Evaluating the Apperception system in the context of explainable AI

Tara Pesman

A thesis presented for the degree of
Master of Artificial Intelligence

First examiner: Rosalie Iemhoff
Second examiner: Jan Broersen

Graduate School of Natural Sciences
Utrecht University
The Netherlands
January 31, 2022

Evaluating the Apperception system in the context of explainable AI

Tara Pesman

Abstract

Recently, Evans et al. published[1] *the Apperception system*: a formalization (the Apperception model) and accompanying implementation (the Apperception engine) of the intuitive notion of 'making sense', which involves the construction of a symbolic causal theory that explains the inputted sensory sequence and satisfies a set of unity conditions inspired by Kant. This work is particularly interesting when placed in the context of explainable AI: In this thesis, I discuss the history, characteristics, and (dis)advantages of classical AI approaches and modern machine learning approaches. The latter have the disadvantage of being a black box, which prevents these approaches from being able to guarantee desirable ethical properties. This problem motivates the research field of explainable AI: the effort of designing AI systems which can compete with modern machine learning approaches in performance, but that are nonetheless explainable (a white box: understandable by humans). I discuss the field in depth, elaborating on which ethical properties are desirable, and which general methods are employed. Then, I give a synopsis of the Apperception system, and assess whether it reaches its objective of formalizing the intuitive notion of 'making sense', after which I place the system in the context of explainable AI, and discuss whether it reaches the goal of combining high performance and explainability. Additionally, I explore potential extensions to the language of the model (specifically to the rules it learns) with the intention of increasing the quality of explanation produced by the system. I conclude that, given the pioneering nature of their work, Evans et al. come close to the goals of creating a high-performing, explainable system that formalizes 'making sense', though there are many improvements that may be made. It is (thus) a good (early) attempt at building explainable AI, and may serve as a foundation for future research in this field.

[1] Richard Evans et al."Making sense of sensory input". In: *Artificial Intelligence* 293 (2021), p. 103438.

Acknowledgements

I would like to express my deep gratitude to my supervisors, who have been there for me throughout this project: Rosalie and Jan. Their genuine interest, expert advice, and support have made all the difference; both for the end result, and for my personal process towards it. As I have written this thesis during especially demanding times (the covid-19 pandemic), Rosalie and Jan's enthusiasm, never-dwindeling interest and optimism, and empathy and flexibility have been invaluable. It was an absolute pleasure working with them.

Contents

1 Introduction

In April 2021, Richard Evans, José Hernández-Oralloc, Johannes Welbla, Pushmeet Kohlia, and Marek Sergot published their paper "Making sense of sensory input", in which they present a formalization and accompanying implementation of what it means for a machine to 'make sense' of sensory data [1]. In their formalization, which I shall refer to as the *Apperception model*, making sense involves constructing a symbolic causal theory that explains the given sensory sequence and satisfies a set of unity conditions. Their implementation of this model, henceforth the *Apperception engine*, synthesizes an extended Datalog program that produces a trace that explains the given sensory sequence as well as satisfies the unity conditions. This way the engine is able to (simultaneously) predict, impute, and retrodict sensory data that is missing from the given sensory sequence. I will refer to the Apperception model and Apperception engine combined as the *Apperception system*.

This work is particularly interesting when placed in the research context of *explainable AI*: the effort of designing AI systems whose inner workings or outputs can be understood by humans. Generally speaking, modern machine learning approaches are powerful in performance, but they are black boxes: they give answers, but the processes that lead to those answers cannot be observed. Classical artificial intelligence approaches, on the other hand, are less powerful in performance, but are white boxes: the processes that lead to their answers can be observed and thus inspected. The effort of the research field of explainable AI is to create systems that can compete in performance with black box systems, but that are also transparent, like white box systems. The Apperception is an attempt at such a system: it is a classical AI system, so it is transparent, but has the ability to learn, similarly to modern machine learning approaches.

In this thesis, I will investigate the Apperception system in two ways. First, I will place the Apperception system in the context of the research field of explainable AI, answering the question: '**What are the strengths and weaknesses of the Apperception system when evaluated in the context of explainable AI?**' Second, I will investigate possible enrichments of the language of the Apperception model. Part of the Apperception model is a set of rules. These rules have the form of definite clauses, however, it is interesting to investigate whether extending these rules (to include i.a. constants, negation, or disjunction) could be beneficial in the context of explainable AI; i.e. whether these extensions would make the Apperception system more explainable. For this second investigation, I will answer the question: '**What are potentially useful extensions to the language of the Apperception model in the context of explainable AI?**'

I will start by describing the historical path, as well as characteristics and (dis)advantages, of classical AI on the one hand, and modern machine learning on the other (chapter 2). Then, I will discuss the research field of explainable AI: its motivation, its methods of achieving XAI, and potential downsides it might encounter (chapter 3). Next, I will describe the Apperception system (chapter 4), as well as evaluate it in the context of the research field of explainable AI (chapter 5). Finally, I will discuss possible ways of enriching the language of the Apperception model (chapter 6) before ending with some concluding remarks (chapter 7).

2 Classical AI & modern machine learning

In this chapter, I will discuss two approaches to building artificial intelligence, viz. classical AI and modern machine learning. I will start with their history (2.1), before I will discuss their characteristics (2.2) and (dis)advantages (2.3).

2.1 History

In this section, I will give a brief overview of the history of Artificial Intelligence with a focus on the different paths therein of logical, knowledge-based approaches on the one hand, and machine learning approaches on the other[1].

2.1.1 AI in the early years

Artificial Intelligence (AI) as an academic discipline was founded in 1956, at a summer conference held at Dartmouth College. The organizing researchers, Marvin Minsky, John McCarthy, Claude Shannon, and Nathan Rochester, wrote in the conference proposal that their quest was based on "the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it" [3, p.64-65]. The goal they set for the conference was "to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves" [3, p.64-65]. Many of the attendees were optimistic about the prospects of reaching this goal, and the predictions made by some set the bar high for the new field [3, p.69]. Although the conference did not lead to any major breakthroughs [3, p.65], it is considered to be the birth of the research field of AI as it gave the field its name, its mission, its first success[2], and its major players [4].

In the decade following the conference, AI researchers used both logical, knowledge-based approaches, and machine learning approaches (based on statistical learning techniques) in the form of 'neural networks' to try and create intelligent machines. The former were inspired by philosophical and psychological accounts of the inner workings of human reasoning processes (the metaphorical 'software' of human intelligence); the latter by the biological architecture of the human brain (the 'hardware'). These two types of approaches are representative of the two sides of the ongoing debate in cognitive science concerning whether human cognition is symbolic or connectionistic: whereas the human *mind* is undoubtably (partially) symbolic, the human *brain* is connectionistic [2, p.1].

Unfortunately, research in machine learning methods encountered serious difficulties early on: in order to perform well, these methods needed access to large amounts of data, as well

[1] For those who are interested in reading about the dichotomy between classical AI and modern ML in more detail, I recommend [2].

[2] Attendees Allen Newell and Herbert Simon from Carnegie Tech presented a mathematical theorem-proving system called the Logic Theorist (LT) [3, p.64-65].

as computing power to process that data, neither of which was available at the time. [3, p.778] This led to machine learning methods falling from of grace in the research field of AI, leaving the main focus on logical, knowledge-based approaches, which had been off to a better start. Formal logic techniques have a long history, going back to the ancient Greeks [5], and in the decade before the Dartmouth conference they were used to build the first programmable, digital computer [4]. This invention of a machine that could manipulate symbols provided inspiration for the very idea of an artificial intelligence. It was thus only natural to approach research in artificial intelligence from a symbolic viewpoint. When reading the Dartmouth conference goal stated above, it is clear that the authors had indeed this kind of symbolic intelligence in mind.

For nearly thirty years, these logical, knowledge-based approaches remained the dominant methodology in AI research. In the first years of AI research (1952-1969), great expectations and enthusiasm prevailed, however, in the decade following (1966-1973), failure to meet those expectations and other difficulties encountered led to a reality check and the first 'AI winter' [3, p.65, p.69], i.e. period of lower research activitities due to low funding, from 1974[3] to 1981[4]. Unfortunately, a second AI winter started around a decade later, when many companies could not deliver on their promises and went bankrupt [3, p.74].

2.1.2 A new era of AI

After machine learning methods had fallen from grace in the field of AI, statistical learning techniques continued development in research areas outside of AI, such as in pattern recognition and information retrieval systems [6, p.755]. By the mid 1980's, data recording and storage, as well as the computational power to leverage that data, had increased enough for statistical learning techniques to perform sufficiently to be operationally useful [7, p.25]. From this time onwards, the amount of research into machine learning methods has been steadily increasing, as has the level of performance achieved by these methods, which helped AI recover commercial attractiveness [3, p.78].

Today, both logical, knowledge-based approaches and machine learning approaches are used in AI research, however, the latter have taken precedence in the last thirty years: they are more popular in academia (in terms of amount of research, interdisciplinarity of researchers working on them, performance, and diversity of application domains) and (more) well-known in mainstream media (*machine learning*, as well as its subset *deep learning* and deep learning's exemplar *neural networks*, have become common terms). An important factor in the shift to machine learning methods from logical, knowledge-based methods is the demand for learning based on, not just any data, but specifically 'real-world' data, i.e. data that is unstructured and may contain noise, errors, or ambiguities, such as typically found on the world wide web. Whereas logical, knowledge-based approaches struggle with this kind of data, machine learning methods have demonstrated to be well-equipped for dealing with real-world messiness. However, despite the increasing success of machine learning methods, logical, knowledge-based systems are far from redundant, as these have their own specialties [8, p.96], i.a. tasks involving structured knowledge or requiring built-in pre-knowledge, and data-scarce tasks (this will be discussed at greater length in section 2.3). Recently, efforts have been made in combining the two approaches, resulting in hybrid systems that feature a mix of advantages of both (this will be discussed at greater length in section 2.3.5).

In the rest of this section, I will discuss the characteristics and (dis)advantages of classical AI and modern ML.

[3] In 1973, the Lighthill report was published: this report, commissioned by the British Science Research Council, gave a critical evaluation of the academic field of AI, and was the basis on which the UK government cut AI funding for all universities except for two. [3, p.70].

[4] In that year, Japan launched a 10-year funding project for AI, to which i.a. the UK and US responded by reinstating AI funding as well [3, p.74].

2.2 Introduction classical AI & modern machine learning

In this section and the next, I discuss the dichotomy in the field of AI of logical, knowledge-based approaches on the one hand, which I shall refer to as *classical artificial intelligence*[5] (*classical AI* for short), and statistical learning approaches, which I shall refer to as *modern machine learning* (*modern ML* for short) on the other[6]. I make specifically this distinction, since I would like to contrast the methods that have been most popular in the field of AI from its conception in the mid 1950's until the mid 1980's (viz. classical AI methods), with the methods that have since then risen to dominance (viz. modern ML methods).

2.2.1 Classical AI

As mentioned (section 2.1), classical AI approaches are based on the hypothesized inner workings of human reasoning processes. The assumption underlying these approaches is described by Allen Newell and Herbert A. Simon in 1976 in their Physical Symbol System Hypothesis (PSSH): "a physical symbol system has the necessary and sufficient means for general intelligent action" [10, p.116]. In other words, it is the assumption that intelligence in any system (be it human or machine) emerges from the manipulation of data structures composed of symbols [6, p.18]. This idea can already be recognized in the Dartmouth conference's goal description (quoted in section 2.1) from 1955.

Classical AI works "by carrying out a series of logic-like reasoning steps over language-like representations" [11, p.17]. These representations, which are "typically propositional in character" and "assert that certain relations hold between certain objects, while each reasoning step computes a further set of relations that follow from those already established, according to a formally specified set of inference rules" [11, p.17, emphasis deleted]. The objects, initial relations, and inference rules that are required to get the system started are manually specified and fed into the system beforehand [7, p.25].

Characteristics

The hallmark characteristic of classical AI systems is their **symbolic representation**, giving them a **strong language bias**. As a corollary, knowledge in classical AI systems is stored **locally**, as opposed to being distributed (e.g. throughout a network, as is the case in modern ML systems). Additionally, the use of symbols makes the representations language-like [11, p.17]—a likeness which is strengthened by the fact that these representations also conform to the principle of **compositionality**, in the sense that "the denotation of a representation is a function of the denotation of its parts and the way those parts are combined" [11, p.18]. For example, in a system based on formal logic, the elementary parts are symbols denoting objects and relations, which combine to form propositions, which can be further combined using logical connectives such as *and* and *or* [11, p.18].

Another defining characteristic of classical AI systems is that they are **rule-driven**: their reasoning processes are dictated by **explicitly defined logics** [7, p.25], which have been **hand-engineered** and hard-coded into the system by the designers. These explicit logics define the series of steps that classical AI systems carry out, which makes them **top-down** systems [12, p.505] that perform **serial computation** [13, p.246]. Finally, the fact that the reasoning processes in classical AI systems are based on explicit logics, makes these systems

[5] Another term that is sometimes used for describing this category of approaches is "Good Old-Fashioned AI", or "GOFAI" for short, which was coined by John Haugeland [9].

[6] A similar dichotomy discussed in the literature consists of symbolic approaches on the one hand and connectionist approaches on the other. (e.g. in [2]).

transparent[7], and they can thus be called 'white boxes'[8].

2.2.2 Modern ML

As discussed (section 2.1), there is an ongoing debate in cognitive science whether human cognition is symbolic (as is the human mind, at least partially) or connectionistic (as is the human brain). Classical AI approaches are inspired by the reasoning processes of the human mind, and are thus based on the assumption that intelligence can be achieved using symbolic representation. This assumption, however, known as the PSSH (see section 2.2.1), is far from undisputed and has been challenged from different directions [16]. Modern ML approaches, in contrast, are inspired by the biological architecture of the human brain, and are thus based on the assumption that intelligence can be achieved using distributed representation[9].

The term *machine learning* refers to the ability of a system to acquire its own knowledge by extracting patterns from raw data [18, p.2-3]. This ability is at the heart of what differentiates modern ML from classical AI (together with differences in representation of knowledge): the former acquires knowledge autonomously (i.e. 'learns'), whereas the latter exhibits intelligent behavior by virtue of human-engineered knowledge that is hard-coded into the system [19, p.3]. It is interesting to note that the terms 'artificial intelligence' and 'intelligent' are often mistakenly seen as synonyms for 'machine learning' and 'learning' respectively [20]. This identification of intelligence with the ability to learn is a relatively recent phenomenon in the field of AI and is a byproduct of the success and consequent popularity of machine learning techniques. However, decision tasks, as done by modern ML systems, are only one part of intelligent behavior: reasoning tasks, as done by classical AI systems, are another, and at least equally[10] important.

Any technique that enables a machine to learn falls within the field of machine learning, however, the most well-known implementation of machine learning is in the form of *deep learning*. In deep learning, hypotheses are represented as graphs, i.e. as networks, consisting of a large number of nodes that work in parallel and exchange information via weighted edges that connect them [3, p.1378]. The adjective 'deep' refers to the fact that these networks are organized in multiple layers, meaning computation paths between inputs and outputs have multiple steps [3, p.1378]. Since these networks were originally inspired by analogy with the network of neurons in the brain, they are often called 'neural networks' [3, p.1378].

Types of modern ML

In contrast to classical AI, modern ML can be split into three main types, based on the kind of feedback that the machine receives in addition to the inputs: *supervised learning*, *unsupervised learning*, and *reinforcement learning* [6, p.650].

In supervised learning, the machines receives inputs accompanied by feedback in the form of outputs. The task is to learn a function that maps from input to output. For example: the

[7] There is no one agreed-upon definition of *transparency*, as its meaning depends heavily on the context in which it is used: when used by system engineers, aspects of transparency such as traceability and verification may be what is meant, whereas for use by ordinary people, it may be aspects such as explainability or intelligibility [14]. Moreover, often the concept is used without being explicitly defined, leaving it unclear what is meant, to whom it relates, and to what extent it is beneficial [15, p.3336]. However, what I refer to here when I say 'classical AI is transparent' is transparency as counterpart of complete opacity, i.e. transparency as some kind of understandability of a system's inner workings. This definition is very broad and rather unspecified, but sufficient for my purposes here.

[8] In contrast to the term 'black box', used to describe processes whose inner workings cannot be observed.

[9] Distributed *representation* is not to be confused with distributed *knowledge*. The latter denotes the occurance in symbolic systems where different pieces of information are stored in multiple locations, yet are combined into one base of distributed knowledge (such as in a multi-agent system). The former, on the other hand, denotes the representing of one piece of knowledge not in a single location, but as a pattern of relationships between locations: this is the natural product of connectionist training methods, as used in modern ML approaches (for more information, see [17]).

[10] If not more important: as described in section 2.1, reasoning tasks used to be the main focus of the field of AI.

input is camera images, and the output is a labeling of those images, e.g. 'bus', 'pedestrian'. After having been trained on example input-output pairs, the machine has learned a function that, given some input, produces an output (e.g. an appriopriate labeling).

In unsupervised learning, the machine receives inputs without any explicit feedback. The goal is to find structure in the data based on heuristics, such as proximity, instead of based on predetermined characteristics as in supervised learning tasks [7, p.25-26]. The most common unsupervised learning task is called *clustering*, in which the machine is trying to find potentially useful groups (clusters) in the data. Here the question the machine tries to answer is which data can be seen as clustered together, instead of answering the question what a cluster might mean (i.e. what label it might be given) as is the case in supervised learning. For example: the input are millions of images from the internet, and the output is a group of similar images that a human might label as 'cats'.

In reinforcement learning, the machine receives inputs accompanied by feedback in the form of rewards or punishments. Given the input, the machine produces an output, which is then reinforced by a reward (if favorable) or discouraged by punishment (if unfavorable). For example: the input is a game of chess (rules, moves by opponent, etc.), and the output is either a victory (reward) or a defeat (punishment). Learning this way is similar to how humans learn through trial-and-error.

In short, supervised learning can be said to be task-driven, unsupervised learning to be data-driven, and reinforcement learning to be based on learning from errors.

Characteristics

As mentioned above, the main characteristic that sets modern ML approaches apart from classical AI approaches is the ability to **learn** knowledge autonomously. Since their learning is guided by the input data, ML approaches are **data-driven**, and thus **bottom-up**, approaches. In contrast to knowledge in classical AI, knowledge learned by modern ML is not represented locally, in symbols, but is represented **distributed**ly, in terms of a set of numbers, vectors, matrixes, or tensors [2]: in neural networks, for example, the learned knowledge is distributed over the network in the form of the matrix of all edge weights. Knowledge learned in distributed representation is often ***entangled***: even if the input data is generated by an underlying process with a number of independent variables, modern ML systems have the tendency to learn latent representations of the data in which those variables are entangled [11, p.18]. As a consequence, modern ML systems can be said to exhibit "**implicit logics**" [7, p.25], in the sense that the patterns they find in the data and the generalizations they make of those patterns are encoded implicitly in the systems' numerical parameters (e.g. the weights in neural networks) [21, p.4]. Because of these implicit logics, modern ML systems are **opaque**[11] systems and can thus be called 'black boxes' [11, p.17]. A final corollary of distributed representation is the possibility for modern ML systems to perform **parallel computation** [13, p.246], e.g. in the form of nodes in neural networks that work in parallel.

2.3 Comparison classical AI & modern ML

In this section, I discuss the advantages and disadvantages of classical AI approaches and modern ML approaches, focusing on those aspects in which the two differ from each other.

[11] Similarly to transparency (see footnote 7), the concept of *opacity* can be used to refer to different abstraction levels or aspects of systems. What I refer to here when I say 'modern ML is opaque' is opacity as unobservability of the inner workings of the system. This, again, is a very broad and rather unspecified definition, but sufficient for my purposes here.

2.3.1 Advantages classical AI

To reiterate (2.2.1), classical AI systems are characterized by their use of symbolic representation that is locally stored, compositional, and has a strong language bias; by being rule-driven (their reasoning processes are dictated by hand-engineered explicit logics), which makes them top-down systems; by their serial computation; and by being transparent.

These characteristics make that classical AI systems provide what may be called *descriptions* (as opposed to mere *discriminations* that modern ML systems provide) [21, p.viii] that are **human-readable**, which brings many corollary advantages, such as being inspectable, and verifiable [1, p.2] (this will be discussed in more detail in 3). Human-readable systems are also **interpretable**, in the sense that their inner workings may be understood, i.e. that it can be understood how the output was produced from the input, as well as how the output changes if the input changes [3, p.1313].

Representing knowledge symbolically, and thus locally, gives classical AI systems the advantage of being **modular**, which allows for **pre-knowledge** to be built into the system, enriching it with a domain-specific model or other background knowledge, such as the explicit logics that dictate classical AI systems' reasoning processes. This makes the system more similar to humans, who also use abstract mental models for their reasoning.

A system using compositional representation is at an advantage when it is confronted by a world that itself exhibits combinatorial structure [11, p.18]. In that case, the representation allows for the possibility to form abstractions and to **generalize** from the system's own experience, since the representations of familiar objects and relationships can enter into new combinations [11, p.18]. Additionally, since relying on compositional representation is a hallmark characteristic of human cognition [22], endowing a system with compositional representation would be the natural choice for building human-like AI.

The combination of a strong language bias and domain-specific pre-knowledge provides a **strong inductive bias** in classical AI systems [23, p.4833], which limits the search space and makes these systems impressively **data-efficient**: they are able to predict and generalize based on only a small amount of data [24, p.1], which keeps them **computationally simple**.

Lastly, classical AI systems are **particularly strong in** doing tasks that require **absolute precision** (due to their being rule-driven); doing tasks that involve **reasoning** and representing specific **propositional contents and inferential relations** between those contents (due to their symbolic representation and being rule-driven) [8, p.93, 95–96]; modeling multi-level **hierarchy and sequential order** (due to their symbolic representation and serial computation); and modeling **abstract problems** (due to their symbolic representation) [25, p.3].

