Generalization to structurally related tasks.** Generalizing to tasks like subtraction requires understanding the related structure to the original task domain. If the learner relies on memorization, this problem presents it with 100 entirely new data points it has to learn.

While memorization becomes increasingly expensive for these variations, making good assumptions in the form of inductive bias can significantly decrease the sample complexity. For instance, a prior understanding of the structural relation of addition and subtraction ($x + y = s \Leftrightarrow s - x = y$) would allow a learner to completely determine (3) from (2a). This demonstrates how good inductive biases can make learning easier in the finite case and feasible in the infinite - even from finite data. Further, without them, an agent has to observe all data points in order to achieve perfect prediction accuracy; For infinite domains, this would require infinite data.

Last, we briefly outline how limiting the hypothesis space by applying inductive biases not only effects the amount of *data* required to form a correct understanding of the domain but also effects the complexity of reasoning *to* an explanation. This is easiest to see by viewing the process of reasoning *to* an explanation as a formal reasoning problem in itself. In line with our previous account of bounded reasoning (section 4.3), the hypothesis space forms a kind of search space a reasoner has to traverse. Importantly, the search space of all possible programs grows exponentially in both program length and the number of primitive operators (Rule et al. 2024), leading to problems of combinatorial explosion of the search space. This makes exploring the entire search space intractable, analogously to arguments from section 4.3.2.

This exploration of inductive biases demonstrates how limiting the hypothesis space is crucial to limit the time and data demands of an agent when learning to solve formal reasoning problems. We will not move on to discussing how this insight can be leveraged in AI architectures.

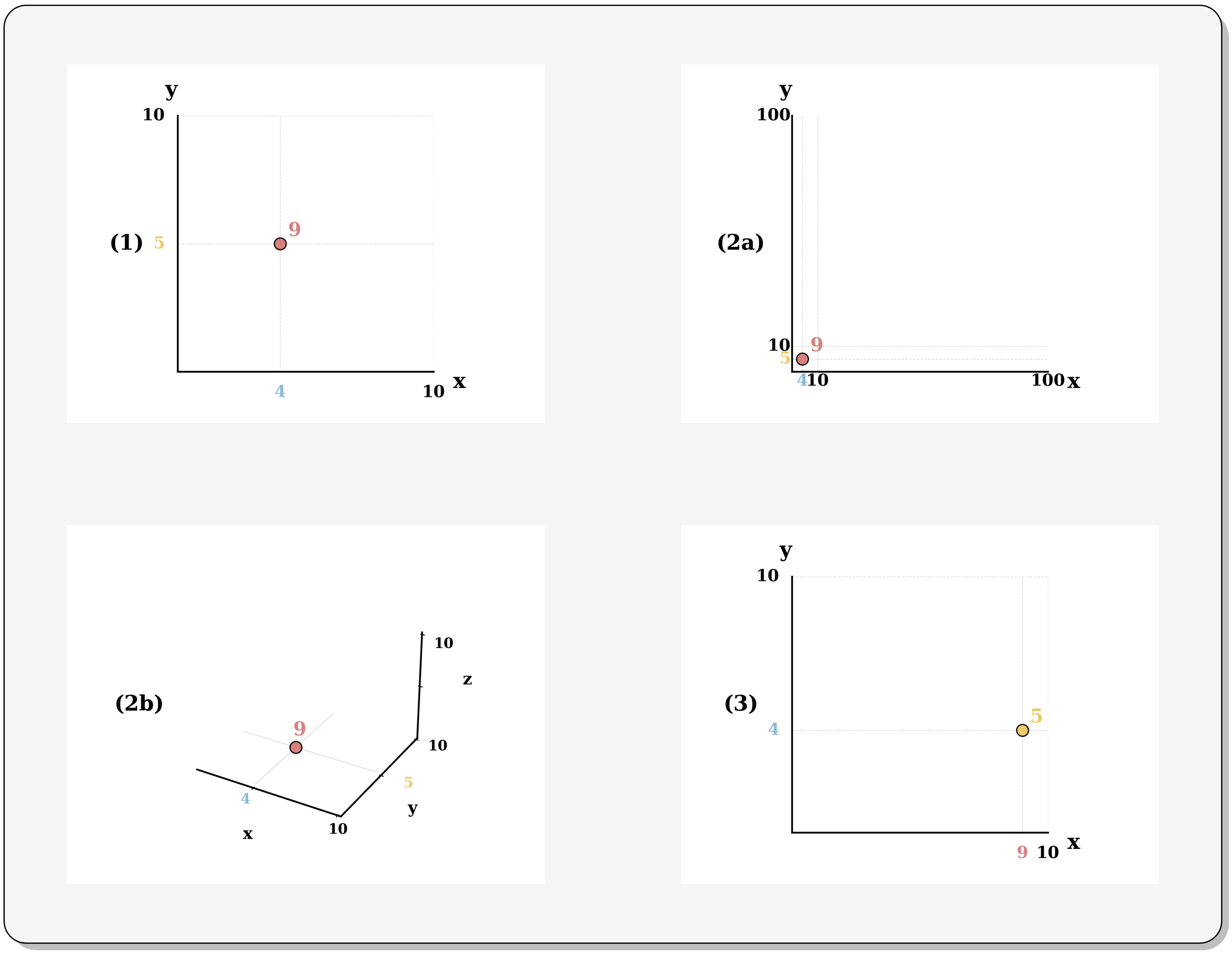
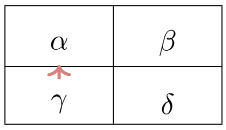


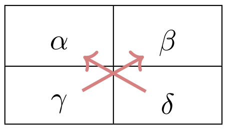
Figure 4.2. Visualizations of different problem domains, expansions and modifications. Indicated in the domain is the labeled example of $x = 4$ and $y = 5$. (1) is the original task of pairwise addition of integers in the range $[0, 10]$. (2a) shows the expansion to integers in $[0, 100]$. (2b) visualizes the introduction of a third summand $z = 0$, (3) the related task of subtraction, where we have the same domain but different labels.

4.4.3 [A] Robust Generalization as Learning Based on Understanding



Curriculum Learning. A promising method to incorporate inductive biases in neural systems is posed by *Curriculum Learning* (Bengio et al. 2009). Inspired by human cognitive development, this method determines the order in which data is represented to a learner. Instead of seeing all of it at once, the problems are ordered incremental increasing stages of difficulty - thereby allowing the learner to use its understanding of easier problem instances to generalize to more difficult ones. Training models with this method can improve generalization abilities, final task performance and learning speed (Graves et al. (2017), Bengio et al. (2009)). Subsequent work has applied this method to improve reasoning on logic problems (Abbe et al. 2024), compositional generalization (H. Zhou et al. 2023) and generalization using abstract relations (Boix-Adsera et al. 2024).

Importantly, this method only empirically improves the performance of neural networks on formal reasoning problems and still does not ensure robust generalization. In the case of neural networks, internal representations of prior understanding are implicit patterns (section 2.4.4, section 3.3). As we argued above however, inductive biases for robust generalization require placing *structural* constraints on the hypothesis space. Thus, it would be natural to employ symbolic modules that can more reliably capture this structure to obtain good inductive biases. We will see an approach to this in the following example.



