Black-box systems, multi-level explanation, and cognitive architectures.

Thesis MSc Artificial Intelligence

Nardi Lam

Student number: 6313159

Supervisor: Brandt van der Gaast

Utrecht University

Contents

Introduction

As artificially intelligent (AI) systems take on increasingly many human tasks and further integrate into society, the field of **explainable AI** is becoming increasingly relevant. Explainable AI tackles the problem of explaining the decisions made or outputs produced by AI systems[1]. In order to improve task performance AI systems become more and more complex, to the extent that even for the designer of a system it may be hard to explain and justify the decisions that the system makes. This leads to systems being applied which may eventually make unexpected and undesirable decisions, and a general lack of trust in AI systems.

Large parts of contemporary AI systems are made up of **black-box systems**: systems which are only defined by associations between their inputs and their (desired) outputs. When judging whether such a system performs its task well, the internal structure of the system does not matter, and so they are often very flexible and their performance can be greatly fine-tuned. However, to justify a system's decisions, some of its internal structure must be clarified, often at the cost of performance. The challenge of explainable AI is then finding the right balance in this regard: the right amount of flexibility to produce new and interesting solutions to cognitive tasks, together with a internal structure which sufficiently clarifies the basis for the system's decisions.

The connection between explanation in AI and explanation in psychology plays an important role in this thesis. This connection is motivated by the fact that AI and cognitive psychology share a significant amount of history. In particular, the 'black-box

[1]In AI, the terms 'decision', 'action', 'output' and 'result' are generally used interchangeably. From an agnostic information processing perspective, these are all just the signal(s) the system outputs to the environment, but depending on the context one of the terms may be more appropriate than the others. If the signal is used to move a robot arm, it can be described as an action, if it is used to inform another action, it may be called a decision. I prefer generic terms like 'output' when discussing the more theoretical aspects and terms like 'decision' when discussing practical implications, to emphasize the consequences these outputs may have when viewing the systems as actors in a societal context.

machine learning models' which are commonly applied today have their origin in cognitive psychology, where they are known as **connectionist models**[2]. The difference is that in psychology, these models were applied not just to perform useful tasks, but also in an attempt to explain something about human cognition. One can see the contemporary problem of explainable AI as a part of this larger task: in order to explain human cognition via connectionist models, there must first be some way to obtain the desired kind of explanation from such a connectionist model, which is essentially the challenge of explainable AI. For this reason, some research in cognitive psychology is tackling very similar problems to that of explainable AI.

My analysis of explainable AI is based on a few assumptions or "guiding principles", which may clarify some aspects of the approach taken. These principles are as follows:

1. Understanding complex AI systems is a desirable goal in and of itself. In some cases, the utility of explainable AI may be framed as mostly instrumental, for example to judge the reliability of a system's performance in the future based on its past decisions. A better understanding of the system will naturally help in making such a judgment. However, I will treat the understanding of AI systems as more of an intrinsically motivated goal, and therefore what constitutes an "understandable" AI system is to a large extent determined by whether people report to understand it, rather than some objective measure of "transparency" or "predictability". This leads to a *pragmatic* view of explanation in AI. That is, the quality of explanations should be evaluated by the amount in which they help people understand AI systems. The questions which explanations need to answer can be similarly determined.

2. We want to understand complex AI systems in a way similar to how we understand humans. This is commonly known as Daniel Dennett's **intentional stance** (Dennett, 1987): it is when we ascribe concepts from human experience, such as beliefs and desires, to objects which we do not (necessarily) expect to actually possess these things. Even so, this way of viewing systems greatly helps us understand them. This idea then gives rise to two more principles:

   (a) We should not expect AI systems to be less complex than natural cognitive

[2]For an early influential account on connectionist systems, see Smolensky (1988).

systems (i.e. brains)[3]. In terms of explanation it can be said that "biological cognizers and ML-programmed computers afford a similar diversity of explanations and epistemically relevant elements" (Zednik, 2019, p. 7). Therefore, techniques which have been developed in psychology to explain human cognitive processes may be useful to explain artificial cognitive processes as well, in a way which does not restrict the scope of 'explainable systems' to only systems which are limited in their complexity compared to the human brain.

(b) We should also not expect the behavior of AI systems to be *more* explainable than human behavior[4]. If we suppose that AI systems may be as complex as the brain, it does not seem reasonable to ask that they are simultaneously more understandable. Of course, this says something about the kind of explanations people should *accept*: if an explanation is good enough when provided by a human agent, it should be good enough when an artificial agent provides it as well. Under this assumption, the kind of explanations people generate can provide a benchmark for what constitutes an 'explainable system'.

Motivated by these principles, I would like to investigate the utility of cognitive architectures in explaining AI systems. Cognitive architectures attempt to describe the structure of human cognition as a whole, which makes them ambitious projects. However, because they make few assumptions on the complexity of the cognitive system studied, this also enables us to extract a lot of useful knowledge from them about how these immensely complex systems may be explained. Moreover, because they are designed to describe human cognition, they also provide the possibility to make analogies between human and AI behavior and explanations thereof.

[3]At the moment, some AI systems are arguably less complex than natural systems, and therefore treating the two as comparable might seem 'overkill'. However, over time these systems will be further developed to attempt virtually every cognitive task humans can perform, which will lead them to become increasingly complex. Furthermore, it seems unreasonable to assume we will always be able to 'do better' than nature, in the sense that we will always be able to generate the same behavior as a complex natural system using a less complex artificial system. Therefore, we shouldn't assume that artificial systems will be somehow more limited in their complexity than the human brain is, and that they are therefore easier to explain.

[4]This position is similarly argued for by Zerilli et al. (2018).