2.3.2 Disadvantages classical AI

Classical AI systems limit the representation of problems to compositions of **symbols**, which can be combined in a language-like fashion. As described earlier, this predetermined form allows for human-readability, however, the flip side of the coin is that it is **not always easy or even possible to represent a problem symbolically** [8, p.94]. In particular tasks that rely on procedural or implicit knowledge, such as sensory or motor processes, are difficult to represent symbolically. Examples of this kind of unstructured data include images (which are necessary for machine vision), audio (which is necessary for speech and voice recognition), and text (which is necessary for machine translation). Since these kinds of tasks happen to be tasks that humans are good at, this presents a problem when attempting to build human-like AI using only classical AI.

In addition to the limitation that the data needs to (have been made to) fit into the symbolic representation scheme, another limitation of classical AI when it comes to data is the amount

it can handle. Due to their serial computation, classical AI systems have **low computational power**, i.e. low speed and energy efficiency in performing operations, which makes them **unable to deal with large amounts of data**. As a result, classical AI systems are mostly applied to highly restricted domains.

A third data-related disadvantage of classical AI systems is their limitation concerning the **quality of data** they can deal with. Data containing noise, errors, or ambiguities lead to great difficulties, as classical AI systems do not naturally[12] show *graceful degradation*: the feature that "imperfections in the data lead to proportionally imperfect but often acceptable performance" [8, p.94]. Instead, small perturbations of the data can lead to large changes in the performance of a classical AI system. This difficulty has also been called the *brittleness problem*[13] [8, p.93], and it is the other side of the coin of the great precision classical AI systems demonstrate.

Lastly, as discussed, an advantage of classical AI systems is the possibility to build-in pre-knowledge in the form of a domain-specific model or other background information. The flip side of the coin is that **pre-knowledge** is not merely a possibility, but a **necessity**: a classical AI system is built on top of some initial (domain-specific) information, which needs to be **hand-engineered** (as discussed in 2.2.1). The difficulty, apart from the sheer time and effort required, is that this knowledge is not always available, nor easy to put into the required symbolic representation.

2.3.3 Advantages modern ML

To reiterate (2.2.2), modern ML systems are characterized by their use of distributed representation, which leads to them learning entangled knowledge and thus exhibiting implicit logics, making them opaque systems; by their ability to perform parallel computation; and by being data-driven, which makes them bottom-up systems.

Where classical AI approaches frequently encounter difficulties when trying to fit information into symbolic, local representation, modern ML approaches enjoy the advantage of the **high representational power** of numerical, distributed knowledge. Generally, this is a more powerful representation, since knowledge needs not be represented in kernels of meaning that have a name and are placed in a web of relationships to each other, which can be difficult in many domains. Consequently, modern ML systems can represent almost any kind of data, including unstructured data [27, p.1], as well as learn almost any type of relation, including highly non-linear correlations [7, p.25], both of which classical AI systems struggle with.

In contrast to classical AI systems, which need manually specified pre-knowledge to get started, modern ML systems are data-driven and so **no hand-engineering** of any of the features or rules of a system is necessary. In fact, it is precisely the ability to discover features in data and learn the rules that guide those features' interactions without any human intervention that is one of the major strengths of modern ML systems [11, p.17].

As discussed (2.3.2), due to the serial computation of classical AI approaches they are limited in the amount and quality of data they can work with. In contrast, modern ML approaches perform parallel computation, which gives them the advantage of **high computational power**, making that they **can deal with large amounts of data**. Additionally, they are also able to deal with **noisy, erroneous, and ambiguous data**, making them suitable for working with data reflecting the messiness of the real world, such as images, audio, and text (as is essential for tasks such as machine vision, speech and voice recognition, and machine translation [7, p.26]). The reason for this flexibility is that, where classical AI

[12] There are ways in which classical AI systems may be enhanced to improve on how graceful their performance degrades: see [8, p.94].

[13] The brittleness problem is more generally defined as the difficulty in which a "system functions well within some bounds and poorly outside of those bounds" [26, p.1]. More on this problem and on the *frame problem* (a special case of the brittleness problem) can be found in [8, p.93-94].

systems have the advantage of high precision (which comes with the downside of brittleness), modern ML systems have the counterpart advantage of *graceful degradation* (which comes with the downside of less precision), i.e. their performance is not significantly impacted by (reasonably small) perturbations in data, thus making them resilient to noise, errors, and ambiguities.

2.3.4 Disadvantages modern ML

Most disadvantages of modern ML systems can be grouped into three general categories: uninterpretability, poor generalization, and data-inefficiency [23, p.4833] [11, p.17].

First, modern ML systems are ***uninterpretable***, in the sense that the system's inner workings cannot be understood. As discussed, the knowledge modern ML systems learn tends to be *entangled* due to their distributed representation, leading them to exhibit implicit logics, which leads to them being opaque systems, also called 'black boxes'. In other words, the high representational power deriving from distributed representation comes at the cost of explanatory power: knowledge learned by modern ML systems is optimized for predictive performance, rather than interpretability, making them complex and consequently **incomprehensible to humans** [7, p.25-26].

Second, modern ML systems suffer from ***poor generalization***. Since the component parts in distributed representations have little or no meaning in isolation, modern ML systems are not compositional [11, p.18], and thus **not modular**. This makes modern ML systems **less suited for use in transfer and lifelong learning** [28, p.20], as well as makes it **difficult to enhance** them with any **domain-specific pre-knowledge**[14] (the other side of the coin of the advantage of not needing any pre-knowledge to get started), making that they perform poorly outside of their comfort zone, i.e. outside of the domain in which they were trained. These two consequences make modern ML approaches typically poor at generalization [30], which is problematic when trying to build human-like AI [30], since "the ability to re-use previously acquired experience and expertise" and "to transfer it to radically different challenges" is a "hallmark of human intelligence" [11, p.17].

Related to their poor generalization, modern ML systems are generally bad at dealing at tasks that involve the binding problem or the occlusion problem. These problems originated in the field of cognitive science [31, p.53], but have proven to be relevant to many other disciplines, including psychology, computational modeling, and philosophy [32, p.7]. The **binding problem** concerns cases "where information from different modalities must somehow be combined into different aspects of one unified object" [1, p.2], e.g. in human vision, when a blue circle and a red square are observed, the two colors and the two shapes are initially processed separately, so the question arises how those different modalities (color and shape) are (correctly) paired again later on in the process. The binding problem is a problem[15] that lies at the very core of neuroscience, since it focuses on fundamental questions concerning "how neurons code the stimuli of the external world, how these stimuli are represented in the brain, and how neurons communicate in general with each other", and even touches on the nature of consciousness, addressing the question "how distributed neural activity (a physiological concept) gives rise to the unity of conscious experience (a psychological concept)" [34]. Since modern ML systems are inspired by the neurological architecture of the human brain, these questions are highly relevant to the study of modern ML as well. For example, the quest to understand how combinatorial structure (an essential part of human cognition) can be instantiated in neural terms is relevant to both research

[14] The closest that modern ML systems come to having pre-knowledge is to be 'pre-trained', e.g. to be trained for facial recognition generally and to then be used for one specific database, which eliminates the need for large amounts of training data from that specific domain in order to perform well. Using these pre-trained systems, modern ML approaches can achieve some form of transfer and lifelong learning, however, even very successful systems have an "extremely narrow sphere of expertise", making the possibilities for extending domains limited. [29]

[15] It has been argued that this is not a single problem, but comprises four distinct subproblems: to read more, see [33].

into the human brain and into modern ML systems based on the human brain [22]. In fact, since combinatorial structure is fundamental for generalization, it has been argued that the binding problem is the main reason for modern ML systems' underperformance when it comes to generalization [30]. The **occlusion problem** concerns cases "in which objects are sometimes visible and sometimes obscured from view" [1, p.3]. For modern ML systems, these cases make certain tasks difficult, such as object tracking and detection[16] [35], object identification[17] [36], and object placement[18] [37, p.648]. In these kinds of tasks, the occlusion of objects leads to impaired performance of modern ML systems, i.a. since it is difficult to enhance them with the pre-knowledge that objects persist over time [1, p.24]. As a final note: classical AI systems are better at dealing with the binding problem and occlusion problem, though it should be noted that the knowledge that gives them an advantage is not learned by the systems themselves, but rather built into them by their designers.

Third, modern ML systems are ***data-inefficient***. The difficulty of building-in any domain-specific pre-knowledge makes that modern ML approaches lack the strong inductive bias that limits the search space in classical AI approaches (as discussed in 2.3.1). As a consequence, modern ML are data-inefficient, relying on large amounts of carefully annotated quality training data to yield good results, and therefore necessarily **computationally complex**. Thus, the use of modern ML approaches is restricted to domains where large amounts of labeled data and computational power are available: this limitation is one of the most mentioned downsides of modern ML. An additional, related downside is the existence of "**adversarial examples**" [38], [39]: data that are designed to manipulate a system's process and thus its output, causing the system to behave in unexpected (and potentially undesirable) ways [40]. Modern ML systems are expected to be robust to small pertubations in their input: for example, in a system designed for object recognition, it is expected that a small[19] pertubation in an image does not change the object category of that image [38, p.2]. However, research has found that by applying imperceptible non-random perturbation to input, and by optimizing to maximize the prediction error of the output, it is possible to arbitrarily change the system's output: in the example above, a slight pertubation to the image would influence the object category of that image [38, p.2]. Modern ML systems are vulnerable to hacking attempts through these pertubations in the data, or "adversarial examples" as they are termed [38, p.2], for two reasons. The first is their need for large amounts of data as input, instead of more controlled input as in the case of classical AI systems, which leaves them exposed to potentially doctored data. The second is their lack of pre-knowledge: when observing a slightly perturbed image (of e.g. a lion), humans (and classical AI systems endowed with an abstract mental model) are able to rely on a set of high-level features (ears, tail, mane, ...) that allows them to "abstract away from low-level arbitrary or incidental details" [29]. Modern ML systems, in contrast, do not have such a built-in model to aid them in their understanding and can hence fall into the trap of adversarial examples. This sensitivity to small changes may be seen as modern ML's version of classical AI's brittleness problem (as discussed in 2.3.2).

2.3.5 Combining classical AI and modern ML

Over the last two decades [28, p.1], AI research has been increasingly interested at the *hybridization* of classical AI approaches and modern ML approaches[20], i.e. combining characteristics of both approaches in one system[21]. The motivation for this movement can be explained by the path of history described earlier (2.1), leading to modern ML systems be-

[16] E.g. pedestrian surveillance, where persons that are partially or fully occluded at certain time steps need to be consistently tracked.

[17] E.g. in image search, where images contain the desired object, but it is partially occluded.

[18] E.g. in augmented reality, where real objects and virtual objects need to be combined into one image with a correct depth hierarchy.

[19] 'Small' in the order of being "typically imperceptible to humans" [29].

[20] For some recent research in this area, see [41], [27], [42], [43], [44], [45], [46], [47], [11].

[21] This can be done either by designing one kind of approach (classical AI or modern ML) which does have some characteristics of the other kind, or by designing a system of which a part is of one kind of approach, and a part is the other kind.

ing applied more frequently and in broader domains every year [27, p.2411], making their downsides pose every bigger problems. Since the shortcomings of modern ML align[22] with the strengths of classical AI [11, p.17] and vice versa, the intuitive solution to the problem is to combine the two, having systems with characteristics of both: e.g. integrating background knowledge (from classical AI) with learning from examples (from modern ML). This would combine the best of both worlds, resulting in systems that have the predicting power of modern ML, but that are also able to reason in the way classical AI can and have the benefits of human interpretability, powerful generalization, and data-efficiency [11, p.21] (the counterparts to modern ML's downsides).

A second reason for combining the two approaches comes from the original goal of AI, viz. to emulate human intelligence. Humans reason and learn using both structured logical knowledge and statistical inference, thus, it is logical when building a machine to be functionally similar to incorporate both of these approaches as well [27, p.2410].

A considerable part of the research into hybrid systems takes place as part of the quest for explainable AI, which I will discuss in the next section.

[22] Some even describe classical AI and modern ML as opposites, in the sense that they are each close to another extreme on the spectrum that runs from fully top-down to fully bottom-up. A hybrid, then, would be an approach that is further away from the extremes and more towards the center of the spectrum: a more equal mix of top-down and bottom-up. [2]

3 Explainable AI

In this chapter, I will discuss the research field of explainable AI, which has developed (partly) as a response to the problems that arise from modern ML systems' uninterpretability. These may be summarized as having an impaired ability to evaluate (and thus guarantee) certain desirable technical and ethical properties; in particular the latter are difficult to formalize or quantify, making it problematic to guarantee them based purely on the output of a system. Instead, if a system is explainable, it *is* possible to guarantee these auxiliary criteria, as the system's process and/or underlying assumptions are transparent and can thus be verified to be sound with respect to the desirable properties.

The chapter is structured as follows: I will start by introducing the research field (3.1), after which I will discuss the desirable ethical properties that cannot be guaranteed in modern ML (3.2), the different methods of designing explainable AI (3.3), and I will end with briefly considering some potential complications that may arise when attempting to build explainable AI (3.4).

3.1 Introduction

As modern ML approaches have taken over precedence in the research field of AI for the last thirty years (as discussed in 2.1), they are applied in ever broader domains [11, p.17] [27, p.2411] [48], including those in which the use of uninterpretable systems has turned out to be undesirable. In applications of modern ML techniques that model broad or abstract phenomena, such as the climate, the economy, or urban traffic [7, p.24] (as was the case for most early applications), there are no significant consequences for incorrect results [49, p.3]. However, in applications that model people, groups, or firms (i.e. legal entities) (as in many novel applications), stakes are higher and the consequences of incorrect results considerable [7, p.24]. Examples of this type of domain include criminal justice, healthcare, welfare, and education [50]. In these high-stake domains, it is crucial that AI systems make correct decisions, both in the performance sense and the ethical sense. In order to evaluate results for correctness, then, AI systems in these domains need not only satisfy standard technical desiderata (such as predictive quality and computability), but also certain ethical desiderata (most noticeably non-discrimination, privacy, and transparency) that are important for decisions concerning legal entities. Since the ethical desiderata are difficult to formalize or quantify [49, p.1, p.3], it is problematic to guarantee them based purely on the output of a system. Instead, it is necessary to evaluate the information underlying the output (e.g. assumptions, deduced rules), however, this cannot be done for systems that are uninterpretable. Guaranteeing the technical desiderata is more difficult for uninterpretable systems as well (albeit to a lesser extent): part of the information that is relevant for evaluating these desiderata can indeed be deduced from the results, but another part is also hidden in the underlying assumptions and rules. Additionally, this difficulty in guaranteeing desiderata is worsening over time, as the gap between what is happening and what we understand about what is happening is growing bigger every year [48].

One solution that is put forward to solve this problem is the development of explainable

systems: since the system's process and/or underlying assumptions are transparent, it can be verified that they are sound with respect to the desirable properties [49, p.1]. Where interpretable systems are able to answer the question 'how was this output produced for this input?', ***explainable*** systems focus more on the question 'why was this output produced for this input?'[1]. This difference in question words, 'why' versus 'how'[2], makes that explainability is both a narrower and a broader concept than interpretability[3]. Narrower (or stronger) in the sense that an explainable system provides information that (usually) gives more clarification than the information provided by interpretable systems: the system is able to give meta-information on its process or on itself as a whole (instead of merely information on its process). This is best illustrated by the cases of explainable systems in which this meta-information takes the form of an extra layer of explanatory information *on top of* the system itself. It is also a broader concept in the sense that an answer to the 'why'-question above may be produced via different routes on different abstraction levels: I will discuss a taxonomy that distinguishes four different forms that an answer may take later in this chapter (3.3). In some cases, a system's interpretability is a sufficient route to its explainability (as insinuated by the '(usually)' above), i.e. it is able to answer the 'why'-question by virtue of having understandable inner workings. In other cases, explainability of a system is achieved via a different route, independent[4] of whether or not that system is also interpretable (e.g. via an extra explanatory layer, as mentioned above). The efforts to develop these explainable systems are brought together in the research field of *explainable AI* (XAI).

There have been calls for XAI the last thirty years [53, p.1057], both from the research and development side, viz. the scientific community, companies (e.g. those in Silicon Valley), as well as from the user and regulations side, viz. the public, law and policy makers [48]. From the law and policy side[5], new legislation requiring XAI has been implemented in response to the ethical issues that arise from the broadening application of modern ML systems that lack important ethical desiderata. In the European Union, for example, the General Data Protection Regulation (GDPR) went into effect as of May 2018. This "set of comprehensive regulations for the collection, storage, and use of personal information" places restrictions on automated individual decision-making that 'signifcantly affect' users [54, p.50]. These restrictions impose that algorithms that make decisions based on user-level predictors provide explanation ("right to explanation") [49, p.2], which has stimulated researchers to design algorithms and evaluation frameworks that enable explanation of results [54, p.50]. This stimulus from legislation adds to the already existing call from the research and development community itself to make modern ML systems more explainable. Researchers both in academia and companies are interested in better understanding the inner workings of the systems they are designing. Not only does this aid in development, it is also necessary to be able to assess the reliability of systems (i.e. to assess whether the correlations used are spurious, non-generalizable, or out-of-date) [7, p.54].

In conclusion, XAI can help to assess various desiderata important for the user and regulations side, as well as improve system performance and trust for the research and development side. Indeed, in the last decade, XAI research has seen an "explosion of interest" [55, p.2], from which it may be concluded that these calls are being heard.

[1] In section 3.3, the concept of explainability will be discussed in more detail. It will become clear that explainable systems may answer other 'why'-questions than just the one mentioned here.

[2] An interesting way of relating 'why' and 'how' with each other is to see them as opposite directions of movement between some goal on the one hand, and the procedure to reach that goal on the other: 'why' points from the procedure to the goal, whereas 'how' points from the goal to the procedure [51]. *Explainability* and *interpretability* relate similarly to each other.

[3] Thus, neither concept subsumes the other.

[4] An example of a system that is not interpretable, but that is nonetheless explainable, is human decision making: human brains are black boxes (and thus uninterpretable), yet human decisions are explainable [52, p.6].

[5] Law was actually one of the first domains to be concerned about modern ML approaches missing the ability to explain [7, p.24-25].

3.2 Desirable ethical properties that modern ML cannot guarantee

As just mentioned, the problems encountered with the ever-broader application of modern ML systems can be reduced to their impaired ability to guarantee certain important desiderata. In this section, I will discuss three ethical desiderata that are well-documented and serve as umbrella terms for most others, viz. *unbiasedness*, *privacy*, and *transparency*.

3.2.1 Unbiasedness

It used to be a common assumption that machines would be the ideal decisions makers, as they would be free from the biases that human decision makers are prone to [7, p.27]. Unfortunately, the machines that have been built so far still base their intelligence in some way or another on human knowledge: either in the form of programmed pre-knowledge (in classical AI systems), or in the form of training data (in modern ML systems). Decisions made by AI can thus not be assumed to be free from human biases.

A system making decisions involving legal entities (i.e. people, groups, or firms) should be unbiased in two ways. First, it should be *non-discriminatory*: it should not hold any illegal biases [7, p.27], i.e. biases concerning relations between "protected characteristics" such as race, gender, pregnancy status, religion, sexuality, and disability [7, p.28]. Even though correlations involving these variables may be found, it is socially unacceptable to re-entrench these existing patterns [7, p.28]. Second, it should be *fair*, i.e. it should not hold any legal but otherwise unwanted biases [7, p.30]: biases based on correlations that are short-lived, arbitrary, or can otherwise be considered to be based on unfair grounds [7, p.30]. An example would be to take an applicant's web browser into account when assessing job suitability: even though this factor is predictively relevant, it may be considered short-lived, arbitrary, and its predictive power might come from an underlying factor of poverty (which in turn might even be a surrogate for a protected characteristic such as race or disability) [7, p.30].

3.2.2 Privacy

Privacy is another well-studied and well-defined desideratum of AI systems and concerns, in this context, the protection of sensitive information in data [49, p.2]. An aspect of modern ML systems that is troublesome with respect to privacy is *profiling*: the process whereby personal data is converted into a group profile, which is then used to predict personal data again [7, p.32]. Privacy concerns include both how the group profile is derived, distributed, and used, and to what extent an individual can control how personal data is collected and processed [7, p.32].

Four issues arising specifically from profiling in modern ML techniques are the following [7, p.33-38]. First, consent has become a "debased currency": the value of giving consent has decreased with the increase in uninformed consent (e.g. due to standard term privacy policies increasing in length to the point of being impractical to read) and manipulated consent (e.g. using screen layout manipulation to nudge users towards giving consent).

Second, issues concerning consent are worsened by so-called "bastard data": profiles and other data products that are born out of linking and transforming data. The creation of bastard data, in turn, motivates the collection of new data, since the integration of data creates new potential purposes. In this way, more and more data is collected and processed, to the point of going well beyond society's idea of privacy. Proponents of bastard data argue that the personal data used are transformed (e.g. through removing obvious identifiers such as names) into data that is non-personal, and thereby fall outside of the scope of data protection law. Opponents have countered this argument with accounts of re-identification.

A third issue arises from this creation of bastard profiles, viz. the protection of "algorithmically assembled groups". A way around privacy laws pertaining to the use of personal data is to inform decisions not on individual profiles, but on group profiles that individuals are linked to. Whereas an individual profile identifies characteristics of a single user and is thus protected by law, group profiles are 'merely' based on relative differences (e.g. the group of users that are 75% more likely to attend a music festival than the rest of the cohort of those users), and thus fall outside of the scope of most[6] data protection laws.