Dream Coder. *Dream Coder* is a neuro-symbolic architecture developed by Ellis et al. (2020) that combines a neural module, that extract patterns from the data with a symbolic one that synthesizes consolidates this fuzzy knowledge into an explicit understanding. The neural network learns during a *wake phase*, where it guesses rules that could describe the data. We can think of this as creating the hypothesis space. A *sleep phase* follows, in which the symbolic system evaluates these hypotheses, consolidates plausible hypotheses and rules out implausible ones. When the system *'wakes up'* again, this verified and refined understanding forms a new basis, from which it can make predictions again. These phases are repeated iteratively to build a logically consistent explicit, rule-based understanding of the data. Bober-Irizar and Banerjee (2024) adapted the *Dream Coder* model to the grid transformation problems in the ARC challenge (section 4.3.3) with great success, tackling the challenges of few-shot learning and broad generalization with structured visual reasoning.

4.4.4 [D] Conclusion

In this section, we explored how the challenges of learning formal reasoning tasks are determined by their infinite domain sizes - rendering robust generalization a necessity. *Dream Coder* architecture demonstrated how System 1 and System 2 can be intertwined in acquisition a correct understanding of the domain. They complement each other in learning, where inductive biases from System 1 can constrain the hypothesis space of System 2 and thereby make the computationally costly reasoning *to* an explanation feasible under practical time constraints. Further, System 2 can help System 1 by supplying structural inductive biases about the domain that eliminates logically implausible pattern-based hypotheses. This interaction can enable a dual process agent's understanding of the domain to more efficiently converge to a correct hypothesis. While we have not discussed System 2, finding good inductive biases faces with rule-based methodology faces an analogous complexity issues as constructing good heuristics. Thus, there are no straight-forward methods for System 2 to self-learn these biases.

We outlined how the combination of System 1 and System 2 in learning improves upon their individual challenges in reasoning *to* an explanation, where System 1 biases utilize System 2 understanding to predict cheap *generalizations*, which shrink the hypothesis space of System 2 and can aid in making inductive reasoning *to* an explanation tractable. System 2 aids System 1 learning by ensuring the *robustness* of its implicit hypotheses, helping the pattern converge towards a structurally sound understanding. As a result, the interaction of the Systems in learning can achieve a more accurate understanding of the data with lower cost, implements a learning strategy that is better than each System could achieve by itself. This qualifies the dual process approach to learning as intelligent.

Chapter Conclusion

This chapter proposed three clear criteria for intelligent behavior based on the definition of intelligence as the capacity to execute a pareto-optimal problem-solving strategy with respect to environmental factors section 1.1.5. These are **(1)** In the case of unbounded resources, the agent must able to implement a perfectly accurate solution **(2)** The agent must be able to execute the most accurate strategy given bounded resources and **(3)** The agent must be able to acquire the most accurate understanding of the domain from limited data and computational resources. Using concepts from theoretical Computer Science, Computational Complexity and Computational Learning Theory allowed us to formally assess the cost and accuracy of a strategy cost in terms of SPACE, TIME, and DATA. Equipped with these resource metrics, we demonstrated the intrinsic complexity of formal reasoning problems based on their characterization in section 1.4.2: requiring exact solutions and logically consistent step-by-step reasoning, as well as having infinite domain sizes.

In section 4.2, we examined criteria **(1)** and argued that solving arbitrarily large problem instances of formal reasoning problems relies on the capacity for arbitrarily deep thought, which we outlined to correspond to the formal definition of Turing Completeness. While System 2 has no problem demonstrating this ability, System 1 shows practical limitations, potentially stemming from an inability to reliably interact with working memory SPACE. Next, we investigated requirement **(2)** in section 4.3, while focusing on the TIME complexity of formal reasoning problems. The NP-hardness of many reasoning problems makes solving them with exact System 2 solutions infeasible. This creates the need for good heuristics, that mimic the quick and dynamic nature of human-like intuition - which System 2 cannot implement or easily learn. System 1 can implement these types of heuristics. By itself, however, System 1 fails to achieve the optimal accuracy given the amount of available resources. Both Systems by themselves fail to display intelligent behavior: System 2 struggles with adapting to cost- and System 1 to accuracy-requirements. However, in combination they complement the respective limitations of the respective other, thus enabling intelligent behavior. Last, we investigated criterion **(3)** in section 4.4, where we identified the limitations of System 2 learning given by its high cost, whereas System 1 failed to capture underlying structures. We argued that inductive biases provide the solution to both issues. While System 1 can provide good inductive biases to itself when presented with ordered data, the combination with System 2 allows for robust generalization: System 1 *generalizes* on the basis of new data and prior System 2 scaffolding, while System 2 validates and consolidates these intuitions and thus enables *robust* understanding of the domain.

In conclusion, this chapter provided strong empirical and theoretical indications that, based on our characterization of DPT (chapter 2) and implementational mappings to AI architectures (chapter 3), neither an agent based exclusively on System 1, nor an agent exclusively based on System 2 can fulfill all three requirements for intelligence.

Conclusion

This work set out to explore the nature and limitations of Artificial Intelligence (AI), while using Dual Process Theory to argue that both human and artificial intelligence emerge from the interaction of two distinct types of information processing. We propose that this not only provides a descriptive account of cognition but also that the possession of dual processes is computationally necessary for intelligent agents operating under real-world resource constraints.

In chapter 1, we motivated the theoretical investigation of AI from the brittle behavior of Large Language Models (LLMs) on formal reasoning tasks (Xu et al. 2025). Their fundamental opaqueness restricts researchers to performance-based behaviorist investigations of these systems, resulting in a lack of theory (van Rooij et al. 2024). We identified central challenges that theoretical investigations of AI face using Marr's Levels (Marr 2010). This highlighted how explaining complex computations on the Algorithmic Level poses an insurmountable obstacle. To overcome this issue, we proposed a modification of Marr's Levels that replaces the Algorithmic Level with the Meta-Algorithmic Level, allowing us to characterize different types of problem solving strategies instead of explicit algorithms.

The introduction of Dual Process Theory (DPT) in chapter 2 equipped us with the necessary tools to understand the relation between the Computational and the Meta-Algorithmic Level of explanation. Drawing on multiple characterizations of two distinct types of thinking, we explored System 1, which relies on fast, approximate and implicit inferences and System 2, which reasons slowly, accurately and explicitly (Kahneman 2011; Evans and Stanovich 2013). Further, we determined the distinction between implicit and explicit processing as defining the Systems. Employing terminology from *Computational Learning Theory (CLT)*, we formalized implicit processing as transduction and explicit processing as induction and outlined how they rely on patterns and structures respectively.

This characterization allowed us to map System 1 to neural networks, which reason based on transduction and System 2 to symbolic systems, which reason based on induction in chapter 3. This showed that the typical attributes of System 1 and System 2 according to DPT are not accidental, but rather emerge from these inference mechanisms. Further, this alignment uncovered an additional dimension of learning and reasoning. This provided us with the nuanced distinction between four types of reasoning: α (acquiring understanding by internalizing patterns), γ (applying patterns to make heuristic predictions), β (learning a structural rule), and δ (applying the rule during to new data).