A final issue concerns the ability of the decision systems using (group) profiles to transform "ordinary personal data" into "sensitive personal data": personal data that relate to especially sensitive characteristics such as "race, political opinions, health and sex life, religious and other beliefs, trade union membership and (...) biometric and genetic data" [7, p.36]. The categories of sensitive personal data is protected by law and can only be used after explicit consent has been given, however, the case of using non-protected ordinary personal data to deduce sensitive personal data is underspecified in data protection law.

3.2.3 Transparency

People have been disturbed for a long time by the idea of a machine making decisions for them, without them understanding why or having any influence on the matter [7, p.38]. In the *public* sphere, it is common to have some kind of 'right to transparency', embodying this sentiment, which usually takes the form of 'freedom of information' rights against public and governmental institutions. "Transparency is seen as one of the bastions of democracy, liberal government, accountability and restraint on arbitrary or self-interested exercise of power", as Edwards and Veale note, and rights thereto against any public bodies "enable an informed public debate, generate trust in and legitimacy for the government, as well as allow individual voters to vote with more information" [7, p.39]. In contrast, in the *private* sphere, secrecy (e.g. in the form of commercial and trade secrets) rather than transparency is the default. Nevertheless, in order to be able to challenge a private person or commercial business, an explanation or some form of lesser transparency is necessary: transparency is the first step in the process of accountability[7].

In research and development, be it private or public, some[8] form of transparency is necessary to be able to understand, review, and make effective alterations to modern ML systems, which are essential steps in improving performance of and increasing trust in those systems.In addition to the more intuitive ways, consider as a less obvious example of the ways in which transparency is desirable: the case of adversarial examples. As discussed (section 2.3.4), the need for large amounts of (thus uncontrolled) data as input and the lack of pre-knowledge make modern ML systems vulnerable to hacking attempts through so-called "adversarial examples": data that are designed to manipulate a system's process and thus its output, causing the system to behave in unexpected (and potentially undesirable) ways [40]. To briefly illustrate, two ways in which adversarial examples have caused ethical difficulties are the following. First, adversarial examples can be used to obtain private information about the individuals in a system's training set, leading to the problem of re-identification [7, p.43], as discussed in 3.2.2. Second, adversarial examples can be used to deceive facial detection systems (an increasingly important method of identification), leading the systems to misidentify faces: either by recognizing the face on camera as belonging to someone else, or by not recognizing it as a human face at all [56]. Sensitivity to adversarial examples can be remedied[9] by "adversarial training"[10]: providing more training data, repeatedly exposing

[6] Noticeably, protection for this "group privacy" *has* been included in the GDPR [7, p.35-36].

[7] Where accountibility is defined as "a party being held to account having to justify their actions, field questions from others, and face appropriate consequences" [7, p.41].

[8] What form of transparency is sufficient depends on the specific case: sometimes interpretability will fit purposes best, other times some form of explainability will be what is needed.

[9] This is not a *sinecure* though, since training a system against one kind of attack can weaken it against others. Therefore, other methods to protect a system against adversarial examples are being developed as well. [29].

[10] To be more precise: this method makes use of the deep learning technique called 'generative adversarial

the system to problematic cases (i.e. to adversarial examples), thus allowing it to adjust its parameters [29]. Finding adversarial examples and correcting for them is made difficult by the opacity of modern ML systems [57].

A natural solution to modern ML systems' impaired ability to guarantee transparency, as welll as privacy and unbiasedness, would be to make them explainable: being interpretable is not sufficient, as it is important to be able to evaluate the assumptions and rules underlying a system's process, which cannot always be deduced from information that merely concerns which steps are part of the process.

3.3 The How of XAI

Now that the motivation for XAI has been discussed, the next questions that arise are 'what constitutes an explainable system?', and 'how might one be built?': these questions are the topic of this section.

3.3.1 On the term *explainable*

Similarly to its umbrella term *transparent* (as discussed in footnote 7), the term *explainable* does not have one agreed-upon definition and its meaning is highly context-dependent. However, now that its meaning is becoming more important with the recent increase in interest in XAI, different papers have been written on the topic [55] [52] [49] [15]. What adds to the difficulty of trying to use a concept that is underdefined, is the plethora of terms related to 'explainable', such as 'human-readable', 'understandable', 'observable', 'inspectable', 'transparent', and 'interpretable'. A combination of these terms is used interchangeably by some (e.g. [58] [59]), while others clearly demarcate some of these terms as different from each other (e.g. [55] [49]). I have chosen to take the latter stance, however, I only define those concepts that are relevant to this thesis, viz. transparency[11] (as defined in footnote 7), interpretability (as defined in 2.3.1 and 3.1), and explainability (as defined 3.1). To review, I use ***transparency*** to refer to the counterpart of complete opacity, i.e. to some kind of understandability of (a sufficient[12] part of) the system. The more specific concepts of interpretability and explainability both fall under this broad umbrella term. I use ***interpretability*** to refer to the feature that the system's inner workings may be understood, i.e. that the question '*how* was this output produced for this input?' may be answered. And I use ***explainability*** to refer to the feature that (a sufficient[13] part of) the system may be explained, i.e. that the question '*why* was this output produced for this input?' may be answered. To elaborate: an explainable system is able to summarize the reasons for its behavior or produce insights about the causes of its decisions, as well as provide relevant responses to questions [55, p.2]. These abilities have more explanatory power than the abilities that are required for a system to be called interpretable, and they enable explainable systems to i.a. be audited, defend their actions, and gain the trust of users [55, p.2].

An answer to the 'why'-question that explainable system answer may take different forms: four different ones in the taxonomy of Guidotti et al. [58], which will be discussed later in the next section. From this taxonomy, it will become clear that for a system to be explainable there are cases in which the feature of interpretability (understanding the system's inner workings) is sufficient, however, generally, it is not necessary.

networks' (GANs).

[11] In this thesis, the term 'transparency' is double-loaded: it is used to denote a characteristic of classical AI systems (which is what is referred to in this section), as well as to denote one of the three ethical desiderata discussed. In other sections, which concept is meant can easily be deduced from the context. In this section, this may be less clear. Hence, to avoid confusion: I am referring to transparency in the former sense here.

[12] What constitutes 'a sufficient part', depends on the specific case: as mentioned in footnote 7, this definition is very broad.

[13] See footnote 12.

3.3.2 Different methods to achieve XAI

Different taxonomies of methods for achieving XAI have been proposed. The following taxonomy has been proposed by Guidotti et al. in their recent survey of different methods for explaining black box models [58][14]. There are two broad categories of solutions to the problem of explaining uninterpretable modern ML systems, a.k.a. 'the black box problem', viz. 'explaining the black box', and 'designing a white box'.

The first category, *explaining the black box* involves reverse engineering and is thus sometimes called "*post hoc* explainability" [52, p.4]: it involves extracting information from systems after they have been trained. In this category, there are three subcategories: 'explaining the model', 'explaining the output', and 'inspecting the model'[15]. The choice between which is most appropriate to use for a certain application depends mostly on motivation: interest in explaining the model (more general) versus interest in explaining the output (input-specific)[16]). The first subcategory is that of *explaining the model*. The aim here is to understand the overall logic behind the black box [58, p.93:11], giving broad information about the model that is not input- or output-specific [7, p.55]. There are different ways in which to achieve this, however, Guidotti et al. focus on the method of creating an additional model that mimics the behavior of the black box, but that is also understandable to humans [58, p.93:12], i.e. that is (partially) a white box. The second subcategory is that of *explaining the output*. The aim here is to understand why the black box has produced a certain output given a certain input. The third and final subcategory is that of *inspecting the model*. The aim here is to create some (visual or textual) representation that helps to understand some specific property of the model itself or of its output.

The second category, *designing a white box* (sometimes called 'transparency by design'), aims to create a model that can compete in performance with black boxes, but that itself is a (partially) white box. One method to achieve this is to combine black box techniques and white box techniques, creating a model with high performance that is nonetheless explainable: or put in the terminology of this thesis, this method involves combining classical AI approaches with modern ML approaches, as has been discussed in section 2.3.5. As mentioned there, the shortcomings of modern ML align with the strenghts of classical AI and vice versa, which makes that combining the two approaches does not only solve modern ML's uninterpretability (as is the motivation discussed here), but also potentially solves its other downsides of poor generalization and data-inefficiency. So, if done well, hybridization has the potential of reaching a "sweet spot in the complexity-expressiveness landscape" [28, p.4], remaining understandable and computable (like classical AI), while being able to represent and learn almost anything (like modern ML).

The Apperception system is an example of this second type of approach: I will give an introduction to this system in the next chapter, but before I do so, I will discuss a number of difficulties that may be encountered when attempting to build XAI.

3.4 Complications of XAI

In this section, I will discuss four potential complications of XAI: (i) the challenge of choosing the appropriate trade-off between understandability and completeness of an explanation; (ii) the difficulty of choosing what information to disclose; (iii) it may be difficult or impossible to achieve; (iv) it may be unnecessary or even insufficient.

Explainability is a trade-off between being complete on the one hand, and being understandable on the other [55, p. 2]. When a system's description is complete (i.e. when

[14] Two other taxonomies, for comparison, are [48] and [55].

[15] For a more detailed treatment of these three subcategories and the methods used to achieve their goals, I refer the reader to [58].

[16] Though these two interests require different methods to achieve, they are equally important and complementary [58, p.93:5].

completeness is maximized), there is an information overload (also called 'infobesity'), i.e. there is more information than useful or even than functional. On the other hand, when a system's description is perfectly understandable (i.e. when understandability is maximized), the description is too simplified to give an accurate depiction of what is really happening in the system. The challenge in creating XAI is to optimize the trade-off between these two desirables [55, p. 2]. If an unsuitable choice is made, problems might arise. Having an explanation that is too simplified (i.e. heavily reduced in completeness compared to the system itself) for the sake of understandability may be misleading, especially if the explanation was designed to increase trust or hide undesirable features of the system [55, p. 2]. Conversely, having an explanation that is highly accurate, but therefore not understandable, is self-evidently also problematic.

Another choice in the development of explainable systems that may lead to problems concerns how much and what information is disclosed. For example, naively disclosing the source code as explanation may lead to issues concerning privacy disclosures, as well as create so-called "gaming" strategies[17] [7, p.43]: these are strategies learned by AI systems or developed by people that will maximize performance, without actually solving the problem that the designers of the system intended it to solve [3, p.1830], which can undermine i.a. the system's efficiency and fairness [7, p.43].

A third complication arises when trying to build XAI for tasks for which traditionally modern ML approaches are most fitting: those tasks are usually inherently complex, explaining why classical AI is not chosen [3, p.1315]. For these tasks, it is not guaranteed that a simple (or at least understandable) explanation can be given for every output. Or if it can be given, it might be very difficult to do so.

Finally, sometimes XAI is not the only or the best solution for a certain problem, despite appearing that way. Krishnan challenges the existence and importance of the black box problem [59]. Her argument is that since certain problems (such as systems being unable to guarantee certain desiderata) may be solved by making systems interpretable, they are often equated with the problem of being uninterpretable. Whereas in reality, these are distinct problems and equating them wrongfully reduces the solution space of the former to the solution space of the latter (viz. only solutions involving interpretable systems). To illustrate: whether XAI provides a solution depends on the motivation for building it [3, p.1315]. If the goal is to understand the domain, then explainability will indeed lead to the right kind of understanding. However, if the goal is to have more confidence and trust in a system's performance, then testing might be of more value than an explanation. As Russell & Norvig put it: "Which would you trust: an experimental aircraft that has never flown before but has a detailed explanation of why it is safe, or an aircraft that safely completed 100 previous flights and has been carefully maintained, but comes with no guaranteed explanation?" [3, p.1315].

[17] This is also called "specification gaming". For a good introduction to this concept, I refer the reader to this blog post on the Google DeepMind website: [60].

4 The Apperception system

In the previous chapter, I have described the research field of XAI, including the different approaches that are taken in building XAI. In this chapter, I will discuss one such attempt at building XAI: viz. the Apperception system (both the model and the engine) as described in "Making sense of sensory input", published in April 2021 by Richard Evans, José Hernández-Oralloc, Johannes Welbla, Pushmeet Kohlia, and Marek Sergot [1][1]. Evans et al.'s goal is to create a system that (with further development) can compete in performance with modern ML approaches, but that is nonetheless explainable. The strategy they have chosen to build this XAI is that of 'designing a white box' (as discussed in 3.3).

The chapter is split into two parts. In the first part, I discuss the formalization of the system, viz. the Apperception model (4.1). In the second part, I discuss the implementation of the model, viz. the Apperception engine (4.2).

4.1 Formalization: the Apperception model

In this section, I will discuss the formalization of what it means to 'make sense' of a sensory sequence, viz. the Apperception model. I will start by discussing the high level idea behind the model (4.1.1), after which I will give the formal definitions of the building blocks of the model (4.1.2–4.1.4), and I will end with an example to illustrate the theory (4.1.5).

4.1.1 The idea

The question that Evans et al. try to answer is "What does it mean to 'make sense' of a sensory sequence?" Their aim is to create a system that is able to do more than just 'perceive' data and make predictions based on them (as modern ML systems are able). Instead, they want to create a system that is able to "apperceive" data, i.e. "to assimilate [data] into a larger collection of ideas, to unify the thought with others in a larger, coherent whole" [31, p.3], and make predictions, as well as retrodictions and imputations.

The high level idea of their answer to the above question is the following. Say a sensory sequence is given that consists of temporally ordered sets of ground atoms describing the world at the given time. Then, to 'make sense' of this sequence, is to produce an explanation in the form of a theory. A theory provides an ontology (delineating the domain), initial conditions (providing a starting point for the dynamics of the system), rules (governing the dynamics of the system), and constraints (necessary for the system to satisfy the unity conditions that will be discussed later in this section). With these four elements, a 'trace' is created: an infinite sensory sequence. The idea is to find a theory whose trace 'covers' the given sensory sequence, i.e. the sensory sequence is a subset of the trace: in that case, the theory is said to be an 'interpretation' of the given sequence. A second requirement that Evans et al. impose on the theory they are looking for is that it satisfies the four 'unity conditions' they propose. These were inspired by Kant, and their purpose is to make sure

[1] When I first started working on this material, it had only been pre-published [31], and so I occasionally refer back to that version when I deem this better suited.

that different parts of the theory—objects, predicates, atoms, states—form a coherent whole: (i) spatial unity: objects are bound together in space via binary relations; (ii) conceptual unity: predicates are bound together into mutually exclusive groups via the constraints; (iii) static unity: atoms in the same time step jointly adhere to the set of constraints; (iv) temporal unity: states are bound together in a temporally ordered sequence via rules. If a theory satisfies these four unity conditions, it is called 'unified'.

In conclusion then, according to Evans et al., what it means to 'make sense' of a sensory sequence is to provide a unified interpretation of that sequence.

4.1.2 An interpretation

Now that we have seen the high level idea, let us take a look at the details. Everything begins with the input, viz. the *sensory sequence*. It consists of a temporally ordered set of sets, in which every set describes the state of the world at a certain time step.

Definition 1. A **sensory sequence** S is an ordered set $(S_1, S_2, \dots)$, where every **state** S_t is a set of grounded atoms providing a partial description of the world at discrete time step t. A grounded atom $p(a) \in S_t$ denotes that object a has property p at time step t. A grounded atom $r(a, b) \in S_t$ denotes that object a relates to object b via relation r at time step t.[2]

According to Evans et al., to 'make sense' of this sensory sequence, involves constructing a symbolic causal theory that is an interpretation of the sensory sequence and satisfies a set of unity conditions. Let us start with the details of the theory. A *theory* consists of four elements: a type signature (ontology), initial conditions (starting point of the dynamics), rules (the dynamics), and constraints. It is therefore written in a new language that Evans et al. designed for modeling dynamics, Datalog$^\ni$, which is a typed extension of Datalog[3] that includes causal rules and constraints.

Definition 2. A **theory** θ is a tuple (ϕ, I, R, C), where

- ϕ is a type signature
- I is a set of initial conditions
- R is a set of rules
- C is a set of constraints

Let us discuss each of these elements in turn.

The *type signature* provides the ontology that defines the universe in which we work: it lists the objects that are present, the properties and relations, the types that those objects can be of or that the properties and relations can take as arguments, and the variables necessary to work with the properties and relations.

Definition 3. The **type signature** ϕ is a tuple (T, O, P, V), where

- $T \subseteq \mathcal{T}$ is a finite set of types

[3] For a brief formal introduction to Datalog, see [61, p.378].

- $O \subseteq \mathcal{O}$ is a finite set of constants representing objects
- $P \subseteq \mathcal{P}$ is a finite set of predicates representing properties and relations
- $V \subseteq \mathcal{V}$ is a finite set of variables

Additionally, we define the functions

- $\kappa_O : O \to T$ for the type of an object
- $\kappa_P : P \to T^*$ for the type of a predicate's arguments
- $\kappa_V : V \to T$ for the type of a variable

Every type signature generates a set of *ground atoms* and a set of *unground atoms*:

Definition 4. Given a type signature ϕ, let

- $G_\phi = \{\, p(a_1, \ldots, a_n) \mid p \in P, \kappa_P(p) = (t_1, \ldots, t_n), a_i \in O, \kappa_O(a_i) = t_i \text{ for all } i = 1 \ldots n \,\}$ be the set of all **ground** atoms that are well-typed according to type signature ϕ; and
- $U_\phi = \{\, p(v_1, \ldots, v_n) \mid p \in P, \kappa_P(p) = (t_1, \ldots, t_n), v_i \in V, \kappa_V(v_i) = t_i \text{ for all } i = 1 \ldots n \,\}$ be the set of all **unground** atoms that are well-typed according to type signature ϕ.

Note that for an atom to be unground, all its arguments need to be unground. This also means that there are atoms that are neither ground nor unground, such as $p(a, X)$.

Not all type signatures are suitable for a given sensory sequence: the constants and predicates in the sensory sequence have to be present in the type signature as well. This may be formalized as follows.

Definition 5. Let $G_S = \bigcup_{t \leq 1} S_t$ be the set of all ground atoms in sensory sequence S. A type signature ϕ is **suitable** for a sensory sequence S, if all atoms in S are well-typed according to ϕ, i.e. if $G_S \subseteq G_\phi$.

The second element of a theory, the *initial conditions*, provides a starting point from which the system's dynamics develop: it consists of a set of ground atoms for the initial time step.

Definition 6. The **initial conditions** I of a theory consist of a set of ground atoms from G_ϕ that give a partial description of the world at the initial time step.

Stating the initial conditions explicitly is necessary because the system is allowed to posit *latent information*: it may invent latent objects and latent predicates in case it is unable to provide an explanation using only the surface information in the sensory sequence. Whereas the initial conditions for the surface information is already explicit, those for the latent information are not, hence the explicit definition.

The third element of a theory, the *rules*, provide the dynamics of the system and consists of two sets of clauses: one set of static rules and one set of causal rules. The *static rules* describe which additional pieces of information (by some called 'ramifications') may be deduced from already known pieces of information *within* a state. The *causal rules*, on the other hand,

describe the dynamics *between* states, linking them in a temporal sequence.

Definition 7. There are two types of rules in Datalog$^\exists$:

- A **static rule** is a definite clause (Horn clause with exactly one positive literal) of the form $\alpha_1 \wedge \ldots \wedge \alpha_n \rightarrow \alpha_0$, where $n \geq 0$ and $\alpha_i \in U_\phi$. This describes the dynamic that if conditions $\alpha_1 \wedge \ldots \wedge \alpha_n$ hold at a certain time step, that then α_0 also holds at the *same* time step.
- A **causal rule** is a clause of the form $\alpha_1 \wedge \ldots \wedge \alpha_n \ni \alpha_0$, where $n \leq 0$ and $\alpha_i \in U_\phi$. This describes the dynamic that if conditions $\alpha_1 \wedge \ldots \wedge \alpha_n$ hold at a certain time step, that then α_0 holds at the *next* time step.

The rules are implicitly universally quantified, i.e. a rule $p(X) \ni q(X)$, for example, describes the dynamic that if some object X has the property p at some time step, then that same object has the property q at the next time step.

The last element of a theory is a set of *constraints*, which are necessary to guarantee part of the unity conditions. There are three kinds of constraints: unary constraints, binary constraints, and uniqueness constraints. *Unary and binary constraints* demarcate predicate groups, i.e. sets of predicates for which any object (of the relevant argument type) can satisfy maximally one out of each set: these constraints guarantee conceptual unity (i.e. all predicates are bound into mutually exclusive groups). *Uniqueness constraints* guarantee that, given some type of object and a binary relation, for each object of that type, there is another unique object to which it is related via the given binary relation: these constraints guarantee spatial unity (i.e. all objects are at least indirectly related to each other).

Definition 8. There are three types of constraints.

- A **unary constraint** is an expression of the form $\forall X : t, p_1(X) \oplus \ldots \oplus p_n(X)$, where $n > 1$, meaning that every object of type t has exactly one of the properties $p_1(X), \ldots, p_n(X)$.
- A **binary constraint** is an expression of the form $\forall X : t_1, \forall Y : t_2, r_1(X,Y) \oplus \ldots \oplus r_n(X,Y)$, where $n > 1$, meaning that every object of type t_1 is related to any object of type t_2 by one of the relations $r_1(X,Y), \ldots, r_n(X,Y)$.
- A **uniqueness constraint** is an expression of the form $\forall X : t_1, \exists! Y : t_2, r(X,Y)$, meaning that every object of type t_1 is related to exactly one object of type t_2.

Now that we have seen the details of a theory, the next step is to look at the first of the two requirements it has to satisfy in order for it to actually 'make sense' of a sensory sequence: the theory must be an interpretation of the sensory sequence.

In order to build up to defining the concept of an interpretation, we need to start by defining the concept of a *trace*: an infinite sequence of sets of sensory data generated by a theory. The goal is to define a theory whose trace includes[4] the sensory sequence that the theory is trying to explain.