Finally, chapter 4 presented the core argument of this work: A complexity-theoretic justification for the necessity of combining System 1 and System 2 to implement intelligent behavior. Based on the definition of intelligence as executing pareto-optimal strategies with respect to the cost-accuracy tradeoff (Johnson and Payne 1985), we posed three clear criteria for intelligent behavior: **(1)** In the case of unbounded resources, the agent must able to implement a perfectly accurate solution **(2)** The agent must be able to execute the most accurate strategy given bounded resources and **(3)** The agent must be able to acquire the most accurate understanding of the domain from limited data and computational resources. Using tools from Computational Complexity and Computational Learning Theory, we demonstrated how properties of formal reasoning tasks result in a high intrinsic complexity of these problems. We showed how an agent operating under real-world resource constraints relies on the principled combination of System 1 and System 2 processing to implement intelligent problem-solving strategies. This analysis provides strong indications of the computational necessity of dual processes for intelligence.

Bibliography

AAAI-20 Fireside Chat with Daniel Kahneman. 2020. https://vimeo.com/390814190?ref=tw-share. Accessed June 23, 2025.

Abbe, Emmanuel, Samy Bengio, Aryo Lotfi, and Kevin Rizk. 2024. “Generalization on the Unseen, Logic Reasoning and Degree Curriculum.” *Journal of Machine Learning Research* 25 (331): 1–58. ISSN: 1533-7928, accessed August 5, 2025.

Adams, Douglas. 1980. *The Hitchhiker’s Guide to the Galaxy.* New York: Harmony Books. Accessed July 24, 2025.

Allen Newell and Herbert A. Simon. 1956. *The Logic Theory Machine.* P-868. July. Accessed July 8, 2025.

Barez, Fazl, Tung-Yu Wu, Iván Arcuschin, Michael Lan, Vincent Wang, Noah Siegel, and Nicolas Collignon. 2025. “Chain-of-Thought Is Not Explainability.”

Bender, Emily M., 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. FAccT ’21. New York, NY, USA: Association for Computing Machinery, March. ISBN: 978-1-4503-8309-7, accessed February 17, 2025. https://doi.org/10.1145/3442188.3445922.

Bengio, Yoshua, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. “Curriculum Learning.” In *Proceedings of the 26th Annual International Conference on Machine Learning,* 41–48. ICML ’09. New York, NY, USA: Association for Computing Machinery, June. ISBN: 978-1-60558-516-1, accessed July 1, 2025. https://doi.org/10.1145/1553374.1553380.

Bober-Irizar, Mikel, and Soumya Banerjee. 2024. “Neural Networks for Abstraction and Reasoning: Towards Broad Generalization in Machines.”

Boix-Adsera, Enric, Omid Saremi, Emmanuel Abbe, Samy Bengio, Etai Littwin, and Joshua Susskind. 2024. *When Can Transformers Reason with Abstract Symbols?,* arXiv:2310.09753, April. Accessed August 5, 2025. https://doi.org/10.48550/arXiv.2310.09753. arXiv: 2310.09753 [cs].

Borowiec, Steven. 2016. “Google’s AlphaGo AI Defeats Human in First Game of Go Contest.” *The Guardian* (March). ISSN: 0261-3077, accessed July 30, 2025.

Brown, Tom, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, et al. 2020. “Language Models Are Few-Shot Learners.” In *Advances in Neural Information Processing Systems,* 33:1877–1901. Curran Associates, Inc. Accessed March 24, 2025.

Brown, Tom B., Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, et al. 2020. *Language Models Are Few-Shot Learners,* arXiv:2005.14165, July. Accessed July 8, 2025. https://doi.org/10.48550/arXiv.2005.14165. arXiv: 2005.14165 [cs].

Browning, Jacob, and LeCun, Yann. 2022. "AI And The Limits Of Language" (August). Accessed July 16, 2025.

Buchanan, Bruce G., and Edward Hance Shortliffe. 1984. *Rule-Based Expert Systems : The MYCIN Experiments of the Stanford Heuristic Programming Project.* Reading, Mass. : Addison-Wesley. ISBN: 978-0-201-10172-0, accessed July 8, 2025.

Buckner, Cameron J. 2024. *From Deep Learning to Rational Machines: What the History of Philosophy Can Teach Us about the Future of Artificial Intelligence.* Oxford University Press. ISBN: 978-0-19-765330-2.

Burgoyne, Alexander P., Cody A. Mashburn, Jason S. Tsukahara, David Z. Hambrick, and Randall W. Engle. 2023. "Understanding the Relationship between Rationality and Intelligence: A Latent-Variable Approach." *Thinking & Reasoning* 29, no. 1 (January): 1–42. ISSN: 1354-6783, 1464-0708, accessed May 29, 2025. https://doi.org/10.1080/13546783.2021.2008003.

Burnell, Ryan, Han Hao, Andrew R. A. Conway, and Jose Hernandez Orallo. 2023. *Revealing the Structure of Language Model Capabilities,* arXiv:2306.10062, June. Accessed August 4, 2025. https://doi.org/10.48550/arXiv.2306.10062. arXiv: 2306.10062 [cs].

Bylander, Tom. 1994. "The Computational Complexity of Propositional STRIPS Planning." *Artificial Intelligence* 69, no. 1 (September): 165–204. ISSN: 0004-3702, accessed July 4, 2025. https://doi.org/10.1016/0004-3702(94)90081-7.

Campbell, Murray, A. Joseph Hoane, and Feng-hsiung Hsu. 2002. "Deep Blue." *Artificial Intelligence* 134, no. 1 (January): 57–83. ISSN: 0004-3702, accessed June 30, 2025. https://doi.org/10.1016/S0004-3702(01)00129-1.

Cheng, Yuxing, Yi Chang, and Yuan Wu. 2025. *A Survey on Data Contamination for Large Language Models,* arXiv:2502.14425, June. Accessed August 6, 2025. https://doi.org/10.48550/arXiv.2502.14425. arXiv: 2502.14425 [cs].

Chollet, François. 2019. *On the Measure of Intelligence,* arXiv:1911.01547, November. Accessed February 17, 2025. https://doi.org/10.48550/arXiv.1911.01547. arXiv: 1911.01547 [cs].

Chvatal, V. 1979. "A Greedy Heuristic for the Set-Covering Problem." *Mathematics of Operations Research* 4 (3): 233–235. ISSN: 0364-765X, accessed July 4, 2025. JSTOR: 3689577.

Crevier, Daniel. 1995. *AI: The Tumultuous History of the Search for Artificial Intelligence.* 2. [print.] New York, NY: Basic Books. ISBN: 978-0-465-02997-6 978-0-465-00104-0.

Cygan, Marek, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. 2015. *Parameterized Algorithms.* Cham: Springer International Publishing. ISBN: 978-3-319-21274-6 978-3-319-21275-3, accessed July 30, 2025. https://doi.org/10.1007/978-3-319-21275-3.

De Mol, Liesbeth. 2025. "Turing Machines." In *The Stanford Encyclopedia of Philosophy,* Summer 2025, edited by Edward N. Zalta and Uri Nodelman. Metaphysics Research Lab, Stanford University. Accessed July 29, 2025.

De Neys, Wim, and Tamara Glumicic. 2008. "Conflict Monitoring in Dual Process Theories of Thinking." *Cognition* 106, no. 3 (March): 1248–1299. ISSN: 0010-0277, accessed June 27, 2025. https://doi.org/10.1016/j.cognition.2007.06.002.

Dean, Walter, and Alberto Naibo. 2024. *Artifical Intelligence and Inherent Mathematical Difficulty,* arXiv:2408.03345, August. Accessed July 30, 2025. https://doi.org/10.48550/arXiv.2408.03345. arXiv: 2408.03345 `[math]`.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,* arXiv:1810.04805, May. Accessed July 8, 2025. https://doi.org/10.48550/arXiv.1810.04805. arXiv: 1810.04805 `[cs]`.

Ellis, Kevin, Catherine Wong, Maxwell Nye, Mathias Sable-Meyer, Luc Cary, Lucas Morales, Luke Hewitt, Armando Solar-Lezama, and Joshua B. Tenenbaum. 2020. *DreamCoder: Growing Generalizable, Interpretable Knowledge with Wake-Sleep Bayesian Program Learning,* arXiv:2006.08381, June. Accessed July 1, 2025. https://doi.org/10.48550/arXiv.2006.08381. arXiv: 2006.08381 `[cs]`.

Evans, Jonathan St B. T. 2006. "Dual System Theories of Cognition: Some Issues." *Proceedings of the Annual Meeting of the Cognitive Science Society* 28 (28). Accessed July 9, 2025.

Evans, Jonathan St B. T., and Keith E. Stanovich. 2013. "Dual-Process Theories of Higher Cognition: Advancing the Debate." *Perspectives on Psychological Science: A Journal of the Association for Psychological Science* 8, no. 3 (May): 223–241. ISSN: 1745-6916. https://doi.org/10.1177/1745691612460685.

Evans, Jonathan St B T and Stanovich, Keith E. 2013. *Dual-Process Theories of Higher Cognition.* https://www.researchgate.net/publication/258179748_Dual-Process_Theories_of_Higher_Cognition. Accessed February 19, 2025.

Feige, Uriel. 1998. "A Threshold of Ln n for Approximating Set Cover." *J. ACM* 45, no. 4 (July): 634–652. ISSN: 0004-5411, accessed July 20, 2025. https://doi.org/10.1145/285055.285059.

Fodor, Jerry A., and Zenon W. Pylyshyn. 1988. "Connectionism and Cognitive Architecture: A Critical Analysis." *Cognition* 28, no. 1 (March): 3–71. ISSN: 0010-0277, accessed July 16, 2025. https://doi.org/10.1016/0010-0277(88)90031-5.

Frederick, Shane. 2005. "Cognitive Reflection and Decision Making." *Journal of Economic Perspectives* 19, no. 4 (December): 25–42. ISSN: 0895-3309, accessed July 21, 2025. https://doi.org/10.1257/089533005775196732.

Gallistel, Charles R., and John Gibbon. 2002. *The Symbolic Foundations of Conditioned Behavior.* 0th ed. Psychology Press, March. ISBN: 978-1-4106-0221-3, accessed August 5, 2025. https://doi.org/10.4324/9781410602213.

Garcez, Artur d'Avila, and Luis C. Lamb. 2020. *Neurosymbolic AI: The 3rd Wave,* arXiv:2012.05876, December. Accessed February 17, 2025. https://doi.org/10.48550/arXiv.2012.05876. arXiv: 2012.05876 [cs].

Geirhos, Robert, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A. Wichmann. 2020. "Shortcut Learning in Deep Neural Networks." *Nature Machine Intelligence* 2, no. 11 (November): 665–673. ISSN: 2522-5839, accessed July 16, 2025. https://doi.org/10.1038/s42256-020-00257-z.

Gigerenzer, Gerd. 2020. "What Is Bounded Rationality?" In *Routledge Handbook of Bounded Rationality.* Routledge. ISBN: 978-1-315-65835-3.

Grattafiori, Aaron, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, et al. 2024. *The Llama 3 Herd of Models,* arXiv:2407.21783, November. Accessed August 4, 2025. https://doi.org/10.48550/arXiv.2407.21783. arXiv: 2407.21783 [cs].

Graves, Alex, Marc G. Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. 2017. *Automated Curriculum Learning for Neural Networks,* arXiv:1704.03003, April. Accessed August 1, 2025. https://doi.org/10.48550/arXiv.1704.03003. arXiv: 1704.03003 [cs].

Graves, Alex, Greg Wayne, and Ivo Danihelka. 2014. *Neural Turing Machines,* arXiv:1410.5401, December. Accessed August 1, 2025. https://doi.org/10.48550/arXiv.1410.5401. arXiv: 1410.5401 [cs].

Grice, Herbert Paul. 1975. "Logic and Conversation." In *Speech Acts,* 41–58. Brill.

Hagendorff, Thilo, Sarah Fabi, and Michal Kosinski. 2023. "Thinking Fast and Slow in Large Language Models." *Nature Computational Science* 3, no. 10 (October): 833–838. ISSN: 2662-8457, accessed February 20, 2025. https://doi.org/10.1038/s43588-023-00527-x. arXiv: 2212.05206 [cs].

Haimes, Jacob, Cenny Wenner, Kunvar Thaman, Vassil Tashev, Clement Neo, Esben Kran, and Jason Schreiber. 2024. *Benchmark Inflation: Revealing LLM Performance Gaps Using Retro-Holdouts,* arXiv:2410.09247, October. Accessed August 6, 2025. https://doi.org/10.48550/arXiv.2410.09247. arXiv: 2410.09247 [cs].

Harnad, Stevan. 1990. "The Symbol Grounding Problem." *Physica D: Nonlinear Phenomena* 42, nos. 1-3 (June): 335–346. ISSN: 01672789, accessed July 8, 2025. https://doi.org/10.1016/0167-2789(90)90087-6. arXiv: cs/9906002.

Hebb, D. O. 1949. *The Organization of Behavior; a Neuropsychological Theory.* xix, 335. The Organization of Behavior; a Neuropsychological Theory. Oxford, England: Wiley.

Hogarth, Robin M. 2001. *Educating Intuition.* University of Chicago Press, June. ISBN: 978-0-226-34860-5.

Huang, Jen-tse, Kaiser Sun, Wenxuan Wang, and Mark Dredze. 2025. *LLMs Do Not Have Human-Like Working Memory,* arXiv:2505.10571, April. Accessed July 18, 2025. https://doi.org/10.48550/arXiv.2505.10571. arXiv: 2505.10571 [q-bio].

Johnson, Eric J., and John W. Payne. 1985. "Effort and Accuracy in Choice." *Management Science* 31 (4): 395–414. JSTOR: 2631455.

Jones, Cameron R., and Benjamin K. Bergen. 2025. *Large Language Models Pass the Turing Test,* arXiv:2503.23674, March. Accessed July 25, 2025. https://doi.org/10.48550/arXiv.2503.23674. arXiv: 2503.23674 [cs].

Kahneman, Daniel. 1973. *Attention and Effort.* Prentice-Hall Series in Experimental Psychology. Englewood Cliffs, New Jersey: Prentice-Hall, Inc. ISBN: 978-0-13-050518-7.

———. 2011. *Thinking, Fast and Slow.* MacMillan.

Kahneman, Daniel, and Shane Frederick. 2002. "Representativeness Revisited: Attribute Substitution in Intuitive Judgment." In *Heuristics and Biases: The Psychology of Intuitive Judgment,* edited by Dale Griffin, Daniel Kahneman, and Thomas Gilovich, 49–81. Cambridge: Cambridge University Press. ISBN: 978-0-521-79260-8, accessed July 21, 2025. https://doi.org/10.1017/CBO9780511808098.004.