Definition 9. Every theory θ produces a **trace**: an infinite sequence $\tau(\theta) = (A_1, A_2, \ldots)$, where each A_t is the smallest set of ground atoms that satisfies the following conditions:

[4] What is meant by this exactly will be defined formally in def.10.

- $I \subseteq A_1$
- It is closed under the static rules:
  If there is a static rule $\beta_1 \wedge \ldots \wedge \beta_m \rightarrow \alpha$ in R and a ground substitution σ such that A_t satisfies $\beta_i\sigma$ for each β_i, then $\alpha\sigma \in A_t$.
- It is closed under the causal rules:
  If there is a causal rule $\beta_1 \wedge \ldots \wedge \beta_m \ni \alpha$ in R and a ground substitution σ such that A_{t-1} satisfies $\beta_i\sigma$ for each β_i, then $\alpha\sigma \in A_t$.
- *Frame axiom*: If α is in A_{t-1} and there is no atom in A_t that is incompossible* with α w.r.t. constraints C, then $\alpha \in A_t$.

  *Two ground atoms are **incompossible** if there is some constraint c in C and some substitution σ such that the ground constraint $c\sigma$ makes it impossible for both atoms to be true.

Including the frame axiom makes a theory more concise: a theory needs to specify only rules that describe how atoms change, instead of also needing to specify rules that describe that all atoms remain the same when possible in a given transition between time steps.

We are now ready to define what it means for a theory to be an interpretation of a sensory sequence.

Definition 10. Given a finite sequence $S = (S_1, \ldots, S_T)$ and a (not necessarily finite) sequence S', if $S = (S'_1, S'_2, \ldots)$ and $S_i \subseteq S'_i$ for all $1 \leq i \leq T$, then $S \sqsubseteq S'$, or in words: S is covered by S', or S' **covers** S.

A theory θ **explains** a sensory sequence S, or θ is an **interpretation** of S, if the trace of θ covers S, i.e. if $S \sqsubseteq \tau(\theta)$.

As Evans et al. describe with clarity: when we provide "a theory θ that explains a sensory sequence S, we make S intelligible by placing it within a bigger picture: while S is a scanty and incomplete description of a fragment of the time-series, $\tau(\theta)$ is a complete and determinate description of the whole time-series" [1, p.7][5].

Now that we have seen how to get to an interpretation that 'explains' a sensory sequence S, the next step is to unify that interpretation so that it 'makes sense' of S.

4.1.3 A unified interpretation

The unity conditions can be seen as meta-constraints on the model that are there to make sure that the different parts of the theory are one coherent whole. Unification takes place on four levels of abstraction: the "trace $\tau(\theta)$ of theory θ is a (i) sequence of (ii) sets of ground atoms composed of (iii) predicates and (iv) objects" [1, p.8].

Definition 11. A theory is **unified** when it guarantees the four unity conditions:

1. Spatial unity: objects are united in space by being connected via chains of relations
2. Conceptual unity: predicates are united by constraints

[5] Evans et al. prove formally that the trace is determinate in theorem 3 in the original paper [1, p.13].

3. Static unity: atoms are united in a state by jointly satisfying constraints and static rules

4. Temporal unity: states are united in a sequence by causal rules

Let us look at each of these conditions in more detail.

Spatial unity dictates that all objects must be united in space[6] via a chain of binary relations: in other words, any object is at least indirectly related to every other object.

Definition 12. A theory θ satisfies **spatial unity** if for each state A_t in $\tau(\theta) = (A_1, A_2, \dots)$, for each pair (x, y) of distinct objects, x and y are connected via a chain of binary atoms $\{r_1(x, z_1), r_2(z_1, z_2), \dots, r_n(z_{n1}, z_n), r_{n+1}(z_n, y)\} \subseteq A_t$.

In order to check for spatial unity, every state in the trace needs to be examined. Even though the trace is infinitely long, it repeats itself at some point[7] , making it sufficient to check just a finite part of it[8].

Conceptual unity dictates that all concepts (i.e. predicates) must be united via constraints: in other words, any concept occurs in some constraint.

Definition 13. A theory $\theta = (\phi, I, R, C)$ satisfies **conceptual unity** if

- for each unary predicate p in ϕ, there is some xor constraint in C of the form $\forall X : t, p(X) \oplus q(X) \oplus \dots$ containing p; and
- for each binary predicate r in ϕ, there is
  - some xor constraint in C of the form $\forall X : t_1, \forall Y : t_2, r(X, Y) \oplus s(X, Y) \oplus \dots$; or
  - some $\exists!$ constraint in C of the form $\forall X : t_1, \exists! Y : t_2, r(X, Y)$

  containing r.

Conceptual unity is necessary to guarantee exhaustiveness and exclusiveness between atoms: both important properties for the functioning of the model. Consider, for example, the xor constraint $on(x) \oplus off(X)$. *Exhaustiveness* ensures that an object cannot be neither *on* nor *off*, i.e. it makes states determinate. *Exclusiveness* ensures that an object cannot be both *on* and *off*, which grounds the notion of incompossibility, which is necessary to constrain the scope of the frame axiom (as is formalized in def.9).

Static unity dictates that all propositions that are true at the same time step must jointly satisfy the set of constraints and adhere to the static rules.

Definition 14. A theory $\theta = (\phi, I, R, C)$ satisfies **static unity** if every state $(A_1, A_2, \dots)$ in $\tau(\theta)$ satisfies all the constraints in C and is closed under the static rules in R.

[6] Note that 'in space' does not mean that the binary relation has to be interpreted as spatial: any relation will do.

[7] This is the informal version of theorem 1 in the original paper: the formal version and its proof can be found in [1, p.7].

[8] This is the informal version of theorem 2 in the original paper: the formal version and its proof can be found in [1, p.7].

Note that the way the trace has been defined (def.9) guarantees that every state is closed under the static rules, leaving only the question whether all constraints are satisfied.

Temporal unity dictates that all states must be united into a sequence by causal rules.

Definition 15. A sequence $(A_1, A_2, \dots)$ of states satisfies **temporal unity** with respect to a set $R_\ni$ of causal rules if, for each $\alpha_1 \wedge \dots \wedge \alpha_n \ni \alpha_0$ in $R_\ni$, for each ground substitution σ, for each time step t, if $\{\alpha_1\sigma, \dots, \alpha_n\sigma\} \subseteq A_t$, then $\alpha_0\sigma \in A_{t+1}$.

Note that the way the trace has been defined (def.9) guarantees temporal unity.

An overview of the unity conditions (which elements are bound together by which unifiers) can be found in table 4.1[9].

Kind of unity	**Unified elements**	**Unifier**
Spatial unity	Objects	Relations
Conceptual unity	Predicates	Constraints
Static unity	Atoms	Constraints & static rules
Temporal unity	States	Causal rules

Table 4.1: Overview of the four unity conditions

We are now ready to define what it means for a theory to be a unified interpretation of a sensory sequence.

Definition 16. A theory θ **makes sense** of a sensory sequence S, or θ is a **unified interpretation** of S, if

- θ explains S; and
- θ satisfies the four unity conditions

Now we know what it means to explain a sensory sequence, the question arises whether a sensory sequence can always be explained, i.e. whether it is always possible to find a unified interpretation given some sensory sequence. The following theorem[10] states that it is, providing some form of completeness for the Apperception model.

Theorem 1. For every apperception task (S, ϕ, C), there exists some interpretation $\theta = (\phi', I, R, C')$ that makes sense of S, where ϕ' extends ϕ and $C' \supseteq C$.

Proof.
The proof consists of a procedure for building a degenerate theory given some undefined sensory sequence: instead of finding and remembering patterns between atoms in the sequence, the theory merely remembers every individual atom.

We will start with defining ϕ' based on $\phi = (T, O, P, V)$ that is given. For each sensor x_i in S, where $i = 1 \dots n$, and for each state S_j, where $j = 1 \dots m$, we create a new unary predicate

[9] This is an adapted version of table 1 in Evans et al.'s pre-publication [31, p.25].
[10] Theorem 4 in the original paper (p.13).

p^i_j. The idea is that $p^i_j(X)$ is true whenever variable X is grounded by the ith object x_i in the jth time step. Let the new predicate take arguments of the type that x_i is of, i.e. if $\kappa_O(x_i) = t$, then let $\kappa_P(p^i_j) = (t)$. Additionally, for each type $t \in T$, create a new variable X_t, where $\kappa_V(X_t) = t$. Now, let $\phi' = (T, O, P', V')$, where

$$P' = P \cup \{p^i_j \mid i = 1 \dots n,\ j = 1 \dots m\}$$

and

$$V' = V \cup \{X_t \mid t \in T\}$$

Next, we define the rest of our theory $\theta = (\phi', I, R, C')$. First, let the initial conditions be the set of newly defined predicates that are true in the first time step:

$$I = \{p^i_1(x_i) \mid i = 1 \dots n\}$$

Second, let the rules contain causal clauses that connects the newly defined predicates temporally: each newly defined predicate for object i at time j is connected to the newly defined predicate for that object at time step $j + 1$:

$$p^i_j(X_t) \ni p^i_{j+1}(X_t)$$

where $i = 1 \dots n$ and $j = 1 \dots m - 1$ (where x_i is of type t). Additionally, let the rules contain static clauses that connect each newly defined predicate for object i at time j to the original predicate that is true for that object at that time: for each unary atom $q(x_i)$ in S:

$$p_i(X_t) \to q(X_t)$$

and similarly, for each binary atom $r(x_i, x_k)$ in S (where x_i is of type t and x_k of type t'):

$$p_i(X_t) \wedge p_k(Y_{t'}) \to r(X_t, Y_{t'})$$

Hence, every object gets a number of newly created predicates, one for each time step, which are only true at that time and when grounded by that object. Each of those predicates, when true, makes true the original atom(s) that involved that object at that time.

Lastly, we extend C to C' by adding constraints that bind all predicates into groups. We define set P_t of all unary predicates of type t:

$$P_t = \{p^i_j \mid \kappa_O(x_i) = t,\ j = 1 \dots m\}$$

Now, say $P_t = \{p'_1, \dots, p'_k\}$. For each type t, we add a unary constraint that binds together all newly defined predicates whose arguments are of that type (or in other words: newly defined predicates that have been created for objects of that type):

$$\forall X_t : t,\ p'_1(X_t) \oplus \dots \oplus p'_k(X_t)$$

The predicates in every thus-created group are indeed mutually exclusive when considered for a single object: every predicate is only true at a specific time step and when grounded by *a specific object*; since a constraint (when used) is grounded by only one object, no two

predicates in one group can be simultaneously true.

Checking the unity conditions
Spatial unity is guaranteed in theorem 1's proof by adding a 'world' object to which all other objects relate: Add object $w \in O$ of new type $t_w \in T$; for every type t, add add a relation $partt(t, t_w) \in P$ and a constraint $\exists! Y : t_w, partt(X, Y) \in C$; for every object x of type t, add an atom $partt(x, w) \in I$.

Conceptual unity is guaranteed in theorem 1's proof by having constraints for the latent predicates that are built-in by design[11].

Static unity is partly guaranteed by the definition of the trace, and is partly built-into the latent predicates by design[12].

Temporal unity is always guaranteed, as it is built-in by design of the trace: thus, it is guaranteed in theorem 1's proof as well. ■

This degenerate unified interpretation provides an upper bound for the complexity of the theory needed to explain a given sensory sequence.

4.1.4 Choosing a unified interpretation

Given some sensory sequence, there are infinitely many possible unified interpretations: how to choose between them? Evans et al. are interested in parsimonious solutions, and thus define a cost function that distinguishes between unified interpretations based on a compound score: they define the cost as the sum of the number of ground atoms in the initial conditions and the number of unground atoms in the rules, or formally:

Definition 17. Given a theory $\theta = (\phi, I, R, C)$, its **cost** is defined as

$$\text{cost}(\theta) = |I| + \sum \left\{ n + 1 \mid \alpha_1 \wedge \ldots \wedge \alpha_n \circ \alpha_0 \in R, \circ \in \{\rightarrow, \ni\} \right\}$$

Now, the notion that we have been building up to and combines the concepts so far discussed is the *apperception task*: given a sensory sequence, a suitable type signature, and 'appropriate' constraints (as defined below), the task is to find the lowest-cost theory that makes sense of S[13].

Definition 18. Given a triple (S, ϕ, C), where S is a sensory sequence, ϕ is a suitable type signature, and C is a set of constraints such that

(i) each predicate in S appears in some constraint in C; and

(ii) S can be extended to satisfy C,

the **apperception task** is to find the lowest-cost theory $\theta = (\phi', I, R, C')$ such that ϕ'

[11] An unanswered question I have, is what is done with the input predicates: there is no mention of constraints binding those together.

[12] This, again, does not seem to be guaranteed for the input predicates, as these are not bound by constraints (see footnote 11).

[13] Giving only a sensory sequence would leave the task "severely under-constrained": an example illustrating this can be found in the original paper [1, p.10].

extends ϕ, $C' \supseteq C$, and θ makes sense of S.

This is the task that the Apperception model is made for, and that the Apperception engine is (thus) designed to implement.

4.1.5 Example

To illustrate the concepts just defined, let us take a look at an example. We will (naturally) start with an apperception task, for which we will consider two[14] possible unified interpretations.

Input: apperception task (S, ϕ, C)
We will start with the sensory sequence $S = (S_1, \ldots, S_{10})$, where

$$\begin{aligned} S_1 &= \{\} \\ S_2 &= \{\mathit{off}(a), \mathit{on}(b)\} \\ S_3 &= \{\mathit{on}(a), \mathit{off}(b)\} \\ S_4 &= \{\mathit{on}(a), \mathit{on}(b)\} \\ S_5 &= \{\mathit{on}(b)\} \\ S_6 &= \{\mathit{on}(a), \mathit{off}(b)\} \\ S_7 &= \{\mathit{on}(a), \mathit{on}(b)\} \\ S_8 &= \{\mathit{off}(a), \mathit{on}(b)\} \\ S_9 &= \{\mathit{on}(a)\} \\ S_{10} &= \{\} \end{aligned} \tag{4.1}$$

Here, there are two objects, sensor a and sensor b, which can be either *on* or *off* at any given time step. Note that at some time steps we have complete information on both sensors (steps 2, 3, 4, 6, 7, 8), whereas in others we only have partial information on one of the two sensors (steps 5, 9) or on neither of the sensors (steps 1, 10).

Next, the type signature:

$$\begin{aligned} \phi = (T, O, P, V), \text{ where} \\ T &= \{\mathit{sensor}\} \\ O &= \{a, b\} \\ P &= \{\mathit{on}(\mathit{sensor}), \mathit{off}(\mathit{sensor})\} \\ V &= \{X : \mathit{sensor}\} \end{aligned} \tag{4.2}$$

And the constraints:

$$C = \{\forall X : \mathit{sensor}, \mathit{on}(X) \oplus \mathit{off}(X)\} \tag{4.3}$$

Output: θ_1 (latent predicates)[15]
The first possible unified interpretation $\theta_1 = (\phi', I, R, C')$ that we will consider involves positing three new predicates to explain the input. In this interpretation, sensors a and b each loop through three states, viz. latent predicates p_1, p_2, p_3, and they are *on* or *off* depending on which state they are in: in states p_1 and p_2 they are *on*, and in state p_3 they are *off*. Both sensors follow the same state transitions, however, since sensor a starts in state p_2 and sensor b starts in state p_1, they are out of sync. Below, the teal colored elements in

[14] For a third possible unified interpretation of this apperception task, see example 8 in [1, p.12].
[15] This is example 6 in [1, p.11].

the extended sets are those learned by the system.

$$\begin{aligned}
&\phi' = (T, O, P, V), \text{ where} \\
&\qquad T = \{sensor\} \\
&\qquad O = \{a\colon sensor, b\colon sensor\} \\
&\qquad P = \{on(sensor), off(sensor), r(sensor), p_1(sensor), p_2(sensor), p_3(sensor)\} \\
&\qquad V = \{X : sensor, Y : sensor\}
\end{aligned} \tag{4.4}$$

In addition to the three new latent unary predicates p_1, p_2, p_3, that are posited to describe the dynamics of the system, the latent binary predicate $r(sensor, sensor)$ is posited to guarantee spatial unity (i.e. all objects are (indirectly) related to each other). This new predicate needs to be constrained in order to guarantee conceptual unity (i.e. all concepts (predicates) must be unified via constraints), and thus the new variable $Y : sensor$ is invented.

$$I = \left\{ \begin{array}{l} p_2(a) \\ p_1(b) \\ r(a,b) \\ r(b,a) \end{array} \right\} \quad R = \left\{ \begin{array}{l} p_1(X) \ni p_2(X) \\ p_2(X) \ni p_3(X) \\ p_3(X) \ni p_1(X) \\ p_1(X) \rightarrow on(X) \\ p_2(X) \rightarrow on(X) \\ p_3(X) \rightarrow off(X) \end{array} \right\} \quad C' = \left\{ \begin{array}{l} \forall X : sensor, on(X) \oplus off(X) \\ \forall X : sensor, p_1(X) \oplus p_2(X) \oplus p_3(X) \\ \forall X : sensor, \exists! Y : sensor, r(X,Y) \end{array} \right\} \tag{4.5}$$

The initial conditions contain the two starting states of the sensors, in terms of the latent predicates posited, as well as the latent relations between the sensors, posited to guarantee spatial unity. The rules contain three causal rules to describe the state transitions, as well as three static rules that link a sensor's state with the sensor's attributes *on* and *off*. The constraints state that every sensor (i) is either *on* or *off*, (ii) is in exactly one state p_1, p_2, or p_3, and (iii) has exactly one sensor that relates to it via relation r.

Having found this theory θ_1, we can compute its trace $\tau(\theta_1)$, and thus predict future values (e.g. for S_{10}), fill in missing values (for S_5 and S_9), and retrodict past values (for S_1).

This interpretation has a cost of $(4 + 12 =)16$.

Output: θ_2 (latent object)[16]
In the second unified interpretation, $\theta_2 = (\phi', I, R, C')$, that we will consider, the states of sensors a and b are interpreted as depending on their context, i.e. on their neighbors, instead of on their internal state. In this interpretation, a new object c is posited that acts as the unobserved neighbor of a and b: the three sensors are positioned in a circle, making b the right neighbor of a, c the right neighbor of b and a the right neighbor of c. The sensors turn *on* and *off* according to the rule: if sensor X's right neighbor is *on* (respectively *off*) at time step t, then X is *off* (respectively *on*) at time step $t + 1$.

[16] This is example 7 in [1, p.12].

$$\begin{aligned}\phi' = (T, O, P, V), \text{ where}\\ T &= \{sensor\}\\ O &= \{a\colon sensor, b\colon sensor, c\colon sensor\}\\ P &= \{on(sensor), off(sensor), r(sensor)\}\\ V &= \{X : sensor, Y : sensor\}\end{aligned} \tag{4.6}$$

$$I = \left\{ \begin{array}{l} on(a)\\ on(b)\\ off(c)\\ r(a,b)\\ r(b,c)\\ r(c,a) \end{array} \right\} \quad R = \left\{ \begin{array}{l} r(X,Y) \wedge off(X) \ni on(Y)\\ r(X,Y) \wedge on(X) \ni off(Y) \end{array} \right\} \quad C' = \left\{ \begin{array}{l} \forall X : sensor, on(X) \oplus off(X)\\ \forall X : sensor, \exists! Y : sensor, r(X,Y) \end{array} \right\} \tag{4.7}$$

This interpretation has a cost of $(6 + 6 =)12$.

4.2 Implementation: the Apperception engine

In this section, I will discuss the computer implementation of the Apperception model, viz. the Apperception engine: the steps of the algorithm (4.2.1), as well as the experiments that Evans et al. designed to test the Apperception engine and their results (4.2.2).

The engine takes the form of a Datalog$^\ni$ interpreter in Answer Set Programming (ASP), and generates a unified interpretation for a given type signature: this can be seen as a kind of unsupervised program synthesis (as discussed in 2.2.2).

4.2.1 Algorithm

Given an apperception task (S, ϕ, C), the Apperception engine starts by enumering templates of increasing complexity, where a template is "a structure for circumscribing a large but finite set of theories" [1, p.15]; it consists of a type signature and some constant bounds for the complexity of the rules in the theory. Formally:

Definition 19. A **template** χ is a tuple $(\phi, N_\rightarrow, N_\ni, N_B)$, where $N_\rightarrow$ is the max number of static rules allowed in R, $N_\ni$ is the max number of causal rules allowed in R, N_B is the max number of body atoms allowed in rules in R.

For each template, it then finds theories that are unified interpretations of the sensory sequence given, and from those it keeps an ongoing log of which theory is the one with the lowest cost so far. When the engine runs out of processing time, it outputs the last-saved lowest-cost theory.

The two non-trivial parts of this algorithm are enumerating templates of increasing complexity, and finding the lowest-cost theory.

Enumerating templates

The algorithm follows a procedure[17] that is guaranteed to visit every possible template eventually. Which sets O', P', V' are possible depends on T, so the algorithm follows a

[17] To see this procedure applied to our example in 4.1.5, see [1, p.16].

Algorithm 1: The Apperception engine algorithm in high level pseudocode

input: (S, ϕ, C): an apperception task
output: θ^*: a unified interpretation of S
initialize a variable tuple to save the lowest-cost theory θ^* and its cost s^*: (s^*, θ^*)
Enumerate templates of increasing complexity:
foreach *template* χ *extending* ϕ *of increasing complexity* **do**
  Find the lowest-cost theory θ:
  if θ *is non-empty* **then**
    if $s < s^*$*, i.e. the cost of* θ *is less than the cost of the lowest-cost theory* θ^* *saved so far* **then**
      replace (s^*, θ^*) by (s, θ)
    end
  end
  if *processing time exceeded* **then**
    return lowest-cost theory θ^* saved so far
  end
end

two-tiered process: First, the algorithm enumerates possible sets T; second, given some T[18], it enumerate tuples $(O, P, V, N_{\rightarrow}, N_{\ni}, N_B)$[19]. However, since there are infinitely many such tuples, a constant bound n is introduced that dictates how many tuples are enumerated. The order in which to consider (T, n) pairs is determined by a standard diagonalization method, which gradually increases both elements. Given a certain (T, n) pair, different methods may be used to enumerate n templates: Evans et al. have chosen to use a Haskell function that minimizes biases towards certain tuples.

Finding the lowest-cost theory

The Datalog$^{\ni}$ interpreter generates all possible combinations of all possible sets of I, R, and C' within the given template, and computes the traces of the accompanying theories up to a finite time limit. It then checks for each theory whether it is indeed a unified interpretation and if so, what its cost is. If it is the lowest encountered so far, it returns that theory for the template under consideration.

4.2.2 Experiments

In order to test the Apperception engine, Evans et al. selected a number of tasks in varying domains: specifically tasks that are simple for humans, but difficult for modern ML approaches. Additionally, "[t]he tasks were chosen to highlight the difference between mere perception (the classification tasks that machine learning systems already excel at) and apperception (assimilating information into a coherent integrated theory, something traditional machine learning systems are not designed to do)" [1, p.21]. The following tasks were selected: elementary cellular automata, drum rhythms and nursery tunes, sequence induction tasks, binding tasks, and occlusion tasks. The results from these tasks show that the Apperception performs well in all five task domains.

Elementary cellular automata (ECAs) are one-dimensional, circular arrays of cells, where the state of each cell depends on its previous state and the previous state of its neighbors. The sensory sequence for elementary cellular automata consists of the states of the cells in

[18] A small remark: The system is not allowed to invent types, as Richard Evans confirms: "The Apperception Engine does not currently support introducing new types, but there is no deep reason why not. I believe it would be (moderately) straightforward to extend it to do this." [62] However, from this two-tiered procedure in the Apperception engine algorithm it does seem that the set of types T that has been inputted can be changed...

[19] This enumeration is kick-started by a base template χ_0, which consists of the type signature that is given, plus three values for the parameters that seem to be hand-coded by the designers (see [1, p.16] to read the original text).

the array.

The sensory sequence for *drum rhythms and nursery tunes* is generated by sensors that can each detect a specific note (*C* to *HighC*) or drum beat (bass drum, snare drum, hi-hat), and that can differentiate between that note's or beat's loudness (0 (unplayed) to 3 (played), which will decrease back to 0).

The *sequence induction* task is the most basic predication task: given a sequence of symbols (in this case: atoms), the goal is to predict the next symbol (atom).

Binding tasks (as discussed in section 2.3.4) involve binding different modalities into a coherent whole. Evans et al. created a multimodal task by taking the cells in the ECA task (of a certain type) and adding some extra cells (of a different type) whose behavior depended on a subset of the original cells. For the system to make sense of the sequence given, it needed to find the pattern connecting the extra cells to the original cells.

Occlusion tasks (as discussed in section 2.3.4) involve recognizing objects that are sometimes (partially) hidden from view. Modern ML approaches encounter difficulties with these tasks as they lack the knowledge that objects persist over time; the Apperception system, "by contrast, was designed to posit latent objects that persist over time" [1, p.24]. Evans et al. created a task where, in a two-dimensional grid, objects move horizontally (left or right, in differing speeds), each in its own row. In each column, a sensor is 'looking up', giving it a cross section of the rows: it can see only the one object at the time; the one that is closest to it. Other objects located in that column at that time are occluded from view. For the system to make sense of the sequence given, it needed to impute the positions-in-time of the objects that were missing in the given sensory data.

In order to assess the results of the engine in these tasks, it is necessary to place the engine in perspective so that its results may be compared to existing approaches. The family of approaches in the research landscape that the engine is closest to is Inductive Logic Programming (ILP). Hence, let us first see how the engine and ILP approaches relate, before evaluating the results of the engine relatively to a state-of-the-art ILP system.

Comparison with ILP approaches

ILP approaches and the Apperception engine both aim to construct a logic program that explains a set of given examples.

There are two broad categories of ILP approaches: bottom-up approaches, which deduce specific clauses from the input and generalize from there, and top-down approaches, which use a language definition to generate clauses and check those against the input ('generate-and-test') [24, p.7]. The Apperception engine is closest to the latter.

An ILP task consists of three sets: one containing positive examples (in the form of ground atoms), one containing negative examples (idem), and one containing background knowledge (facts and rules in the form of clauses). An apperception task contains three sets as well. One set contains the sensory sequence that is to be explained: this is analogous to the ILP sets of positive and negative examples, except that a sensory sequence consists of temporally ordered sets of only positive examples. The two other sets contain the type signature and the constraints: these are analogous to the ILP set of background knowledge, except that the type signature and constraints do not contain any facts or rules, but define a language and constraints on this language respectively. Any facts and rules in the output of the Apperception engine are learned.

An ILP hypothesis consists of a set of rules in the form of a logic program, that, when applied to the given background knowledge, "entails as many positive and as few negative examples as possible" [23, p.2], or in more strict cases, "entails all the positive examples but

does not entail any of the negative example" [24, p.1] [20]. A theory as constructed by the Apperception engine consists of four sets: two extending the given sets containing the type signature and the constraints, one containing the initial conditions (in the form of ground atoms), and one that, analogously to an ILP hypothesis, contains rules, such that applying those rules to the initial conditions creates the input sensory sequence.

Two recent developments that bring ILP approaches closer to the Apperception engine have been the following. First, the successful implementation of the ability to invent new predicate symbols when the provided background knowledge is not sufficient to explain the input examples [23, p.3][21]. And second, representing the hypothesis in alternative ways to Prolog programs (i.e. definite logic programs), e.g. Datalog programs (which allows a program to be encoded as a satisfiability problem, as done in [24]) and ASP programs (allowing for no, one, or multiple models (answer sets), instead of just one (as in Prolog): the non-monotonicity of these programs makes these extended ILP approaches suitable for modeling commonsense reasoning [23, p.4]).

Hence, ILP approaches and the Apperception engine are very similar.

Results

Now we are ready to discuss the results. For each kind of task, Evans et al. ran the system 30 times (except for the ECA task: this was ran 256 times[22]). Predictive accuracy per task was measured in a binary fashion: the system's performance counted as 'accurate' if it predicted *every* hidden sensor value correctly, and as 'inaccurate' otherwise. Given this "rather exacting" measure (as Evans et al. describe it [1, p.21]), the Apperception system performs well in all five domains[23] :

ECA task	97.3%
Rhythms and tunes task	73.3%
Sequence induction task	76.7%
Binding task	85.0%
Occlusion task	91.0%

In order to put these results in perspective, Evans et al. gave the same tasks to ILASP: a state-of-the-art ILP system, which learns answer set programs instead of (merely) definite logic programs [1, p.28]. ILASP was unable to solve the ECA task, the rhythms and tunes task, and the sequence induction task, as they required too much memory. As for the binding task and the occlusion task, there the size of the solutions increased exponentially with larger problem sizes, which is not the case for the Apperception engine. This highlights the main criticism of ILP approaches, viz. "the lack of efficacy of its methods and algorithms" [21, p.129], and the point where the Apperception engine provides an advantage: it can work with "significantly larger" problem sizes [1, footnote 46, p.37].

A second advantage that the Apperception engine has compared to ILP approaches in the context of XAI is the extent to which it 'explains' the output. ILP is more general, learning formulas which do not (necesssarily) have any coherence, and are thus unlikely to have much meaning to a human reader. Furthermore, since logic is so widely applicable, the fact that some phenomenon can be captured in logical formulas is not necessarily very informative. In contrast, the Apperception engine's formulas do have coherence, owing to the unity conditions, making them more informative and meaningful for humans. Additionally, the

[20] These are just two examples: more semantics are possible (see [63, p.635]).

[21] As Cropper et al. note: "Whilst predicate invention has attracted interest since the beginnings of ILP, most previous attempts have been unsuccessful resulting in no support for predicate invention in popular ILP systems" [23, p.3]

[22] The reason is that there are 256 possible update rules for the cells in the array.

[23] For more details of the results per task, see [1, p.22].

pre-knowledge given to the Apperception engine (viz. a type signature and constraints, as well as the causal and temporal aspects of the language bias) is much richer in information than the background knowledge that may be inserted in an ILP system (containing atoms or Horn clauses). Hence, the Apperception engine adheres better to the definition of what it means 'to explain' (as given in 3.1), than ILP approaches do.

In conclusion, the Apperception engine surpasses the approaches closest to it, viz. ILP approaches, in both level of performance and quality of explanation: precisely the two subgoals Evans et al. set for their system (as discussed in the introduction to this chapter).

5 Evaluation of the Apperception system

In this chapter, I will evaluate both parts of the Apperception system in the context of XAI. I will start by critically assessing the key aspects of the Apperception model (5.1). Then, I will discuss the characteristics and (dis)advantages of the Apperception engine from the perspective of chapter 3 (5.2). And I will conclude by answering my first research question: 'What are the strengths and weaknesses of the Apperception system when evaluated in the context of explainable AI?' (5.3).

5.1 Evaluating the Apperception model

In this section, I will discuss several important aspects of the Apperception model from a more philosophical, or meta, level: the meaning of theorem 1 (5.1.1), the role of the unity conditions (5.1.2), the role of the cost definition (5.1.3), and the concept of 'making sense' (5.1.4).

5.1.1 Theorem 1

Let us begin by considering theorem 1 (p.25), which proves that, given any finite sensory sequence, a unified interpretation can always be found. This is not a surprising result: given finite data, it is always possible to describe some dependency relation between that data. Instead, theorem 1 provides an upper bound on the complexity of the theory necessary to make sense of the sensory sequence. However, since this is a rather high upper bound, theorem 1 is also not a very strong result. It treats the inputted data as a random sequence with no underlying pattern, and creates an artificial (or as Evans et al. call it: "degenerate") solution: every time-stamped atom in the given sequence is related to a newly created latent predicate specific for that atom, and then those predicates are temporally linked. This creates a structure resembling a clothing line for each object: of latent predicates that are temporally ordered by design, which then point towards the original given atoms that are true in that particular state. Hence, this unified interpretation that can always be found is large and, in Evans et al.'s own words, "unilluminating" [1, p.14].

What theorem 1 does not do, is prove anything about whether the Apperception model actually is a "good" model, i.e. a model that creates high quality solutions: meaning, in this context, to what extent Evans et al.'s goal of designing an AI system with (i) performance that can compete with modern ML, yet (ii) is explainable, is reached. In this context, quality of explanation can be quantified as the conciseness of the explanation. This is not generally the case, since in some contexts a smaller explanation is produced by means of 'meta-concepts': the products of compounding multiple simpler concepts into one more complex concept. The more meta-levels up, the more complex a concept is, and the less understandable it becomes (in the sense of serving as an explanation for input of the simplest kind of building blocks). When (higher-level) meta-concepts are used, the individual building blocks of the explanation are more difficult to understand, (potentially) offsetting the positive

effect of the conciseness of the whole explanation on understandability. In the Apperception system, however, the possibility of concept-compounding is very limited. The building blocks of the explanation are either given in the input, or they are invented by the system (for latent objects and latent predicates). In the former case, building blocks are of the simplest kind, as there is only one level in the input and thus no place for compounding to occur. In the latter case, building blocks are of the simplest kind or (maximally) one meta-level up, when compounding the concepts from the input. Either way, their complexity is always low and thus their understandability high, making a smaller explanation indeed a better explanation in the context of the Apperception model. In practice, this means that (a) the positing of latent objects and predicates should be minimized (the underlying principle being that the interpretation of new information should, as much as possible, be done in terms of concepts already known about the world (Occam's razor)); and (b) the use of those known concepts should be minimized (e.g. not using more atoms in the rules than strictly necessary).

In order to guarantee that the Apperception model produces high-performing, concise solutions, Evans et al. introduce the extra requirement of the unity conditions (for subgoal (i)), and the cost function to select between theories (for subgoal (ii)). This makes that there are several layers of selection between the space of all possible theories (given some sensory sequence) and the lowest-cost unified interpretation that is the final output: from the set of all theories, the set of all interpretations is selected (layer 1); from there, the set of all unified interpretations (layer 2); and from that set, the lowest-cost unified interpretation is chosen (layer 3).

The key question is thus whether the design choices Evans et al. made regarding these two additional layers indeed lead to a high-performing system that provides a sufficient explanation (as described in XAI terms) or that actually 'makes sense' (in their own terms). Let us consider each in turn.

5.1.2 Unity conditions

For the unity conditions, there are two levels at which decisions need to be made: which entities to unite, and what relations to unite them with. Regarding the first level, Evans et al. demand the objects, predicates, atoms, and states of a theory to be united (see table 4.1). Of course, other choices are possible, including not uniting some of these entities, and uniting entities of different kinds. Regarding the second level, Evans at al. require for example that there is at least an indirect relation between any two atoms. Here too, there are many alternative options; e.g. requiring all atoms to be related in the form of a chain $(R(a,b), R(b,c), R(c,d), \dots)$ or in a stronger form such as 'if $R(a,a)$, then $R(b,b)$'. (I will not go deeper into the considerations for choosing between the many possibilities[1].)

Evans et al.'s reason for choosing their specific unity conditions is that they are inspired by Kant's discussion of the "synthetic unity of apperception" in the *Critique of Pure Reason* [1, p.1]. They do not go deeper into their reasoning for why Kant, but they do provide an argument in favor of (these particular) unity conditions: the results of ablation tests that show that the model's performance (as measured in predication accuracy) "deteriorates noticably" [1, p.3][2] when any one[3] of the unity conditions is no longer guaranteed. Since performance is measured in terms of predictive accuracy, these tests only show that the unity conditions help to reach the first subgoal (performance on the level of modern ML systems). For the second subgoal (explainability), the unity conditions do not help, as the

[1] Unity is, similarly to explainability, a very broad concept: Evans et al. make rather minimal, and arguably obvious, choices in formalizing it, which is fine when doing pioneering work.

[2] For the full results, see [1, p.27].

[3] To be precise: They performed ablation tests for spatial unity and conceptual unity. Taking out temporal unity cannot be tested, as it is guaranteed by the definition of the trace. Taking out static unity is not necessary to test, since the first part (viz. that every state in the trace is closed under the static rules) is guaranteed by the definition of the trace and can thus not be taken out, and taking out the second part (viz. that every state in the trace satisfies the constraints) is indirectly checked by the ablation test for conceptual unity (where the constraints are taken out entirely).

theory in theorem 1's proof illustrates clearly: it is bound by the unity conditions, yet, it is questionable whether this 'unilluminating' theory still 'makes sense' of the given sequence. I am using the verb to 'make sense' here not in the formally defined way of definition 16—the theory obviously makes sense that way—but I am using it in the way that the verb is used intuitively, or in Evans et al.'s words: "[w]e have an intuitive understanding of what is involved in making sense of the sensory stream" [1, p.1]. This 'intuitive understanding' of making sense is what Evans et al. are trying to formalize, and whether the theory in theorem 1's proof indeed makes sense in this second way is certainly debatable. As Evans et al. themselves describe: "the interpretation provided by Theorem [1] is degenerate and unilluminating: it treats each object entirely separately (failing to capture any regularities between objects' behaviour) and treats every time-step entirely separately (failing to capture any laws that hold over multiple time-steps)" [1, p.14].

The deeper issue that I would like to raise here is of (what may be seen as a form of) completeness of the model. Starting from an intuitive notion that picks out a certain referent, a formal notion is defined with the goal of making the process of picking out that referent more exact. However, for the research process to come full circle, it is important to investigate the other direction as well, viz. whether the referent of the formal notion indeed corresponds to the referent of the intuitive notion. The theory of the proof of theorem 1 is an example of a case where it is doubtful whether what is picked out by the formal notion still matches what would be picked out by the intuitive notion. An interesting related point is whether the theory in theorem 1's proof can be said to even have any meaning at all. In related work that Evans et al. discuss [1, p.31], it has been argued that meaningful concepts may be created from abstract terms by grounding their meaning in the roles they play in the dynamics of the model [64, p.458, 467, 478], which is similar to Ludwig Wittgenstein's philosophical view of "meaning as use" [65]. This argument would lead to dismissing that the concepts in the theory in theorem 1's proof get (much) meaning assigned, since the meaning that the predicates in the atoms of the sensory sequence get from their context in this theory is none: they do not relate to each other; they only relate to meaningless object-in-time-identifiers (the latent predicates), which only relate to each other arbitrarily (viz. in a artificially designed (temporal) order). It is worthwhile to consider whether, if the theory in theorem 1's proof is not creating any meaning for the elements that are to be explained, it can (still) be said to make sense of those elements.

Since the unity conditions do not aid in reaching Evans et al.'s second subgoal of making the system explainable, they introduce the cost definition, which acts as a selection tool to choose the 'best' theory: meaning, in the context of XAI, the most explaining theory.

5.1.3 Cost definition

What would constitute a good cost definition given the goal? Just as with the notion of 'making sense', the effort here is to find a translation from an intuitive notion (here: a 'good' explanation) to a formal notion (here: a quantification of the quality of an explanation).

Similarly to the unity conditions, there are two levels at which decisions need to be made for the cost definition: which elements to take into account, and how to take them into account. Regarding the first level, the choice of which elements to take into account, the options that may be considered are all elements of a theory: $\theta = (\phi', I, R, C')$, where $\phi' = (T', O', P', V')$. Not every element is equally suitable, however:

- T': types. This is not an option, since the system does not support extending types, so always $T' = T$.
- O': objects. This is a suitable option.
- P': predicates. This is a suitable option.

- V': variables. This is not an option, since it is an indirect measure: how V is extended to V' is directly dependent on how the sets O and P are extended to O' and P'.
- I: initial conditions. This is a suitable option.
- R: rules. This is a suitable option.
- C': constraints. This is not an option, since it is an indirect measure: how C is extended to C' is directly dependent on how the set P is extended to the set P'.

In conclusion, elements that are taken into account for the first level, which I call ***relevant measures***, may only be based on the sets O', P', I, R. Regarding the second level, the choice of how to take the chosen relevant measures into account, there is the option to count elements: based on e.g. its 'natural size', i.e. its cardinality (e.g. the number of rules in the set R); or some more contrived criterion (e.g. the number unground atoms in the rules in the set R). Alternatively, there are criteria that do not involve counting, such as Kolmogorov or Levin complexity (as Evans et al. mention [1, footnote 19, p.10]).

For the first level, Evans et al. have chosen a compound score of I and R. It may be argued that these are indeed the two most appropriate elements to take into account from the options described above, viz. O', P', I, R. The system learns the initial conditions and rules to explain the dynamics of the sensory sequence, and it can posit latent objects and predicates if necessary to supplement the given type signature (ontology). Hence, I and R may be seen as the more fundamental building blocks of the explanation. For the second level, Evans et al. have decided to count: the simplest choice is to count the elements of the set, i.e. the atoms in I and the rules in R. However, since these are different kinds of elements, it is dubious to compound their counts. In I, there are no other options for counting, but in R there are many alternative options (number of only static or only causal rules, number of body atoms, number of head atoms, to name a few). The question is: 'What is the most informative measure of the amount of information in the rules?' Counting entire rules is not the answer, as this does not differentiate between short (simple) rules and longer (complex) rules, which *does* matter for the goal of the defining a cost, viz. to differentiate between unified interpretations based on their explainability. Evans et al. have chosen to count the number of unground atoms in the rules, which is a more accurate measure of the amount of information in the rules than counting the number of rules would be.

I would like to argue here that using only one cost definition in the Apperception model is suboptimal, and that it would be better to use multiple cost definitions at the same time; letting a human make the final call about which solution fits the given task best. When making use of a single cost definition, as done by Evans et al., potentially valuable solutions may be missed. To illustrate, consider our example in 4.1.5 and the unified interpretations θ_1 and θ_2 that make sense of it. Here, there are two distinct kinds of theories[4]: the former mainly invents latent predicates to explain the sensory sequence, whereas the latter mainly invents latent objects. As discussed (4.1.5), the human narratives for interpreting these theories are very different, and both are interesting and potentially relevant, as is evident from the fact that they are used by Evans et al. to show the diversity of solutions that can be generated by the Apperception model. Since these two unified interpretations do not have the same cost, only one of them can have the lowest cost, making that (at least) one of them would be missed. And even if they would have had the same, indeed-lowest cost, then still only one of them would have been outputted, as the Apperception engine tie-breaks[5]. There are cases in which it is useful to be able to see these different options, which are close to each other in cost, and choose which narrative fits best as the solution for the task. Hence, this is an argument in favor of having multiple cost definitions: each returning a solution of a certain kind (optimizing one relevant measure), between which a human can then make the final call. Do note that I am not arguing that using a single cost definition

[4] As mentioned in footnote 14, a third kind of theory is discussed in the original paper as well.

[5] As can be seen in algorithm 4.2.1, the lowest-cost theory that is saved is only overridden if a theory comes along with a lower cost: it does not save a theory of equally low cost.

is an inferior choice *an sich*. In many other contexts, the goal is to optimize a quantifiable measure, such as efficiency of an algorithm in terms of speed or computational resources necessary. Here, however, the goal is to optimize an intuitive notion (viz. explainability of a solution), which is formalized as a quantifiable goal (viz. the cost of the solution), and the correspondence between these goals is not perfect (as discussed in 5.1.2). So it is therefore helpful to generate multiple potential solutions and have a human select between them.