Kahneman, Daniel, and Gary Klein. 2009. "Conditions for Intuitive Expertise: A Failure to Disagree." *The American Psychologist* 64, no. 6 (September): 515–526. ISSN: 1935-990X. https://doi.org/10.1037/a0016755.

Kambhampati, Subbarao, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Saldyt, and Anil Murthy. 2024. *LLMs Can't Plan, But Can Help Planning in LLM-Modulo Frameworks,* arXiv:2402.01817, June. Accessed February 17, 2025. https://doi.org/10.48550/arXiv.2402.01817. arXiv: 2402.01817 [cs].

Karp, Richard M. 1972. "Reducibility among Combinatorial Problems." In *Complexity of Computer Computations: Proceedings of a Symposium on the Complexity of Computer Computations, Held March 20–22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, and Sponsored by the Office of Naval Research, Mathematics Program, IBM World Trade Corporation, and the IBM Research Mathematical Sciences Department,* edited by Raymond E. Miller, James W. Thatcher, and Jean D. Bohlinger, 85–103. Boston, MA: Springer US. ISBN: 978-1-4684-2001-2, accessed July 30, 2025. https://doi.org/10.1007/978-1-4684-2001-2_9.

Katz, Yarden. 2012. *Noam Chomsky on Where Artificial Intelligence Went Wrong,* November. Accessed April 3, 2025.

Keren, Gideon, and Yaacov Schul. 2009. "Two Is Not Always Better Than One: A Critical Evaluation of Two-System Theories." *Perspectives on Psychological Science* 4, no. 6 (November): 533–550. ISSN: 1745-6916, 1745-6924, accessed May 1, 2025. https://doi.org/10.1111/j.1745-6924.2009.01164.x.

Keysers, Daniel, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, et al. 2020. *Measuring Compositional Generalization: A Comprehensive Method on Realistic Data,* arXiv:1912.09713, June. Accessed July 28, 2025. https://doi.org/10.48550/arXiv.1912.09713. arXiv: 1912.09713 [cs].

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton. 2012. "ImageNet Classification with Deep Convolutional Neural Networks." In *Advances in Neural Information Processing Systems,* vol. 25. Curran Associates, Inc. Accessed July 8, 2025.

Kruglanski, Arie W., and Gerd Gigerenzer. 2011. "Intuitive and Deliberate Judgments Are Based on Common Principles." *Psychological Review* 118, no. 1 (January): 97–109. ISSN: 1939-1471, 0033-295X, accessed May 1, 2025. https://doi.org/10.1037/a0020762.

Lake, Brenden M., and Marco Baroni. 2018. *Generalization without Systematicity: On the Compositional Skills of Sequence-to-Sequence Recurrent Networks,* arXiv:1711.00350, June. Accessed July 28, 2025. https://doi.org/10.48550/arXiv.1711.00350. arXiv: 1711.00350 [cs].

Lapuschkin, Sebastian, Stephan Wäldchen, Alexander Binder, Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. 2019. "Unmasking Clever Hans Predictors and Assessing What Machines Really Learn." *Nature Communications* 10, no. 1 (March): 1096. ISSN: 2041-1723, accessed July 16, 2025. https://doi.org/10.1038/s41467-019-08987-4.

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. 2015. "Deep Learning." *Nature* 521, no. 7553 (May): 436–444. ISSN: 1476-4687, accessed July 8, 2025. https://doi.org/10.1038/nature14539.

LeGris, Solim, Wai Keen Vong, Brenden M. Lake, and Todd M. Gureckis. 2024. *H-ARC: A Robust Estimate of Human Performance on the Abstraction and Reasoning Corpus Benchmark,* arXiv:2409.01374, September. Accessed August 5, 2025. https://doi.org/10.48550/arXiv.2409.01374. arXiv: 2409.01374 [cs].

Levin, Janet. 2023. "Functionalism." In *The Stanford Encyclopedia of Philosophy,* Summer 2023, edited by Edward N. Zalta and Uri Nodelman. Metaphysics Research Lab, Stanford University. Accessed June 23, 2025.

Lewis, Martha, and Melanie Mitchell. 2024. *Evaluating the Robustness of Analogical Reasoning in Large Language Models,* arXiv:2411.14215, November. Accessed February 17, 2025. https://doi.org/10.48550/arXiv.2411.14215. arXiv: 2411.14215 [cs].

Li, Wen-Ding, Keya Hu, Carter Larsen, Yuqing Wu, Simon Alford, Caleb Woo, Spencer M. Dunn, et al. 2024. *Combining Induction and Transduction for Abstract Reasoning,* arXiv:2411.02272, December. Accessed March 18, 2025. https://doi.org/10.48550/arXiv.2411.02272. arXiv: 2411.02272 [cs].

Lichtenstein, David, and Michael Sipser. 1980. "GO Is Polynominal-Space Hard." *Journal of the ACM* 27 (2): 393–401. https://doi.org/doi:10.1145/322186.322201.

Marcus, Gary. 2020. *The Next Decade in AI: Four Steps Towards Robust Artificial Intelligence,* arXiv:2002.06177, February. Accessed April 29, 2025. https://doi.org/10.48550/arXiv.2002.06177. arXiv: 2002.06177 [cs].

———. 2022. "Deep Learning Alone Isn't Getting Us To Human-Like AI" (August). Accessed July 16, 2025.

Marr, David. 2010. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information.* MIT Press, July. ISBN: 978-0-262-28898-9.

McCarthy, J., Minsky, M. L., Rochester, N., and Shannon, C. E. 1955. *A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence.*

McDermott, Drew. 1976. "Artificial Intelligence Meets Natural Stupidity." *SIGART Bull.,* no. 57 (April): 4–9. ISSN: 0163-5719, accessed June 23, 2025. https://doi.org/10.1145/1045339.1045340.

Merullo, Jack, Carsten Eickhoff, and Ellie Pavlick. 2024. *Language Models Implement Simple Word2Vec-style Vector Arithmetic,* arXiv:2305.16130, April. Accessed April 24, 2025. https://doi.org/10.48550/arXiv.2305.16130. arXiv: 2305.16130 [cs].

Minsky, Marvin, and Seymour A. Papert. 1969. *Perceptrons: An Introduction to Computational Geometry.* The MIT Press. ISBN: 978-0-262-34393-0, accessed July 8, 2025. https://doi.org/10.7551/mitpress/11301.001.0001.

Mirzadeh, Iman, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. 2024. *GSM-Symbolic: Understanding the Limitations of Mathematical Reasoning in Large Language Models,* arXiv:2410.05229, October. Accessed August 4, 2025. https://doi.org/10.48550/arXiv.2410.05229. arXiv: 2410.05229 [cs].

Mitchell, Melanie. 2021. *Why AI Is Harder Than We Think,* arXiv:2104.12871, April. Accessed May 15, 2025. https://doi.org/10.48550/arXiv.2104.12871. arXiv: 2104.12871 [cs].

———. 2023. "AI's Challenge of Understanding the World." *Science* 382, no. 6671 (November): eadm8175. Accessed February 17, 2025. https://doi.org/10.1126/science.adm8175.