Finally, Evans et al. have also performed ablation tests for their cost minimization, which show that the cost definition they have chosen does not (significantly) improve the model's performance, i.e. does not help in reaching the first subgoal. However, they note that "there are independent reasons to want to minimize the size of the interpretation", as "[s]horter interpretations are more human-readable, and transfer better to new situations (since they tend to be more general, as they have fewer atoms in the bodies of the rules)" [1, p.28]; i.e. the cost filter helps in reaching the second subgoal. Minimizing the cost is one way of creating shorter, easier-to-understand unified interpretations; another, more meta, way could be to extend the language of the model. The most common purpose of extending a language, viz. to increase its expressive power, is not relevant in the case of the Apperception model, as theorem 1 already tells us that any sensory sequence may be described in the language that Evans et al. have chosen. Instead, extending the language can be useful if it makes the representation of the information in a unified interpretation more understandable: for example by making it more concise. For the second research question of this thesis, I have researched in this direction, resulting in chapter 6.

5.1.4 Making sense

We have seen that Evans et al.'s definition of what it formally means to 'make sense' (def.16) only includes the first and second layers, viz. that the theory has to be an interpretation, and that it has to be unified. However, as the previous account of the unified interpretation in the proof of theorem 1 shows, it is debatable whether every theory that passes these two filters can be said to 'make sense' *intuitively*; i.e. whether the formal notion of 'making sense' indeed matches the intuitive notion of 'making sense' that it is meant to formalize. What is missing when applying only the first two filters, is the guarantee that the unified interpretation produced is small enough to be general and understandable. For this reason, Evans et al. introduce the third filter of the cost function, however, this is not integrated into the formal definition of 'making sense', but only used as an extra layer to select between the theories that qualify as 'making sense' already, i.e. between unified interpretations. I would like to argue that the cost function *should* be included in the definition of 'making sense'. Evans et al. state that, in their paper, they attempt to answer "a central question in unsupervised learning: what does it mean to "make sense" of a sensory sequence?" [1, p.1]. Their answer is that "making sense involves constructing a symbolic causal theory that both explains the sensory sequence and also satisfies a set of unity conditions" [1, p.1]. Hence, their formalization of making sense only involves the first layer ('explains') and the second layer ('satisfies a set of unity conditions'), which is indeed reflected in their formal definition of 'making sense' (def.16). However, as discussed (5.1.1), Evans et al.'s goal in terms of the context of XAI is to build a high-performing, explainable AI. The first subgoal is indeed achieved by their formal definition of 'making sense', but reaching the second subgoal requires the third layer of guaranteeing that the theory is low(est)-cost. Hence, I think it is an unfortunate choice of words to define the combination of the first two layers as 'making sense', and would like it better if that terms is reserved for the final product of all three layers.

5.2 Evaluating the Apperception engine

In this section, I will assess the Apperception engine from the perspective of chapter 3. I will start with a clarification, explaining my focus for this section (5.2.1), after which I will continue by discussing which of the characteristics of classical AI and modern ML apply to the engine, and which (dis)advantages are associated with those (5.2.2).

5.2.1 Clarification

As discussed, the Apperception system has two components: the Apperception model (a formalization), and the Apperception engine (its implementation). The model merely formalizes what it means to 'make sense' of a sensory sequence, but is not a computer (in the sense of taking an input and producing an output). The engine, on the other hand, *is* the computer: the one that implements the model. When making an evaluation of a system in the context of XAI, it concerns the computing part; the part that goes through a process, which either produces an output that constitutes the explanation, or which itself constitutes the explanation. In the case of the Apperception system, therefore, the engine is the component to assess: this is the hybrid part, consisting of a classical AI approach that also has the ability to learn, similarly to modern ML approaches: it therefore falls into the XAI category of 'designing a white box' (as discussed in section 3.3.2).

However, another clarification needs to be made. Given the engine, there are two computations[6] that the engine performs that may be considered for evaluation: the generation of a theory, and the generation of the trace. For the first computation, the input is an apperception task, and the goal is to make sense of the inputted sequence given the type signature and constraints by outputting a unified interpretation. For the second computation, this output, a set of logical formulas that together represent a body of knowledge (i.e. a knowledge base), is the input, and the goal is to be able to predict, impute, or retrodict missing parts of the sensory sequence that is to be explained by outputting a trace that covers the sequence. The inventiveness of Evans et al.'s approach is this second step: the first output of the system (the theory), is a knowledge base, which is then computed in a fully classical way (not involving learning) to produce a second output (the trace that covers the sensory sequence). The combination of the first and second outputs together provide a strong explanation: the sensory sequence is explained by the trace, and the trace is explained by the theory (as it provides the ontology and explicit rules that give rise to the trace).

In this section, I will evaluate the process of the first computation, as there the hybridity of the engine is put to use: there, the engine uses its ability to learn. I will start by briefly discussing the Apperception engine's characteristics, after which I will discuss its (dis)advantages, and I will end with briefly discussing possible improvements to it.

5.2.2 In the context of XAI

Since the engine is a classical AI approach, but then with the ability to learn, it has most of the characteristics of classical AI, except that it is partially data-driven / bottom-up (instead of fully rule-driven / top-down) and that it is partially self-learning (instead of fully dependent on pre-knowledge). This is summarized in table 5.1, which lists the typical characteristics of classical AI approaches (first column) and modern ML approaches (second column) (as discussed in 2.2.1 and 2.2.2), as well as the hybridization of those characteristics present in the Apperception engine (third column).

This fusing of characteristics has its effect on the (dis)advantages of the system as well: I will discuss how they are influenced next. It might be of use to reference table 5.2 while reading, as it summarizes the text: it lists the typical (dis)advantages of classical AI approaches (first two columns) (as discussed in 2.3.1 and 2.3.2) and modern ML approaches (middle two

[6] In the sense of taking an input and producing an output.

classical AI characteristics	modern ML characteristics	Apperception engine characteristics
Symbolic representation, local storage ↳ Strong language bias	Distributed representation, global storage	Symbolic representation, local storage ↳ Strong language bias
Compositionality	Entangled knowledge	Compositionality
Rule-driven, top-down	Data-driven, bottom-up	Data-driven, bottom-up
Explicitly defined logics	Implicitly defined logics	Explicitly defined logics
Hand-engineered pre-knowledge	Self-learning	Hand-engineered pre-knowledge Self-learning
Serial computation	Parallel computation	Serial computation
Transparent (white box)	Opaque (black box)	Transparent (white box)

Table 5.1: Overview of characteristics of classical AI, modern ML, and the Apperception engine.

columns) (as discussed in 2.3.3 and 2.3.4), as well as the combination of those (dis)advantages that the Apperception engine has (final two columns).

The engine is **human-readable**, since the system takes concrete steps, which are represented symbolically (which does come with the potential downside of **not always being able to (easily) represent knowledge**), and it is thus **interpretable**, since the output can be traced back via a chain of these steps to the original input.

Due to its symbolic nature, the engine is **modular**: both with respect to its inner workings (it has a number of separate processes it loops through per run, which may be built into a different system), as well as with respect to its representation of knowledge (information is stored locally and can thus be transferred as isolated kernels of knowledge to be used at another time or place). This second kind of modularity makes it possible for the engine to be enriched with **pre-knowledge**, in the sense of pre-loading the system with knowledge that it can use in producing its output. Generally, classical AI systems need to be given all knowledge they use in generating their outputs beforehand, and that knowledge needs to come from somewhere, so it is hand-engineered by the designers. As discussed in 2.3.2, this kind of pre-knowledge can be difficult and/or time-consuming to produce. This is different for the Apperception engine: there, only a minimal amount of (hand-engineered) knowledge is necessary to get the system started, after which the system is able to add new knowledge to that base by itself. It works as follows. What is particular to the Apperception engine, is that the input already contains part of the output. The general scheme for AI systems is to get input in the form of data, and to produce an output in the form of an analysis of that data: i.e. the input and the output are of different kinds of information. This is slightly different for the engine, however, as its input consists of data to be analyzed (viz. the sensory sequence), plus part of what will be the output (viz. a type signature and constraints). Hence, everytime the engine produces an output, part of that information can be put back into the system as part of the input, i.e. as part of the knowledge base that the system takes as its starting point: it thereby forgoes the need for additional hand-engineered knowledge before it can improve its performance. (Do note that the possibility to enrich the engine with this (special) kind of pre-knowledge (that is given as input, instead of being built-in by design, or given via some designer backdoor, as is generally the case) is limited, as the input only contains part of the output, viz. the type signature and the constraints, but not the initial conditions or the rules. The engine is thus unable to use previously produced initial conditions or rules for a new run or task.) The reason this is beneficial is that the engine produces information that is actually new, instead of only deductions from information that has been given (as is the case in classical AI), since it has the **ability to learn**: a hallmark characteristic of modern ML approaches, making the system truly hybrid. The transferring of knowledge learned by a system for one task to another session involving a different task, possibly in a new domain, is a form of **generalization** called **transfer & lifelong learning**[7] [23, p.4835], which is considered "a hallmark of human intelligence" [11, p.17], "a corner stone of human learning" [28, p.20], and "essential for human-like AI" [23, p.4833]. It is made possible by the combination of modular representation of knowledge (a typical characteristic of classical AI) and learning ability (a typical characteristic of modern ML) [23, p.4833], and is thus a benefit only reserved for hybrid approaches. This kind of learning makes the engine more self-contained, in the sense of being able to progress its knowledge, and thus enrich the system('s performance), without external influence. It stands in contrast to non-hybrid systems, as classical AI needs to be fed new information from designers in order to improve its performance, and the knowledge learned by modern ML systems cannot be extracted to be used again, due to its entangled nature.

The combination of a strong language bias (in the form of the causal language of Datalog$^\ni$ determining the form of the rules) and the requirement of the unity conditions, gives the engine a **strong inductive bias**, making it **data-efficient**: "able to make sense of the shortest and scantiest of sensory sequences" [1, p.2]. Because the system works with small

[7] Currently, the system can only reuse part of its output, and only through human intervention of taking the right part of the output and using it as input for a new run, however, Evans et al. mention that useful future work would be to extend and automate this process [1, p.37].

classical AI advantages	classical AI disadvantages	modern ML advantages	modern ML disadvantages	Apperception engine advantages	Apperception engine disadvantages
Human-readable ↳ Interpretable	Not always possible to represent symbolically	High representational power	Not human-readable ↳ Uninterpretable	Human-readable ↳ Interpretable	Not always possible to represent symbolically
Modular ↳ Pre-knowledge possible ↳↳ Generalizable	Hand-engineered pre-knowledge necessary	Learning ability	Not modular ↳ Pre-knowledge and transfer & lifelong learning not possible ↳↳ Poor generalization	Modular ↳ Pre-knowledge partly possible + Learning ability ↳ Transfer & lifelong learning (= form of generalization)	
Strong inductive bias ↳ Data-efficient ↳ Computationally simple	Low computational power ↳ Cannot handle large amounts of data	High computational power ↳ Can handle large amounts of data	No strong inductive bias ↳ Data-inefficient ↳ Computationally complex	Strong inductive bias ↳ Data-efficient ↳ Computationally simple	Low computational power ↳ Cannot handle large amounts of data
High precision	High quality data necessary	Can handle low quality data	Less precision	High precision	High quality data necessary

Table 5.2: Overview of (dis)advantages of classical AI, modern ML, and the Apperception engine.

amounts of data, it is **computationally simple** (as is typical for classical AI approaches), not needing many computational resources to run. However, because the engine uses serial computation, the speed at which it computes is (relatively) low (when compared to modern ML approaches), making that the engine also has the typical **low computational power** of classical AI, making it "restricted to small-to-medium-size problems" (i.e. **cannot deal with large amounts of data**) [1, p.37][8]. Finally, the typical characteristic of classical AI of being **highly precise** is also present in the engine: it produces solutions that explain the input exactly, which does mean that it **needs high quality data** (i.e. is unable to deal with noisy, erroneous, or ambiguous data).

5.3 Conclusion

To conclude this chapter, I will first recap the narrative so far (the quest of XAI, and the stories of the model and the engine), before answering the first research question 'What are the strengths and weaknesses of the Apperception system when evaluated in the context of explainable AI?'

As discussed in 2.3.4, the downsides of modern ML are uninterpretability, poor generalization, and data-inefficiency. Particularly the downside of uninterpretability is problematic, as it hinders guaranteeing certain desirable ethical properties (unbiasedness, privacy, transparency) (as discussed in 3.2). The research field of XAI aims to solve this problem: the goal is to design systems with the high performance of modern ML, but to make them explainable as well, which allows desirable ethical properties to be guaranteed. This is the context in which Evans et al. have designed their Apperception system: the goals was to design a system which is (i) high-performing (in terms of predictive accuracy), and (ii) explainable (in their system: provides concise explanations).

Evans et al. designed the formalization of their system (the model) in such a way that an implementation for it (in their case: the engine) is most likely able to reach those goals: viz. by building in two extra filters for selecting between solutions: the unity conditions and the cost definition. Starting from interpretations (theories whose trace covers the given sensory sequence), the first filter results in the set of unified interpretations (those interpretations that satisfy the unity conditions), and the second filter reduces that set to the set of lowest-cost unified interpretations. The unity conditions help for subgoal (i) (as shown by ablation tests), but not for subgoal (ii) (as illustrated by the unified interpretation in the proof of theorem 1: it is questionable whether it 'makes sense' intuitively, despite satisfying the unity conditions). The cost definition helps for subgoal (ii) (as it selects for more concise, and thus more understandable, unified interpretations), but not for subgoal (i) (as shown by ablation tests). Together, these two filters help guarantee that a high quality solution, in terms of predictive accuracy and conciseness of explanation, is selected.

In the formal definition that Evans et al. give for what it means to 'make sense' of sensory data, only the first filter of the unity conditions is included. Since both filters are necessary in order to guarantee the overall goal, I have argued that both filters should be included in the definition. It seems that Evans et. al.'s philosophical considerations (to include the unity conditions suggested by Kant, but not the cost definition suggested for practical reasons) have taken priority over their objective of designing a system that formalizes what it means to make sense intuitively.

Another point I raised regarding this definition, is the issue of completeness: the question whether the formal notion and the intuitive notion of 'making sense' stand in perfect correspondence, i.e. pick out the same referent. Given what we have seen so far, I think it is a *good* correspondence, but not *perfect*, as is to be expected: this is pioneering work in

[8] Evans et al. mention this as one of the limitations of their system, and believe one way to improve on this in future work is to use what they call "curriculum learning" [1, p.37], which is essentially the extended and automated version of transfer & lifelong learning mentioned in footnote 7.

the field. A way to compensate for this imperfect correspondence is to use multiple cost definitions, thereby keeping multiple possible referents in the game that can be picked out by the formal notion, between which may then be selected by a human that referent (or those referents) which come closest to the intuitive notion in the given task environment.

To return to the larger context of XAI: the implementation of the model, i.e. the engine, indeed does not suffer from the three downsides that modern ML approaches generally encounter. The engine is interpretable, as it is able to answer the question '*how* was this output produced for this input?' by virtue of its symbolic representation, which allows the following of every step in its process (in contrast to the parallel, entangled processes of modern ML systems). Additionally, the engine is able to generalize, in the form of transfer and lifelong learning, because of the combination of its modularity and ability to learn. Relatedly, this allows for better performance on the binding and occlusion problems (as discussed in 4.2.2) than modern ML approaches do. And finally, on the data-efficiency level, the engine does not need the large amounts of data that modern ML approaches need, making it computationally simple (though less so than the average, non-learning classical AI). Nonetheless, there are points of improvement for the engine, two of which are the following. First, the generalization process is only possible through human intervention: it would enhance the engine to make that process automatic. Second, the engine does not scale well, i.e. cannot deal with large problem sizes, which currently prevents it from performing on the level of modern ML. (Though it should be noted that competing techniques such as ILP approaches do worse in this area, and so the Apperception engine's performance, though not enough *absolutely*, is still praiseworthy *relatively*.) A solution would be to choose a different implementation altogether[9].

So, to answer the first research question 'What are the strengths and weaknesses of the Apperception system when evaluated in the context of explainable AI?': The strengths of the system are: its relatively high performance; how close it comes to formalizing the intuitive concept of 'making sense'; and that it indeed solves the problems that modern ML systems encounter, viz. uninterpretability, poor generalization, and data-inefficiency. Its weaknesses are: that it cannot (yet) compete in performance with modern ML approaches; and that the definitions of 'make sense' and of the cost may be improved. Improving performance is largely a matter of improving scalability, which is the most pressing issue to address and (mainly) concerns the implementation. Delineating the formal concept of 'making sense' differently is a simple matter of adjusting terminology (though it is of fundamental importance to the overall quest). And thirdly, considering alternative cost definitions takes time and effort: I have made a start in this direction in the next chapter.

[9]Richard Evans is working on this (as discussed in footnote 6). Additionally, he has used a different implementation for the Apperception model before, but then for an extended version of it, designed for settings with unparsed data [66].

6 Extensions to the Apperception model

In this chapter, my goal is to provide some exploratory research that may hopefully serve as a starting point for more in-depth future research into extensions to the language of the Apperception model. I will start by discussing my investigation: some context and the extensions I considered (6.1), before I turn to the three extensions which I investigated in more detail, viz. adding constants (6.2), negation (6.3), and disjunction (6.4) to the rules. I will conclude by answering my second research question: 'What are potentially useful extensions to the language of the Apperception model in the context of explainable AI?' (6.5).

6.1 Introduction

In this section, I will start by discussing the trade-off between expressive power and reasoning complexity that every designer of AI systems faces (6.1.1), as well as discussing on a high level the different extensions that I have considered (6.1.2).

6.1.1 Trade-off expressive power and reasoning complexity

Two important qualities of a computational system are its expressive power and its reasoning complexity. *Expressive power* (EP) concerns what information the system is able to model, or in other words, the diversity of information that a system's representation is able to distinguish between. Generally speaking, higher EP is preferable, as this allows the system to model a broader range of information. *Reasoning complexity* (RC) concerns the computability of the system, or put differently, whether and how easily a solution can be produced. Hence, this determines whether and how easily the system is implementable. Generally speaking, lower RC is preferable, as this improves workability of the system. EP and RC are positively correlated and thus give rise to a trade-off. One area of research in which this balance is studied intensively is in formal language theory, where languages are catergorized according to their balance between EP and RC. In this area, the well-known Chomsky hierarchy is used as a discrete categorization scale, consisting of so-called *complexity classes*. These classes are based on natural categories arising from the realm of finite automata. Since the Apperception system is not based in finite automata theory, however, it does not fit neatly into one of the Chomsky complexity classes: it is not Turing-complete (*recursively enumerable* class), nor does it fall into the category under Turing-complete (*context-sensitive* class)[1].

Instead, Evans et al. have chosen a balance between EP and RC based on their intended application of the system. The goal was to create a system that is easy to implement on the one hand, i.e. that has low enough RC; and that is able to model the input provided

[1] For readers interested in the complexity of Datalog and other logic programming languages, I recommend [61].

on the other hand, i.e. that has high enough EP. The way to manipulate the balance here, is in choosing what inputs to provide the machine with, as this influences how much EP is necessary, and thus how low RC can be kept. Evans et al. have chosen to keep the input simple, by i.a. confining it to a finite domain. To illustrate how fundamental this choice is for the system they built, consider the following line of reasoning that shows that the system would no longer be deterministic, nor would it satisfy the unity conditions, if the domain is infinite. Suppose the domain of the Apperception system is infinite, i.e. either P or O (or both) is infinite. Then the set of ground atoms is not finite. Then the set of possible states is not finite. Then the infinite trace need not contain a repeated state[2]. Then it need not be decidable whether some ground atom α, given some theory θ, appears in the trace of θ [3]. Then we cannot check every time step of a trace. Then we cannot check for spatial unity and static unity (as this requires checking every time step [1, p.9]). Hence, finiteness is essential to the Apperception system. (Do note that this is not an argument against the Apperception system: the system has simply not been designed to be able to deal with infinite domains, but can easily be adapted to be able to do so.) By confining the input of the Apperception system to a finite domain, it is always possible to describe a dependency between inputted data (as discussed in 5.1.1 and confirmed by theorem 1). This is why the cost definition is so important: it is always possible to find a unified interpretation; the more interesting question is what unified interpretation is the better one.

I think Evans et al. have made an appriopiate choice in the trade-off between EP and RC given the intended use of their system. I am therefore not interested in changing this balance point. Instead, it is more interesting to investigate extensions that keep the system around the same level of EP and RC, but that do improve the system on another level. In the context of XAI, I am interested in improving the quality of the explanation, i.e. the understandability of the explanation, which can be reduced to conciseness of the explanation (as discussed in 5.1.1). Conciseness of an explanation may be improved by adding syntactical elements called *syntactical sugar*: elements that are not essential to the system (i.e. do not improve EP[4]), but do make it easier for humans to understand the representation. This adding of inessential elements to a system may seem to go against the notion of Occam's razor. However, in this context, Occam's razor is not uniquely defined: an explanation can take the form of the system itself, or merely of the system's output, either case leading to different arguments as to which elements should be minimized. In Evans et al.'s case, it is the output of the system that serves as the explanation and so it is *its* size that should be minimized in order to increase understandability; the understandability of the system as a whole, on the other hand, does not matter, and so the size of the system need not be minimized.

In conclusion, the goal is to, given some potential extensions, find a relative proof, i.e. a proof showing that the considered extension allows for more concise explanations of (certain) situations than the original system is already able to model, instead of finding an abolute proof, i.e. a proof showing that the considered extension allows for more EP and so for the modeling of more situations.

6.1.2 Potential extensions

The goal of the Apperception model is to 'make sense' of the provided input. An important choice that needs to be made before investigating potential extensions to the model's language, is whether to extend that input as well, or whether to keep it in its original form. Either option leads to a different inquiry: when extending the input in addition to the language of the model, one shows that the system may be extended to be applied in a 'richer' world; when keeping the original input and only extending the language of the model, one

[2] As theorem 1 in the original paper (mentioned in footnote 7) does not hold.

[3] As theorem 2 in the original paper (mentioned in footnote 8) does not hold.

[4] Conciseness may also be improved by adding syntactical elements that *are* essential and thus improve EP, however, as discussed, the goal for this thesis is to keep the balancing point between EP and RC approximately the same.

shows that the Apperception model may be improved in the given world. I have chosen to focus on the second inquiry: I am interested in improve upon the model as presented by Evans et al.[5].

The next step is to decide on which parts of the model to extend. The input that the model is provided with has a starting point (the first time step) from which it then develops over time: a theory makes sense of that starting point with initial conditions, whereas the dynamics of how that starting point develops over time is explained by the rules. These two parts (the initial conditions and the rules) of the theory are therefore primary. The other parts of the theory, i.e. the type signature and the constraints, are on different levels: the type signature is on a more fundamental level, providing an ontology for the world in which the input takes place, and the constraints are on a meta-level, regulating how certain parts of the ontology (viz. the predicates) relate to each other. In investigating extensions to the model, the natural place to start is with the primary parts, i.e. the initial conditions and rules. Of course, changes to these parts will influence the other two parts of the theory, i.e. with the type signature and constraints, so those should be investigated as a subsequent step.

Since the rules are the place where most of the logic in the model is found (the initial conditions consist of atoms only), I have looked into extending the forms they may take. Evans et al. restrict them to definite clauses (Horn clauses with exactly one positive literal) containing only unground atoms (atoms containing only variables, no constants). I have considered a number of different extensions: adding constants, quantification, negation, or disjunction to the rules; making the rules probabilistic; and allowing for exogenous states. Note that these are extensions to the language of the *model*: they are possible in theory, however, they might not be implementable in the same way, i.e. by the *engine* (a Datalog$^\ni$ interpreter in ASP, as discussed in 4.2). This need not be a problem though, as there is more than one way of implementing the model[6].

Constants

In the original system, only unground atoms are allowed in the rules: a simple extension, therefore, is to add the possibility of having ground atoms there as well. Evans et al. have made the deliberate decision to not allow for constants in the rules. This is to prevent special-case rules that apply to particular objects[7], and thus forces the theory to be general [1, p.6]. Having the system produce only general theories "was inspired by reading Kant, in which rules are always universally quantified"; "the more general philosophical claim is that constants are an aberration invented by Frege" [62]. I think it is nonetheless valuable to explore the possibility of extending the model's language, even if doing so goes against the philosophical considerations that Evans et al. focus on. Adding constants is the extension that I have spent most time thinking through, and I will discuss my exploration of the topic in 6.2.

Quantification

In the original system, the rules have the form of Horn clauses, in which a limited form of universal quantification is already present implicitly. This extension entails adding quantification explicitly, both universal and existential, and thus allows for e.g. bringing quantifiers into formulas, applying logical operators to quantifiers, and quantifying existentially. As Horn clauses lack the EP of first order logic [68, p.351], it is a natural step when extending them to consider adding quantification, since that would make the system reducable to first

[5] At least as much as possible: in 6.2.2, I will make the case to, in addition to investigating the extensions, also investigate (and potentially divert from) the original cost definition in the process, which is already a deviation from the original model.

[6] Richard Evans himself, for example, mentioned he is "working on a total reimplementation using rather different (and hopefully more scalable) underlying techniques" [67].

[7] Richard Evans mentions that it *is* possible to learn object-specific rules, but "only by adding lots of conditions to the body of a rule making it more and more specific" [62], which leads to a higher cost and thus the system is biased against theories with these kinds of rules.

order logic. However, as I have chosen to keep the balance between EP and RC approximately the same when extending the Apperception system, this extension does not fit to that goal. Therefore, I did not pursue this option further.

Negation

The original system does not work with negation, nor does the input include any form of it. When extending the system to include negation, there are different kinds of negation that may be considered: I have looked into classical negation and (stratified) negation as failure (NAF), which I will discuss in section 6.3.

Modeling non-determinism

In their pre-publication, Evans et al. mention that a limitation of their system was that all causal rules are fully deterministic [31, p.65]. Three potential ways of implementing non-determinism are: (i) introducing exogenous actions; (ii) adding disjunction to the rules; and (iii) making the rules probabilistic.

Exogenous actions

The first option to implement non-determinism, proposed by Evans et al. in their pre-publication [31, p.65-66] is to add so-called *exogenous actions* to the system: "actions that affect the state but do not themselves need to be explained". In the published paper [1, p.37], they briefly mention how they have executed this proposal in the form of an "extended theory": a theory in which every time step has initial conditions, instead of only the first time step. An extended theory $\theta = (\phi, \{I_1, ..., I_T\}, R, C)$ generates a trace $\tau(\theta) = (A_1, A_2, ...)$ as normal (see definition 9), except now $I_t \subseteq A_t$ (instead of $I \subseteq A_1$). This models non-determinism as it allows the abduction of atoms whose truth-values depend on the initial conditions (which are state-specific), instead of on the rules (which are trace-general).

I am critical of this method. Exogenous actions, here, seem to serve a similar purpose to inventing latent predicates and objects[8]: if there is not enough information in the input itself to explain its dynamics, latent information is invented. However, arguably, positing a relatively small[9] extension to the type signature (as in the case of latent predicates and objects) is less of a leap than positing new information for every single time step (as in the case of Evans et al.'s extended theory).

Disjunction

In the original system, the rules have the form of Horn clauses and so they have conjunctive bodies and atomic heads. An extension could be to consider allowing disjunction in the rules. Allowing disjunction in the body of a rule would not increase the system's EP, but *would* allow combining multiple rules with the same head into a single rule, thereby reducing the size of the output of the system and thus improving understandability. Allowing disjunction in the head of a rule, on the other hand, *does* increase the system's EP. I will expand on my exploration of adding disjunction to the rules in section 6.4.

Probabilistic rules

The third and final option of modeling non-determinism discussed here is making the rules probabilistic. The idea is that every rule specifies a small number of outcomes, each with a certain probability (summing to 1 in total). Pasula et al. [69, p.316][10] have done this,

[8] Additionally, a parallel with oracles in computability comes to mind (for those readers familiar with the notion), in the sense that adding exogenous actions seems similar to adding an oracle to every time step: it explains the information in that time step in an almost *ex machina* fashion.

[9] Small, since there is an inductive bias against inventing a larger number of latent objects or predicates, as the cost definition is based on parsimoniousness.

[10] This paper presents a system for learning a state transition model from data of the form (s, a, s'), where s is the current state, a the action performed, and s' the next state. The model learns the probability distribution $p(s \mid s, a)$, which is represented in the output as "dynamical rules": first-order clauses determining the next state given a current state and an action. These come close to the Apperception model's causal

creating rules of the form

$$\alpha_1 \wedge ... \wedge \alpha_n \quad \circ \quad \begin{cases} p_1 & \alpha_{01} \\ ... & ... \\ p_m & \alpha_{0m} \end{cases}$$

where α_i is some unground atom, p_j is a probability, and $\circ$ is the arrow for either a static or a causal rule. The problem with this approach, however, is that it becomes too similar to standard reinforcement learning (discussed in 2.2.2), making it quite far away from the goal of creating explainable AI. Therefore, I did not pursue this option further.

6.2 Constants

In this section, I will discuss my thoughts on investigating constants to the language of the Apperception model. I will start by considering the new shape of the rules and what meaning may be attributed to that (6.2.1), then I will explore how one may go about an investigation into adding constants (though this discussion is applicable to other extensions as well) (6.2.2), and I will end by discussing in what ways adding constants influences other parts of the system (6.2.3).

6.2.1 Form and meaning of rules with constants

When adding constants to the **bodies**, rules are of the form $\alpha_1 \wedge \ldots \wedge \alpha_{n-1} \wedge p_n(a) \quad \circ \quad \alpha_0$, where α_i is some unground atom, $p_n(a)$ is a ground atom, and $\circ$ is the arrow for either a static or a causal rule, and $p(a)$ is a ground atom. Similarly to a rule without constants, a rule with constants is a template, applicable to any objects of the right types that satisfy its predicates, however, now there is also a part of the body which is not a template, but only applicable to a specified object: ground atom $p_n(a)$. In the case of a **static** rule, the ground atom $p_n(a)$ can be seen as a kind of 'prerequisite' for being able to deduce the information in the head in that same time step. In the case of a **causal** rule, the ground atom $p_n(a)$ can be seen as a 'determiner' for a decision point, i.e. as information about a specific object that determines whether the information in the head will be true in the next time step. When thinking about these rules in this way, it would make sense to have groups of rules that have the same unground template in their bodies, but combined with different prerequisites or determiners, which hence lead to different outcomes (i.e. have different heads):

$$\alpha_1 \wedge \ldots \wedge \alpha_{n-1} \wedge \quad \begin{cases} p_{n1}(a) & \circ & \alpha_{01} \\ p_{n2}(a) & \circ & \alpha_{02} \\ \vdots & \vdots & \vdots \end{cases}$$

When adding constants to the **heads**, rules are of the form $\alpha_1 \wedge \ldots \wedge \alpha_n \quad \circ \quad p(a)$, where α_i is some unground atom, $\circ$ is the arrow for either a static or a causal rule, and $p(a)$ is a ground atom. A rule of this form represents a situation in which a piece of information about a specific object (the ground atom in the head) may be deduced from information about some unspecified objects of specified types (the template in the rule body), either in the same time step (**static** rule), or in the previous time step (**causal** rule).

Note that for constants to be potentially useful for making solutions more concise, there need to be multiple objects of a's type, otherwise the type characterization can be used to single out a specific object (by virtue of being the only object of that specific type)[11].

rules, but have probabilistic outcomes in each head.

[11] A thought that might arise is that constants in rules are beneficial only when the model is given a 'wrong' type signature, in the sense of a poorly chosen type division. This is not true, as objects of the same types have to satisfy the same groups of predicates as specified in the constraints, and so splitting one type

6.2.2 Exploratory study

Before formally investigating, it is useful to sharpen intuitions by creating an example dataset that would most likely have a conciser explanation in the extended model than in the original model.

Consider the following example, which is an adapted version of our example in 4.1.5: the teal colored elements in the sets are added.

The sensory sequence $S = (S_1, \ldots, S_{10})$, where

$$\begin{aligned}
S_1 &= \{\mathit{fly}(f_1)\} \\
S_2 &= \{\mathit{off}(a), \mathit{on}(b), \mathit{not}(f_1), \mathit{fly}(f_2)\} \\
S_3 &= \{\mathit{on}(a), \mathit{off}(b), \mathit{not}(f_1), \mathit{fly}(f_2)\} \\
S_4 &= \{\mathit{on}(a), \mathit{on}(b), \mathit{fly}(f_1), \mathit{not}(f_2)\} \\
S_5 &= \{\mathit{on}(b), \mathit{fly}(f_2)\} \\
S_6 &= \{\mathit{on}(a), \mathit{off}(b), \mathit{not}(f_1), \mathit{fly}(f_2)\} \\
S_7 &= \{\mathit{on}(a), \mathit{on}(b), \mathit{fly}(f_1), \mathit{not}(f_2)\} \\
S_8 &= \{\mathit{off}(a), \mathit{on}(b), \mathit{not}(f_1), \mathit{fly}(f_2)\} \\
S_9 &= \{\mathit{on}(a), \mathit{fly}(f_2)\} \\
S_{10} &= \{\}
\end{aligned} \tag{6.1}$$

The type signature:

$$\begin{aligned}
&\phi = (T, O, P, V), \text{ where} \\
&\qquad T = \{\mathit{sensor}, \mathit{flag}\} \\
&\qquad O = \{a, b, f_1, f_2\} \\
&\qquad P = \{\mathit{on}(\mathit{sensor}), \mathit{off}(\mathit{sensor}), \mathit{fly}(\mathit{flag}), \mathit{not}(\mathit{flag})\} \\
&\qquad V = \{X : \mathit{sensor}, Y : \mathit{flag}\}
\end{aligned} \tag{6.2}$$

And the constraints:

$$C = \left\{ \begin{array}{l} \forall X : \mathit{sensor}, \mathit{on}(X) \oplus \mathit{off}(X) \\ \forall Y : \mathit{flag}, \mathit{fly}(X) \oplus \mathit{not}(X) \end{array} \right\} \tag{6.3}$$

Here, two objects have been added to the model of the new type *flag*, viz. f_1 and f_2, and at any given moment[12] each is either flying (*fly*) or not flying (*not*). The dynamics of the flags depend[13] on the dynamics of the sensors a and b: flag f_1 flies when both sensors a and b are on, i.e. it can be seen as a logical *and*-gate; flag f_2 flies when either sensor a exclusive–or b is on, i.e. it can be seen as a logical *xor*-gate.

Logically, for the subset of rules describing the dynamics of the flags something similar to the following would be expected (notice the constants in the heads):

$$R_{\text{flags}} = \left\{ \begin{array}{l} \mathit{on}(X_1) \wedge \mathit{on}(X_2) \rightarrow \mathit{fly}(f_1) \\ \mathit{on}(X_1) \wedge \mathit{on}(X_2) \rightarrow \mathit{not}(f_2) \\ \mathit{on}(X_1) \wedge \mathit{off}(X_2) \rightarrow \mathit{fly}(f_2) \\ \mathit{on}(X_1) \wedge \mathit{off}(X_2) \rightarrow \mathit{not}(f_1) \end{array} \right\} \tag{6.4}$$

into two would increase the number of predicates, and therefore constraints, by a factor two, thus drastically increasing the size of the explanation.

[12] I have added the data for the flags to the sensory sequence in a similar fashion to Evans et al.'s original example, viz. with some data missing: of course, more or less data could have been added.

[13] This is one possible human narrative to explain the patterns in S. Of course, other narratives are possible.

Note that latent predicates cannot be used to fulfill the function of the constants in these last two rules: the reason is that, in that case, the heads would contain variables instead of constants, which is problematic in two ways. First, having (only) $fly(Y)$ in the head would lead to underspecification, since the variable Y is not specified in the body. Second, the rules would (therefore) be the same for both flags, whereas their dynamics are different. Hence, there seems merit in being able to model these dynamics with constants in the rules.

In order to confirm these intuitions, the follow-up is to implement both models and compare the conciseness of the explanations they produce. If the extended model's explanation is indeed conciser, then the next step is to look for a general formal proof showing that the explanation found by the extended model is always equally or more concise when compared to the original model, i.e. that the explanation produced by the extended model is always of equal or lower cost than the explanation produced by the original model.

Finding such a relative proof, however, is more troublesome when the cost definition is a compound score, as is the case in the original cost definition, where the number of atoms in I and in R are compounded. These numbers are independent from each other (able to increase or decrease without affecting the other), making it difficult to prove anything about their sum: it might be possible to show some relation between the numbers of atoms in R in the original and extended systems, and similarly for the numbers of atoms in I, however, the final proof concerns the ratio between those relations:

$$\begin{aligned} R_{\text{extended model}} + I_{\text{extended model}} &\leq R_{\text{original model}} + I_{\text{original model}} \\ R_{\text{extended model}} - R_{\text{original model}} &\leq I_{\text{original model}} - I_{\text{extended model}} \\ \frac{R_{\text{extended model}} - R_{\text{original model}}}{I_{\text{original model}} - I_{\text{extended model}}} &\leq 1 \end{aligned}$$

When the cost definition only involves a single relevant measure, it is not necessary to deal with that extra metalayer of how the relations of multiple relevant measures relate to each other. This is a second argument for changing the original cost definition: in 5.1.3, I made the case for multiple cost definitions instead of only one.

Since the cost definition plays such an important role, it is only natural to investigate this definition as well, and to ask: 'Is this the best way to define conciseness?' In this investigation, the cost definition is the determining factor in the comparison of explanations between the original model and the extended model: a 'good' solution is, in the context of XAI, an 'understandable' solution, which is, in the context of the Apperception model, a 'concise' solution (as discussed in 5.1.1), which is formalized by the cost definition (defining what elements of the solution count towards conciseness). In the model, the choice of cost definition is crucial, as it is highly influential on the model's completeness, i.e. the question of whether the referent(s) of the formation notion matches the referent(s) of the intuitive notion of 'making sense' (as discussed in 5.1.2). Hence, the cost definition is decisive, both in this investigation, and for the model. It is therefore an important choice whether the cost definition is considered a given or not. In the former case, the focus is on investigating the extension, doing research 'within the arena', i.e. working with the original cost definition and seeing whether adding constants gives conciser solutions according to this definition. In the latter case, the cost definition itself is also subject of investigation, doing research both within the arena *and* 'into the arena itself', i.e. working with different cost definitions in investigating adding constants, and seeing how they relate to each other with respect to the referents they pick out. In the light of the two arguments in favor of changing the original cost definition given above, I think the latter option would be most favorable in an investigation in the context of XAI.

6.2.3 Influences of constants on other parts of the system

As mentioned (6.1.2), extending the rules (and inventing the accompanying initial conditions) influences the other parts of the system as well.

First of all, it needs to be checked whether or not the **unity conditions** can still be guaranteed. I think they can be. Spatial unity (all objects are indirectly related, see def.12) does not involve, nor is influenced by, rules. Neither does conceptual unity (all predicates are bound together into mutually exclusive groups, see def.13). Static unity consists of two requirements: every state must (i) satisfy all the constraints, and (ii) be closed under the static rules (see def.14). In the original model, the second requirement (which involves rules) need not be checked, as it is guaranteed by the definition of the trace (def. 9), and this remains the case for the model extended with constants. Fourth and final, temporal unity (all states are bound into a temporal sequence by the causal rules, see def.15) involves the causal rules, however, this requirement too is (and remains) satisfied automatically by the definition of the trace.

Secondly, when staying 'within the arena' and adhering to the original **cost definition**, making some adjustments to it seems appropriate. Originally, the cost is the sum of the total number of ground atoms in the initial conditions plus the total number of unground atoms in the rules. When adding constants, the rules will also contain ground atoms. Evans et al.'s motivation for defining a cost is to select for "parsimonious interpretations" [1, p.10], in the sense of 'containing the least amount of information necessary'. However, what this means functionally may be interpreted in different ways for the question under consideration. One interpretation is to count ground atoms as more costly than unground atoms[14]., as they provide more specific information. Another interpretation is to simply count the amount of pieces of information (ground and unground atoms alike), instead of differentiating between different kinds of information.[15]

Third and final, the two main steps in the implementation (as discussed in 4.2.1), viz. template enumeration and theory generation, need to be augmented as well. In **template enumeration**, if in the original system a single rule is generated (containing only unground atoms), then in the system extended with constants a whole set of rules would have to be generated instead, containing the rule with only unground atoms, but also all rules generated using all possible combinations of substitutions with all the objects in the type signature. In **theory generation**, concerning parameter N_B (the max number of body atoms in a rule), a similar question as pertaining to the cost definition comes up: whether or not to treat ground and unground atoms in the rules equally. In case the choice is made to split up N_B into two separate parameters, theory generation would become more complex, as each of the new parameters would have to grow independently, adding a step to the generation process.

6.3 Negation

In the Apperception system, there is no negation: adding some form of it could work as syntactical sugar, making the produced explanations more concise. Two kinds of negation are *negation as failure* (NAF) and *classical negation*: the former is used for a query that fails in the sense that it do not succeed (notation: `not` p), whereas the latter is used for a query that fails in the stronger sense that its negation succeeds (notation: $\neg p$) [70, p.1-2]. As mentioned (6.1.2), for my investigation I have chosen not to extend the original input. Here, that means that any negation the system works with, is negation it has deduced itself; most standardly, this negation takes the form of NAF. NAF may be implemented by extending Evans et al.'s rules from Horn clauses to *normal logic programs* (NLPs), which, in addition to allowing functions in terms, allow for NAF in the body of a rule [1, p.5][16].

It is possible to add classical negation to the system as well: Gelfond & Lifschitz have devel-

[14] A question that does arise, in that case, is what ratios the costs of the ground atoms in the initial conditions, the unground atoms in the rules, and the ground atoms in the rules have.

[15] The two other interpretations, viz. not counting the ground atoms, and counting them as less costly than unground atoms, seem illogical given the arguments just provided.

[16] For a brief formal introduction to logic programming with stratified negation (including notes on its complexity), see [61, p.393].

oped *extended logic programs* (ELPs) for this purpose, which combine NLPs with classical negation [70][17]. Classical logic differentiates between *true*, *false*, and *undecidable*. In NLPs, NAF binds together the classical *false* and the classical *undecidable* into NAF's *false*, but in ELPs, the added classical negation allows for the classical trichotomy to be preserved. However, in systems where the Closed World Assumption[18] (CWA) holds, NAF and classical negation coincide, thus reducing ELP to NLP [70, p.7]. Since the CWA holds in the Apperception system[19], adding classical negation next to NAF would not make a difference.

When working with NAF, sometimes multiple contradictory pieces of information may be deduced, leading to conflicts. A way of solving that difficulty is to add *stratification*. The idea of stratification is to group the (available and deducable) information into layers, which can then be placed in a hierarchy of importance that determines which pieces of information are prioritized. Adding stratified NAF to the Apperception system is not straightforward: especially the question of how to get the stratification (i.e. the partitioning of information and the ordering of those parts) is difficult. Since the stratification is over specific pieces of information that are deduced by the system, it is complicated to handcode it beforehand; instead, the system might try to learn it as well, but then stratification iteration has to be added to the algorithm, which is already not very scalable with only template iteration. That said, as discussed (6.1.1), the goal is to reduce complexity of the output—not of the system itself—and so for this goal adding stratified NAF might be worth the effort. Due to time constraints, I did not investigate this extension further, but it would be an interesting topic for future research[20].

6.4 Disjunction

In the Apperception system, rule bodies are conjunctive and rule heads are atomic. A potentially interesting[21] extension would be to allow disjunction in the rules; it is generally acknowledged that disjunctive rules for knowledge representation are useful [74, p.365], i.a. because they can increase EP (though this is not our goal) and allow for problems of lower complexity to be represented in a simpler and more natural way (precisely our goal—so adding disjunction may be worth the increase in EP) [75, p.499].