———. 2024. "The Metaphors of Artificial Intelligence." *Science* 386, no. 6723 (November): eadt6140. Accessed February 17, 2025. https://doi.org/10.1126/science.adt6140.

Moosavi-Dezfooli, Seyed-Mohsen, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. 2017. "Universal Adversarial Perturbations." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 1765–1773. Accessed July 16, 2025.

Moravec, Hans. 1988. *Mind Children: The Future of Robot and Human Intelligence.* Harvard University Press. ISBN: 978-0-674-57618-6.

Nadin, Mihai. 2023. "Intelligence at Any Price? A Criterion for Defining AI." *AI & SOCIETY* 38, no. 5 (October): 1813–1817. ISSN: 1435-5655, accessed August 4, 2025. https://doi.org/10.1007/s00146-023-01695-0.

Newell, Allen, and Herbert A. Simon. 1976. "Computer Science as Empirical Inquiry: Symbols and Search." *Commun. ACM* 19, no. 3 (March): 113–126. ISSN: 0001-0782, accessed July 8, 2025. https://doi.org/10.1145/360018.360022.

Open AI. 2024a. *Sora.* https://openai.com/de-DE/sora/. Accessed July 16, 2025.

———. 2024b. *Learning to Reason with LLMs.* https://openai.com/de-DE/index/learning-to-reason-with-llms/, September. Accessed August 5, 2025.

Paritosh, Praveen, Ka Wong, and Kurt Bollacker. 2023. *System 2 Is What We Need.* https://system2.ai/p/system-2-is-what-we-need. Accessed August 3, 2025.

Patil, Avinash, and Aryan Jadon. 2025. *Advancing Reasoning in Large Language Models: Promising Methods and Approaches,* arXiv:2502.03671, May. Accessed July 16, 2025. https://doi.org/10.48550/arXiv.2502.03671. arXiv: 2502.03671 [cs].

Pearl, Judea. 2009. *Causality.* Cambridge University Press, September. ISBN: 978-0-521-89560-6.

Pearl, Judea. 2019. “The Seven Tools of Causal Inference, with Reflections on Machine Learning.” *Commun. ACM* 62, no. 3 (February): 54–60. ISSN: 0001-0782, accessed July 21, 2025. https://doi.org/10.1145/3241036.

Pendurkar, Sumedh, Taoan Huang, Sven Koenig, and Guni Sharon. 2022. *The (Un)Scalability of Heuristic Approximators for NP-Hard Search Problems,* arXiv:2209.03393, December. Accessed August 7, 2025. https://doi.org/10.48550/arXiv.2209.03393. arXiv: 2209.03393 `[cs]`.

Perković, Gabrijela, Antun Drobnjak, and Ivica Botički. 2024. “Hallucinations in LLMs: Understanding and Addressing Challenges.” In *2024 47th MIPRO ICT and Electronics Convention (MIPRO),* 2084–2088. May. Accessed July 24, 2025. https://doi.org/10.1109/MIPRO60963.2024.10569238.

Qiu, Ruizhong, Zhe Xu, Wenxuan Bao, and Hanghang Tong. 2025. *Ask, and It Shall Be given: On the Turing Completeness of Prompting,* arXiv:2411.01992, February. Accessed June 16, 2025. https://doi.org/10.48550/arXiv.2411.01992. arXiv: 2411.01992 `[cs]`.

Rescorla, Michael. 2024. “The Computational Theory of Mind.” In *The Stanford Encyclopedia of Philosophy,* Winter 2024, edited by Edward N. Zalta and Uri Nodelman. Metaphysics Research Lab, Stanford University. Accessed June 23, 2025.

Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. 2016. *"Why Should I Trust You?": Explaining the Predictions of Any Classifier,* arXiv:1602.04938, August. Accessed July 4, 2025. https://doi.org/10.48550/arXiv.1602.04938. arXiv: 1602.04938 `[cs]`.

Rosenblatt, F. 1958. “The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain.” *Psychological Review* (US) 65 (6): 386–408. ISSN: 1939-1471. https://doi.org/10.1037/h0042519.

Rothschild, Daniel. 2025. *Language and Thought: The View from LLMs,* arXiv:2505.13561, May. Accessed August 4, 2025. https://doi.org/10.48550/arXiv.2505.13561. arXiv: 2505.13561 `[cs]`.

Rule, Joshua S., Steven T. Piantadosi, Andrew Cropper, Kevin Ellis, Maxwell Nye, and Joshua B. Tenenbaum. 2024. “Symbolic Metaprogram Search Improves Learning Efficiency and Explains Rule Learning in Humans.” *Nature Communications* 15, no. 1 (August): 6847. ISSN: 2041-1723, accessed August 7, 2025. https://doi.org/10.1038/s41467-024-50966-x.

Rumelhart, David E, Geoffrey E Hintont, and Ronald J Williams. 1986. “Learning Representations by Back-Propagating Errors.”

Schaeffer, Rylan, Brando Miranda, and Sanmi Koyejo. 2023. “Are Emergent Abilities of Large Language Models a Mirage?” *Advances in Neural Information Processing Systems* 36 (December): 55565–55581. Accessed April 8, 2025.

Schick, Timo, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. *Toolformer: Language Models Can Teach Themselves to Use Tools,* arXiv:2302.04761, February. Accessed August 4, 2025. https://doi.org/10.48550/arXiv.2302.04761. arXiv: 2302.04761 `[cs]`.

Schuurmans, Dale. 2023. *Memory Augmented Large Language Models Are Computationally Universal,* arXiv:2301.04589, January. Accessed July 18, 2025. https://doi.org/10.48550/arXiv.2301.04589. arXiv: 2301.04589 [cs].

Shojaee, Parshin, Iman Mirzadeh, Keivan Alizadeh, Maxwell Horton, Samy Bengio, and Mehrdad Farajtabar. 2025a. "The Illusion of Thinking: Understanding the Strengths and Limitations of Reasoning Models via the Lens of Problem Complexity."

———. 2025b. *The Illusion of Thinking: Understanding the Strengths and Limitations of Reasoning Models via the Lens of Problem Complexity,* arXiv:2506.06941, June. Accessed July 20, 2025. https://doi.org/10.48550/arXiv.2506.06941. arXiv: 2506.06941 [cs].

Silver, David, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, et al. 2016. "Mastering the Game of Go with Deep Neural Networks and Tree Search." *Nature* 529, no. 7587 (January): 484–489. ISSN: 1476-4687, accessed July 16, 2025. https://doi.org/10.1038/nature16961.

Simon, A. 1990. "INVARIANTS OF HUMAN BEHAVIOR." *Annual Review of Psychology* 41 (1): 1–20.

Simon, Herbert Alexander. 1997. *Models of Bounded Rationality: Empirically Grounded Economic Reason.* MIT Press. ISBN: 978-0-262-19372-6.

Song, Rui, Yingji Li, Lida Shi, Fausto Giunchiglia, and Hao Xu. 2024. *Shortcut Learning in In-Context Learning: A Survey,* arXiv:2411.02018, November. Accessed July 16, 2025. https://doi.org/10.48550/arXiv.2411.02018. arXiv: 2411.02018 [cs].

Stanovich, Keith. 2011. *Rationality and the Reflective Mind.* OUP USA, February. ISBN: 978-0-19-534114-0.

Stanovich, Keith E. 2009. *What Intelligence Tests Miss: The Psychology of Rational Thought.* xv, 308. What Intelligence Tests Miss: The Psychology of Rational Thought. New Haven, CT, US: Yale University Press. ISBN: 978-0-300-12385-2 978-0-300-16462-6.

Stevenson, Claire E., Alexandra Pafford, Han L. J. van der Maas, and Melanie Mitchell. 2024. *Can Large Language Models Generalize Analogy Solving like People Can?,* arXiv:2411.02348, November. Accessed February 17, 2025. https://doi.org/10.48550/arXiv.2411.02348. arXiv: 2411.02348 [cs].

Subbarao Kambhampati. 2024. *ACL 2024 Keynote: Can LLMs Reason & Plan?,* August. Accessed February 17, 2025.

Thorstad, David. 2025. "The Complexity–Coherence Trade-Off in Cognition." *Mind* (April): fzaf015. ISSN: 0026-4423, accessed April 24, 2025. https://doi.org/10.1093/mind/fzaf015.

Turing, Alan. 1936. "ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM."

———. 1950. *Computing Machinery and Intelligence.* https://philpapers.org/rec/TURCMA. Accessed July 25, 2025.

Tversky, Amos, and Daniel Kahneman. 1974. "Judgment under Uncertainty: Heuristics and Biases." *Science* 185, no. 4157 (September): 1124–1131. Accessed July 21, 2025. https://doi.org/10.1126/science.185.4157.1124.

———. 1983. "Extensional versus Intuitive Reasoning: The Conjunction Fallacy in Probability Judgment." *Psychological Review* (US) 90 (4): 293–315. ISSN: 1939-1471. https://doi.org/10.1037/0033-295X.90.4.293.

Valiant, Leslie G. 2003. "Three Problems in Computer Science." *Journal of the ACM* 50, no. 1 (January): 96–99. ISSN: 0004-5411, 1557-735X, accessed July 16, 2025. https://doi.org/10.1145/602382.602410.

van Rooij, Iris, Olivia Guest, Federico Adolfi, Ronald de Haan, Antonina Kolokolova, and Patricia Rich. 2024. "Reclaiming AI as a Theoretical Tool for Cognitive Science." *Computational Brain & Behavior* 7, no. 4 (December): 616–636. ISSN: 2522-087X, accessed February 17, 2025. https://doi.org/10.1007/s42113-024-00217-5.

Vapnik, Vladimir. 1995. *The Nature of Statistical Learning Theory.* Second Edition. Springer.

———. 2006. "Noninductive Methods of Inference: Direct Inference Instead of Generalization (2000– . . .)" In *Estimation of Dependences Based on Empirical Data,* edited by Vladimir Vapnik, 459–478. New York, NY: Springer. ISBN: 978-0-387-34239-9, accessed June 26, 2025. https://doi.org/10.1007/0-387-34239-7_13.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. "Attention Is All You Need." In *Advances in Neural Information Processing Systems,* vol. 30. Curran Associates, Inc. Accessed March 24, 2025.

Walraet, Matthieu, and John Tromp. 2016. "A Googolplex of Go Games." In *Computers and Games,* edited by Aske Plaat, Walter Kosters, and Jaap van den Herik, 191–201. Cham: Springer International Publishing. ISBN: 978-3-319-50935-8. https://doi.org/10.1007/978-3-319-50935-8_18.

Webb, Taylor, Keith J. Holyoak, and Hongjing Lu. 2023. "Emergent Analogical Reasoning in Large Language Models." *Nature Human Behaviour* 7, no. 9 (September): 1526–1541. ISSN: 2397-3374, accessed July 28, 2025. https://doi.org/10.1038/s41562-023-01659-w.

Wei, Jason, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, et al. 2022. *Emergent Abilities of Large Language Models,* arXiv:2206.07682, October. Accessed April 8, 2025. https://doi.org/10.48550/arXiv.2206.07682. arXiv: 2206.07682 [cs].

Wei, Jason, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, 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 (December): 24824–24837. Accessed February 24, 2025.

Xu, Fengli, Qianyue Hao, Zefang Zong, Jingwei Wang, Yunke Zhang, Jingyi Wang, Xiaochong Lan, et al. 2025. *Towards Large Reasoning Models: A Survey of Reinforced Reasoning with Large Language Models,* arXiv:2501.09686, January. Accessed February 17, 2025. https://doi.org/10.48550/arXiv.2501.09686. arXiv: 2501.09686 [cs].

Yan, Yang, Yu Lu, Renjun Xu, and Zhenzhong Lan. 2025. *Do PhD-level LLMs Truly Grasp Elementary Addition? Probing Rule Learning vs. Memorization in Large Language Models,* arXiv:2504.05262, April. Accessed August 6, 2025. https://doi.org/10.48550/arXiv.2504.05262. arXiv: 2504.05262 [cs].

Yuan, Yu, Lili Zhao, Kai Zhang, Guangting Zheng, and Qi Liu. 2024. *Do LLMs Overcome Shortcut Learning? An Evaluation of Shortcut Challenges in Large Language Models,* arXiv:2410.13343, October. Accessed July 16, 2025. https://doi.org/10.48550/arXiv.2410.13343. arXiv: 2410.13343 [cs].

Zellinger, Michael J., and Matt Thomson. 2025. *Economic Evaluation of LLMs,* arXiv:2507.03834, July. Accessed August 6, 2025. https://doi.org/10.48550/arXiv.2507.03834. arXiv: 2507.03834 [cs].

Zhang, Chunhui, Yiren Jian, Zhongyu Ouyang, and Soroush Vosoughi. 2024. "Working Memory Identifies Reasoning Limits in Language Models." In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing,* edited by Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, 16896–16922. Miami, Florida, USA: Association for Computational Linguistics, November. Accessed July 18, 2025. https://doi.org/10.18653/v1/2024.emnlp-main.938.

Zhou, Denny, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, et al. 2023. *Least-to-Most Prompting Enables Complex Reasoning in Large Language Models,* arXiv:2205.10625, April. Accessed July 18, 2025. https://doi.org/10.48550/arXiv.2205.10625. arXiv: 2205.10625 [cs].

Zhou, Hattie, Arwen Bradley, Etai Littwin, Noam Razin, Omid Saremi, Josh Susskind, Samy Bengio, and Preetum Nakkiran. 2023. *What Algorithms Can Transformers Learn? A Study in Length Generalization,* arXiv:2310.16028, October. Accessed August 5, 2025. https://doi.org/10.48550/arXiv.2310.16028. arXiv: 2310.16028 [cs].

Chapter 5

Appendix

5.1 List of Abbreviations

Abbr.	**Term/Concept**
AI	Artificial Intelligence
CLT	Computational Learning Theory
CoT	Chain-of-Thought
DPT	Dual Process Theory
LLM	Large Language Model
LRM	Large Reasoning Model

5.2 SET-COVER

5.2.1 The Problem

Input: A finite universe $U = e_1, ..., e_n$, where $|U| = n$ and a set S of subsets of U, where $|S| = m$.

Output: The smallest subset S' of S so that their union contains every element of U (the set is covered).

5.2.2 The Exact Algorithm

Strategy: The most straightforward algorithm to find the exact solution for SET-COVER is to exhaustively explore a binary search-tree, where every node represents an element of S, for which we can either decide to include it in S' or not. For every leaf, we denote whether the union over the nodes in its branch covers U. Out of the candidate sets that fulfill this condition, we output the smallest one as S'.