A theory with disjunctive rules would create a family of traces, and it would make sense of a sensory sequence if one of the generated traces covered the given sensory sequence. Allowing disjunction in the **body** of a rule would not add EP, since the original system can represent the same information as multiple separate rules with the same head[22], however, it may still increase conciseness and thus quality of explanation. Allowing disjunction in the **head** of a rule, on the other hand, *would* add EP, since this allows for the modeling of new situations, viz. situations with indefinite knowledge [76, p.1]. For **static** rules, disjunction models the ability to, mutually exclusively, deduce multiple pieces of information within the given time step, so the information that can be deduced within every state is non-deterministic. For **causal** rules, disjunction models a fork in the temporal path of the development of the system, making the trace of the theory multidimensional (similar to a decision tree, instead of a line), so non-deterministic.

Richard Evans, Pushmeet Kohli, and Marek Sergot have published another paper, together

[17] Note that the set of ELPs contains the set of NLPs (and definite logic programs) as a special case [71, p.1].

[18] Originally defined as "[t]he implicit representation of negative facts presumes total knowledge about the domain being represented" [72, p.6], or put differently: if some fact is true in the system, then that fact is also known by the system, and conversely, those facts which are not known by the system can be concluded to be false in the system.

[19] Since the trace of a theory "is a complete and determinate description of the whole time-series" [1, p.7].

[20] Richard Evans suggested this research topic in our email conversation [73].

[21] Note that it may be argued that making the system disjunctive (or non-deterministic in another way) is undesirable: allowing disjunction adds possibilities, which is precisely the opposite of the goal of designing a predictive (and thus deterministic) system.

[22] Note that this is not the case for *exclusive or* (XOR).

with Matko Bošnjak, Lars Buesing, Kevin Ellis, and David Pfau, in which they extend the Apperception system to be able to handle inputs that contain disjunction [66]. Their goal in that paper is to adapt the original system so that it is able to work with raw input, rather than with already discretized input. The adapted system includes a neural network that maps raw sensory input to disjunctive input: hence the extension. My approach differs from theirs (in addition to my not looking into changing the implementation of the model), in that I am not looking into changing the language of the input, but into changing the language of the system only. One way this might be done is to expand the rules from Horn clauses to disjunctive logic programs (DLPs), which are normal logic programs (NLPs, as discussed in 6.3), but with the added possibility of disjunction in the head[23].

To explore the possibility of disjunction in the rules further, consider the following infinite-domain scenario. Suppose the situation is such that for all inputted data patterns can be found and thus predictions can be made, except for the data concerning one element in the system: say, whether some object a is *on* or *off*. This uncertainty could be modeled in the system as $on(a) \vee off(a)$. However, in this case, it would not make sense for the system to, during the creation of a predication, collapse to one of the two options ($on(a)$ or $off(a)$), since that collapse would not be based on anything (there is no predictability). However, on second thought, the probability distribution will most likely not be spread equally across the (in this case: two) options, so then it would make more sense to always collapse to the option with the highest chance (i.e. that appears most frequently in the sensory sequence). But, taking this line of reasoning further, then that most likely option (say $on(a)$) could just be taken as the predication, instead of first modeling a disjunction of all options and then collapsing to one of them...

At this point even more questions come to mind. First of all, what mechanism decides between the options? And with what chances? With equal chances? If with equal chances, then oftentimes not representative of real world scenarios. If with unequal chances, then how are the exact chances determined? Are these learned as well? But if so, then this becomes more of an optimization algorithm...

Secondly, if the input sensory sequence represents one path of the multidimensional trace that is to be deduced, how then can rules governing all those paths ever be deduced from that one given path? And if some rules are deduced, then do they mean anything? It is, after all, possible to make up many more rule sets that make sense of the input sensory sequence by playing around with the different rules and chances in their heads... So then the questions arise: How does such a rule relate to the goal of explainable AI? And, does this provide sufficient explanation?

Lastly, what about *exclusive or*? *Exclusive or* is a more limited version of *inclusive or*, creating a less multidimensional trace (for $a \vee b$: just a branch for a and one for b, instead of having a third as well for $a \wedge b$). Hence, the difference between *exclusive or* and *inclusive or* is whether or not there is the possibility for multiple options in the head of a rule to be true simultaneously. Especially when the predicates in the head are in the same predicate group (i.e. an object can satisfy only one of them at any time), using *exclusive or* can be useful.

Hence, there are quite some difficulties in the theoretical realm. Additionally, in the practical realm, a lot of adaptations would need to be made to the original system, fundamentally redesigning the Apperception engine algorithm. A further complication is the increase in RC associated with the increase in EP, making the implementation of the system harder (too hard?) to execute [77, p.179]. The ultimate question, then, is whether the win in understandability is worth the theoretical and practical difficulties encountered when extending the system with disjunction.

[23] For a brief formal introduction to DLP (including notes on its complexity), see [61, p.398]; for an extensive comparison of DLPs and NLPs, see [76].

6.5 Conclusion

To conclude this chapter, I will answer the second research question 'What are potentially useful extensions to the language of the Apperception model in the context of XAI?' by recapping my findings.

When extending the language of a formal system, most standardly the objective is to change the balance between EP and RC, viz. to increase EP, while not increasing RC too much. Here, this is not the case. My interest in looking into extending the language of the Apperception model was to see whether the quality of the solution it produced could be improved: in the context of XAI, 'quality' means 'explanability', which, in the context of the Apperception model, can be reduced to 'conciseness'.

So, to answer the second research question 'What are potentially useful extensions to the language of the Apperception model in the context of XAI?': I have focused on extending the rules of the model, for which I have briefly considered a number of different extensions, out of which I examined three in more depth: adding constants, negation, and disjunction.

In my discussion of adding constants (which also turned out a general test case for adding an extension), I made the case that an investigation into extending the language of the Apperception model in the context of XAI (i.e. with the motivation of increasing explainability) should include (or be preceded by) a reconsideration of the original cost definition, as it plays a crucial role in the model and therefore in an investigation. I gave the following two arguments in favor of diverting from the original cost definition. First, when working with a compound score, it is difficult to provide a general proof showing that the extended model under consideration produces equally or more concise solutions when compared to the original model. Second, when working with a single cost definition, potentially valuable solutions are lost, as the model's formalization of the intuitive notion of 'making sense' is not perfect (making it sensible to keep multiple different options for solutions open and have a human decide which of those is most appropriate in the given task) (I argued for this in 5.1.3).

In my discussion of adding negation, I concluded that enhancing the model with stratified negation as failure would definitely be an interesting option in the context of XAI. Though building this extension into the model and adapting the engine (or changing implementation altogether) is a big effort, it is most likely worthwhile.

Finally, in my discussion of adding disjunction, I mainly explored which questions come up when considering the theory and the implementation of this extension. In the end, I think adding disjunction might not be very fruitful; because of the many uncertainties, but also because it alters the model drastically, potentially to the point where it might not be able to predict / retrodict / impute anymore, as it stops being deterministic.

In conclusion, there are a number of extensions to the language of the Apperception model that might allow for conciser solutions, however, future work in this direction is necessary to give a definite answer.

7 Conclusion

In this concluding chapter, I will briefly discuss whether Evans et al., with their design of the Apperception system, reach the goals of creating a high-performing, explainable system that formalizes the intuitive notion of "making sense".

The goal of formalizing the intuitive notion of "making sense" is specific to the Apperception system. The model creates theories, which it then filters through three layers before returning one of them as a solution: it must be an 'interpretation' of the given sensory sequence, it must satisfy the four unity conditions, and it must be lowest-cost as defined by the cost definition. The definition of 'making sense' as given by Evans et al. only includes the first two layers, however, the third layer of the cost definition is of crucial importance and should, in my opinion, also be included. Additionally, since this third layer is so important, it is sensible to do more research into alternative definitions and their effect on all three subgoals.[1]

The goals of high performance and explainability are specific to the research field of XAI. The engine is high in performance (in terms of predictive accuracy) compared to competing approaches such as inductive logic programming, though it is not (yet) at the performance level that modern machine learning approaches are (generally) at. However, this is pioneering work, and so the system can still be developed substantially: given its current performance, there is a promising future for this kind of work. The system is explainable, in the sense of producing concise, human-readable solutions. Since the rules that the system learns are only of one specific form (viz. Horn clauses), it is interesting to investigate whether making extensions to the language of the model improves explainability, i.e. increases conciseness of solutions. Exploratory work into adding constants, negation, or disjunction to the rules gave mixed results. Adding constants is doable, though whether it is (significantly) beneficial for decreasing the size of the explanation is not clear yet. Adding negation seems most hopeful; specifically stratified negation as failure would be interesting to investigate further, despite the considerable adaptations to the model and engine that are necessary. Adding disjunction could turn out very worthwhile, though there are a lot of unknowns, both in the practical and in the philosophical sense. More in-depth research is necessary before any definitive conclusions may be drawn, for which the research done in this thesis may be a starting point.[2]

In conclusion, the Apperception system is a good early attempt at building XAI and may serve as foundational work for future research.

[1] These points are worked out in more detail in the conclusion of chapter 5.
[2] These points are worked out in more detail in the conclusion of chapter 6.

Bibliography

[1] Richard Evans et al. "Making sense of sensory input". In: *Artificial Intelligence* 293 (2021), p. 103438.

[2] Jieshu Wang. *Symbolism vs. Connectionism: A Closing Gap in Artificial Intelligence*. http://wangjieshu.com/2017/12/23/symbol-vs-connectionism-a-closing-gap-in-artificial-intelligence/. Published: 2017-12-27. Accessed: 2020-11-26.

[3] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (4th edition)*. Pearson, 2020, pp. 1–1136.

[4] Wikipedia contributors. *History of artificial intelligence — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=History_of_artificial_intelligence&oldid=1055962348. [Online; accessed 25-November-2021]. 2021.

[5] Susanne Bobzien. "Ancient Logic". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Summer 2020. Metaphysics Research Lab, Stanford University, 2020.

[6] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (2nd edition)*. Prentice Hall, 2002, pp. 1–1080.

[7] Lilian Edwards and Michael Veale. "Slave to the algorithm: Why a right to an explanation is probably not the remedy you are looking for". In: *Duke L. & Tech. Rev.* 16 (2017), pp. 18–84.

[8] Margaret A. Boden. "GOFAI". In: *The Cambridge Handbook of Artificial Intelligence*. Ed. by Keith Frankish and William M.Editors Ramsey. Cambridge University Press, 2014, pp. 89–107. DOI: 10.1017/CBO9781139046855.007.

[9] John Haugeland. *Artificial Intelligence: The Very Idea*. Cambridge: MIT Press, 1985.

[10] Allen Newell and Herbert A Simon. "Computer science as empirical inquiry: Symbols and search". In: *ACM Turing award lectures*. 2007, p. 1975.

[11] Marta Garnelo and Murray Shanahan. "Reconciling deep learning with symbolic artificial intelligence: representing objects and relations". In: *Current Opinion in Behavioral Sciences* 29 (2019), pp. 17–23.

[12] Selmer Bringsjord. "The logicist manifesto: At long last let logic-based artificial intelligence become a field unto itself". In: *Journal of Applied Logic* 6.4 (2008), pp. 502–525.

[13] Demis Hassabis et al. "Neuroscience-inspired artificial intelligence". In: *Neuron* 95.2 (2017), pp. 245–258.

[14] *Human Robot Interaction Workshop on Explainable Robotic Systems*. 2018. URL: https://explainableroboticsystems.wordpress.com/ (visited on 02/19/2021).

[15] Heike Felzmann et al. "Towards transparency by design for artificial intelligence". In: *Science and Engineering Ethics* (2020), pp. 1–29.

[16] Nils J Nilsson. "The physical symbol system hypothesis: status and prospects". In: *50 years of artificial intelligence*. Springer, 2007, pp. 9–17.

[17] Cameron Buckner and James Garson. "Connectionism". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Fall 2019. Metaphysics Research Lab, Stanford University, 2019.

[18] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[19] Chris Nicholson. "Symbolic Reasoning (Symbolic AI) and Machine Learning". In: *PathMind AI Wiki* (). URL: https://wiki.pathmind.com/symbolic-reasoning (visited on 02/01/2021).

[20] Rosalie Iemhoff. personal communication. Jan. 14, 2021.

[21] Artur SD'Avila Garcez, Luis C Lamb, and Dov M Gabbay. *Neural-symbolic cognitive reasoning*. Springer Science & Business Media, 2008.

[22] Frank van der Velde and Marc de Kamps. "Neural blackboard architectures of combinatorial structures in cognition". In: *Behavioral and Brain Sciences* 29 (2006), pp. 1–72.

[23] Andrew Cropper, Sebastijan Dumančić, and Stephen H Muggleton. "Turning 30: New ideas in inductive logic programming". In: *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence* (2020), pp. 4833–4839.

[24] Richard Evans and Edward Grefenstette. "Learning explanatory rules from noisy data". In: *Journal of Artificial Intelligence Research* 61 (2018), pp. 1–64.

[25] Henry Lieberman. *Symbolic vs. Subsymbolic AI*. Lecture slides for MIT course "MAS S63: Integrative Theories of Mind and Cognition". Feb. 2016. URL: https://courses.media.mit.edu/2016spring/mass63/schedule/ (visited on 02/05/2021).

[26] Andrew J Lohn. "Estimating the Brittleness of AI: Safety Integrity Levels and the Need for Testing Out-Of-Distribution Performance". In: *arXiv preprint arXiv:2009.00802* (2020).

[27] Zhiting Hu et al. "Harnessing deep neural networks with logic rules". In: *arXiv preprint arXiv:1603.06318* (2016).

[28] Artur d'Avila Garcez et al. "Neural-symbolic learning and reasoning: contributions and challenges". In: *2015 AAAI Spring Symposium Series*. 2015.

[29] Douglas Heaven. "Why deep-learning AIs are so easy to fool". In: *Nature* 574.7777 (2019), pp. 163–166.

[30] Klaus Greff, Sjoerd van Steenkiste, and Jürgen Schmidhuber. "On the Binding Problem in Artificial Neural Networks". In: *arXiv preprint arXiv:2012.05208* (2020).

[31] Richard Evans et al. "Making sense of sensory input". In: *arXiv preprint arXiv:1910.02227* (2019).

[32] Adina L Roskies. "The binding problem". In: *Neuron* 24.1 (1999), pp. 7–9.

[33] Jerome Feldman. "The neural binding problem (s)". In: *Cognitive neurodynamics* 7.1 (2013), pp. 1–11.

[34] Michael H Herzog. *Binding problem*. Tech. rep. Springer, 2009.

[35] Himanshu Chandel and Sonia Vatta. "Occlusion detection and handling: a review". In: *International Journal of Computer Applications* 120.10 (2015).

[36] Benjamin Chandler and Ennio Mingolla. "Mitigation of effects of occlusion on object recognition with deep neural networks through low-level image completion". In: *Computational intelligence and neuroscience* 2016 (2016).

[37] Bernhard Preim and Charl Botha. "Image-guided surgery and augmented reality". In: *Visual computing for medicine: theory, algorithms, and applications (second edition)*. Morgan Kaufmann, 2014. Chap. 18, pp. 625–663.

[38] Christian Szegedy et al. "Intriguing properties of neural networks". In: *arXiv preprint arXiv:1312.6199* (2013).

[39] Anh Nguyen, Jason Yosinski, and Jeff Clune. "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 427–436.

[40] Ben Dickson. *Is AI research headed in the right direction?* https://bdtechtalks.com/2019/03/25/richard-sutton-artificial-intelligence-research/. Published: 2019-03-25. Accessed: 2020-11-23.

[41] Marta Garnelo, Kai Arulkumaran, and Murray Shanahan. "Towards deep symbolic reinforcement learning". In: *arXiv preprint arXiv:1609.05518* (2016).

[42] Ken Kansky et al. "Schema networks: Zero-shot transfer with a generative causal model of intuitive physics". In: *arXiv preprint arXiv:1706.04317* (2017).

[43] Qianli Liao and Tomaso Poggio. *Object-oriented deep learning*. Tech. rep. Center for Brains, Minds and Machines (CBMM), 2017.

[44] Qunzhi Zhang and Didier Sornette. "Learning like humans with Deep Symbolic Networks". In: *arXiv preprint arXiv:1707.03377* (2017).

[45] Gary Marcus. "Deep learning: A critical appraisal". In: *arXiv preprint arXiv:1801.00631* (2018).

[46] Amy Zhang et al. "Composable planning with attributes". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 5842–5851.

[47] Jiayuan Mao et al. "The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision". In: *arXiv preprint arXiv:1904.12584* (2019).

[48] Paul Voosen. "Computer says no: why making AIs fair, accountable and transparent is crucial". In: *Science Magazine* (July 6, 2017). URL: https://www.sciencemag.org/news/2017/07/how-ai-detectives-are-cracking-open-black-box-deep-learning (visited on 01/19/2021).

[49] Finale Doshi-Velez and Been Kim. "Towards a rigorous science of interpretable machine learning". In: *arXiv preprint arXiv:1702.08608* (2017).

[50] Ian Sample. "Computer says no: why making AIs fair, accountable and transparent is crucial". In: *The Guardian* (Nov. 5, 2017). URL: https://www.theguardian.com/science/2017/nov/05/computer-says-no-why-making-ais-fair-accountable-and-transparent-is-crucial (visited on 01/18/2021).

[51] Jan Broersen. personal communication. Sept. 26, 2021.

[52] Zachary C Lipton. "The Mythos of Model Interpretability: In machine learning, the concept of interpretability is both important and slippery." In: *Queue* 16.3 (2018), pp. 31–57.

[53] Alan B Tickle et al. "The truth will come to light: Directions and challenges in extracting the knowledge embedded within trained artificial neural networks". In: *IEEE Transactions on Neural Networks* 9.6 (1998), pp. 1057–1068.

[54] Bryce Goodman and Seth Flaxman. "European Union regulations on algorithmic decision-making and a "right to explanation"". In: *AI magazine* 38.3 (2017), pp. 50–57.

[55] Leilani H Gilpin et al. "Explaining explanations: An overview of interpretability of machine learning". In: *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*. IEEE. 2018, pp. 80–89.

[56] Mahmood Sharif et al. "Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition". In: *Proceedings of the 2016 acm sigsac conference on computer and communications security*. 2016, pp. 1528–1540.

[57] Ben Dickson. *The security threats of neural networks and deep learning algorithms*. https://bdtechtalks.com/2018/12/27/deep-learning-adversarial-attacks-ai-malware/. Published: 2018-12-27. Accessed: 2021-02-16.

[58] Riccardo Guidotti et al. "A survey of methods for explaining black box models". In: *ACM computing surveys (CSUR)* 51.5 (2018), pp. 1–42.

[59] Maya Krishnan. "Against Interpretability: a Critical Examination of the Interpretability Problem in Machine Learning". In: *Philosophy & Technology* (2019), pp. 1–16.

[60] Victoria Krakovna et al. "Specification gaming: the flip side of AI ingenuity". In: *DeepMind Blog* (2020). URL: https://deepmind.com/blog/article/Specification-gaming-the-flip-side-of-AI-ingenuity.

[61] Evgeny Dantsin et al. "Complexity and expressive power of logic programming". In: *ACM Computing Surveys (CSUR)* 33.3 (2001), pp. 374–425.

[62] Richard Evans. personal communication. Apr. 14, 2021.

[63] Stephen Muggleton and Luc De Raedt. "Inductive logic programming: Theory and methods". In: *The Journal of Logic Programming* 19 (1994), pp. 629–679.

[64] Tomer D Ullman, Noah D Goodman, and Joshua B Tenenbaum. "Theory learning as stochastic search in the language of thought". In: *Cognitive Development* 27.4 (2012), pp. 455–480.

[65] Anat Biletzki and Anat Matar. "Ludwig Wittgenstein". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Winter 2021. Metaphysics Research Lab, Stanford University, 2021.

[66] Richard Evans et al. "Making sense of raw input". In: *Artificial Intelligence* 299 (2021), p. 103521.

[67] Richard Evans. personal communication. May 20, 2020.

[68] R Kowalski. "The relation between logic programming and logic specification". In: *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences* 312.1522 (1984), pp. 345–361.

[69] Hanna M Pasula, Luke S Zettlemoyer, and Leslie Pack Kaelbling. "Learning symbolic models of stochastic domains". In: *Journal of Artificial Intelligence Research* 29 (2007), pp. 309–352.

[70] Michael Gelfond and Vladimir Lifschitz. "Classical negation in logic programs and disjunctive databases". In: *New generation computing* 9.3-4 (1991), pp. 365–385.

[71] Marek Sergot. *Lecture notes for course 491 Knowledge Representation: Extended Logic Programs*. Feb. 2014.

[72] Raymond Reiter. "On closed world data bases". In: *Readings in artificial intelligence*. Elsevier, 1981, pp. 119–140.

[73] Richard Evans. personal communication. May 29, 2020.

[74] Thomas Eiter, Georg Gottlob, and Heikki Mannila. "Disjunctive datalog". In: *ACM Transactions on Database Systems (TODS)* 22.3 (1997), pp. 364–418.

[75] Nicola Leone et al. "The DLV system for knowledge representation and reasoning". In: *ACM Transactions on Computational Logic (TOCL)* 7.3 (2006), pp. 499–562.

[76] Heng Zhang and Yan Zhang. "Disjunctive logic programs versus normal logic programs". In: *arXiv preprint arXiv:1304.0620* (2013).

[77] Christoph Koch, Nicola Leone, and Gerald Pfeifer. "Enhancing disjunctive logic programming systems by SAT checkers". In: *Artificial Intelligence* 151.1-2 (2003), pp. 177–212.