Cost-Accuracy: Because this requires iteration over 2^m leaves (and otherwise only linear operations like intersecting sets, counting and finding the smallest number), this algorithm runs in $\mathcal{O}(2^m)$.

5.2.3 The Greedy Algorithm

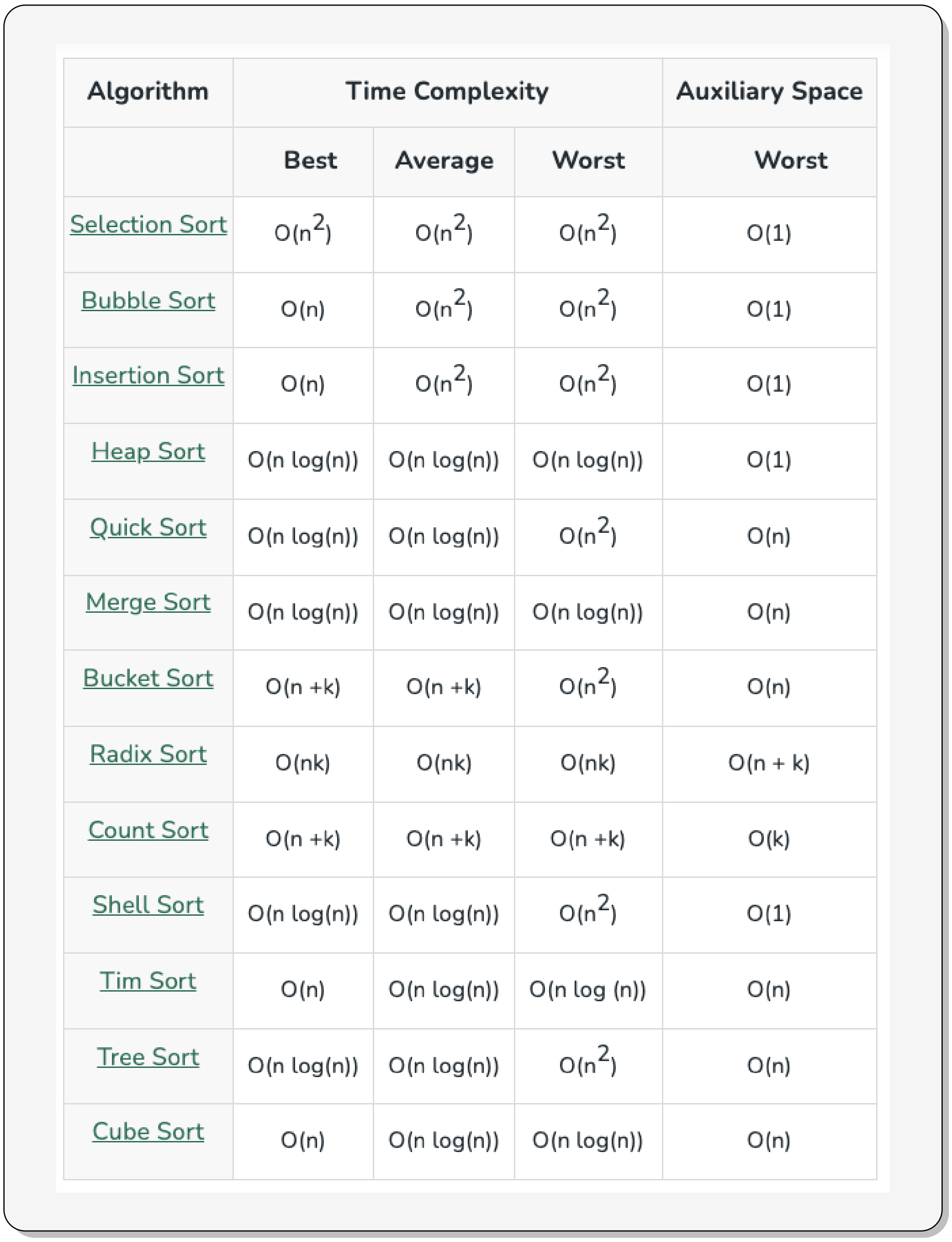
Strategy: Iterative over S and include the set in S' that covers the most uncovered elements from U.
Cost-Accuracy: The greedy algorithm iterates over S at most n times, so it takes $\mathcal{O}(n \cdot m)$.

Optimality: This algorithm guarantees an approximation factor of $1 + ln(n)$ (Chvatal 1979). Assuming P $\neq$ NP, there exists no better polynomial-time approximation of the problem Feige (1998).

5.2.4 Cost Comparison

Assumption: For the sake of simplicity, since there are two parameters that measure the input length n and m, we relate them for a simple comparison of run times and assume that m is $\mathcal{O}(n)$. This allows us to illustrate an example on the lower end of possible problem complexity. In this case, the exact algorithm takes $\mathcal{O}(2^n)$ and the greedy algorithm $\mathcal{O}(n^2)$.

5.3 The Time and Space Complexities of Sorting Algorithms



Algorithm	Time Complexity			Auxiliary Space
	Best	Average	Worst	Worst
Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$	O(1)
Bubble Sort	O(n)	$O(n^2)$	$O(n^2)$	O(1)
Insertion Sort	O(n)	$O(n^2)$	$O(n^2)$	O(1)
Heap Sort	O(n log(n))	O(n log(n))	O(n log(n))	O(1)
Quick Sort	O(n log(n))	O(n log(n))	$O(n^2)$	O(n)
Merge Sort	O(n log(n))	O(n log(n))	O(n log(n))	O(n)
Bucket Sort	O(n +k)	O(n +k)	$O(n^2)$	O(n)
Radix Sort	O(nk)	O(nk)	O(nk)	O(n + k)
Count Sort	O(n +k)	O(n +k)	O(n +k)	O(k)
Shell Sort	O(n log(n))	O(n log(n))	$O(n^2)$	O(1)
Tim Sort	O(n)	O(n log(n))	O(n log (n))	O(n)
Tree Sort	O(n log(n))	O(n log(n))	$O(n^2)$	O(n)
Cube Sort	O(n)	O(n log(n))	O(n log(n))	O(n)

Figure 5.1. Time and space complexities of different sorting algorithms. Screenshot from *Geeks for Geeks*, https://www.geeksforgeeks.org/dsa/time-complexities-of-all-sorting-algorithms/

5.4 Dual Process Original Characterization

Type 1 process (intuitive)	**Type 2 process (reflective)**
Defining features	
Does not require working memory	*Requires working memory*
Autonomous	*Cognitive decoupling; mental simulation*
Typical correlates	
Fast	Slow
High capacity	Capacity limited
Parallel	Serial
Nonconscious	Conscious
Biased responses	Normative responses
Contextualized	Abstract
Automatic	Controlled
Associative	Rule-based
Experience-based decision making	Consequential decision making
Independent of cognitive ability	Correlated with cognitive ability
System 1 (old mind)	**System 2 (new mind)**
Evolved early	Evolved late
Similar to animal cognition	Distinctively human
Implicit knowledge	Explicit knowledge
Basic emotions	Complex emotions

Figure 5.2. Original table with collection of System 1 and System 2 characterizations from Evans and Stanovich 2013, only adjusted for formatting.