First, I will clarify what is meant by explanation, and look at a number of different kinds of explanations which may be given about a system or its behavior. Then, I will consider which of these explanations are the kind which we are currently missing in explainable AI. I will look at several common techniques which are applied in explainable AI, and go over their goals and limitations. I will then look at cognitive architectures, and evaluate certain representative examples of them in terms of their explanatory utility. Finally, I will provide some suggestions of how ideas from cognitive architectures may provide new ways of constructing explanations of black-box systems used in AI.

Chapter 1

What are explanations?

In this chapter, I will review a number of contemporary publications in the field of explainable AI to sketch the environment in which this study takes place. I will use them to discuss the relevant notions of *understanding* and *explanation*, and we will build up the necessary terminology to properly evaluate cognitive architectures in the context of explainable AI.

Recently, a number of authors have proposed that to construct useful explanations of AI systems, it is necessary to take into account things we know about the human psychology of explanation and understanding. It must be taken into account what properties of explanations are effective in providing the understanding to the one receiving the explanation. Observations in the social sciences about these properties of explanations have been summarized by T. Miller (2019). Additionally, other authors (Mittelstadt et al., 2019; Páez, 2019) have formulated arguments about what kind of explanations are important to AI on a theoretical basis.

These observations and arguments are helpful to eventually ascertain the explanatory utility of cognitive architectures. However, to understand and compare their viewpoints it is first necessary to build up an overview of what exactly constitutes an explanation, and how different kinds of explanations may complement each other.

1.1 Defining 'explanation'.

The simplest definition of what an explanation is would be "an answer to a why-question" (as used for example by Van Fraassen, 1988, p. 138). However, this definition is very broad and on its own not very enlightening. A good point is made by Páez (2019) regarding the terminology used by AI researchers. When talking about 'explaining' AI, what exactly constitutes an explanation often remains vague, and many different kind of methods for understanding an AI system are lumped together. According to Páez, traditionally in scientific explanation, an explanation of some event x "must reveal, depending on which theory of explanation one adopts, either the true causal structure of x or the natural laws that determine x or its relationship with factors that make x more or less probable" (Páez, 2019, p. 445). Other methods of providing one with an understanding of x are categorized by Páez as "alternative sources of understanding". With regard to the usage of this terminology by other authors, he writes:

> "[Many] authors in the field refer to these alternative paths to understanding as "explanations," a usage that threatens to trivialize the term. If whatever makes an opaque model or its decisions better understood is called an explanation, the term ceases to have any definitive meaning. My argument throughout the paper has only focused on the notion of explanation as it has been traditionally understood in the philosophy of science and epistemology (e.g. , causal models, covering-law models, probabilistic approaches, etc.). It is in this sense that there are alternative sources of understanding." (Páez, 2019, p. 450)

Accordingly, it might be wise to restrict 'explanation' to a stricter definition such as the one Páez employs. However, this tendency of AI researchers to call many different sources of understanding 'explanations' does make sense from an intuitive point of view. For example, when people ask for explanations in ordinary situations, it is often acceptable to deliver an analogy relating the concept to be explained to one which is well understood by the person asking the question. In the interest of accessibility (which seems fitting for this topic), I would like not to stray too far from the broader common-sense definition of an explanation. As explainable AI is an interdisciplinary research field, people will come into it with different preconceptions of what terms mean, so to get everyone on the same page extra clarification is often needed. When assigning general terms a very precise meaning, the clarification needed to get everyone on the same page might eventually be greater than using new, more specific terminology, because people will make more assumptions about the meaning of a term if it already has a widespread 'common-sense' meaning.

For that reason, I will use the bare term 'explanation' only in its broadest sense. If we consider *explaining* to be a communicative act, an **explanation** is simply that which is communicated[1]. More specifically, it is communicated by some **explainer** to some **explainee**, with the aim of augmenting the understanding of the explainee in some way. How this happens exactly and whether it is successful is dependent on all three factors: (1) who is doing the explaining, (2) who is being explained to, and (3) how is it explained.

The idea that the goal of an explanation is to provide understanding, and that all these three factors come into play, leads to a *pragmatic* theory of explanation. Many traditional theories of explanation are *nonpragmatic* in the sense that they presume there is a *right* way to explain things, usually in the form of some set of conditions that can be enumerated, and that these conditions primarily concern the form and content of the explanation, not the circumstances in which this explanation is communicated, by who, or to whom. Traditional theories of explanation try to "formulate a set of objective, nonpragmatic criteria that [they] think all scientific explanations must satisfy to be evaluated highly. These criteria [are] universal in the sense that they are not to vary from one explanation to the next, but are to be ones applicable to all scientific explanations" (Achinstein, 2010, p. 137; as cited in Woodward, 2019, section 6.3). As such, our (very general) notion of explanation is pragmatic, in the sense than it does not presuppose any 'objective' criteria of what makes something an explanation, other than its usefulness in providing understanding. However, as we further distinguish between specific kinds of explanations, I will gradually reintroduce these more formal criteria under more specific terms.

1.2 Types of understanding and types of explanations.

There are various different kinds of explanations, and each may provide us with different kinds of understanding. To be able to concretely discuss their explanatory properties and the kind of effects they (aim) to have, it is helpful to construct a 'taxonomy' of explanations of sorts.

To start with, we will consider the effect these explanations may have, which is the understanding they provide. Páez (2019) distinguishes between a few different kinds of understanding. Suppose we have some circumstances C, some system S and an event E

[1]This notion of explanation is similar to that of the pragmatic theory of explanation by Peter Achinstein, an overview of which is given in Woodward (2019, section 6.3). Note that I do not mean to signify any specific theory of explanation as being 'correct' here, but that I am only trying to use clear terminology.

which was somehow the result of the system acting. Then the types of understanding according to Páez are:

1. **Understanding-why**: 'Which aspects of *C* and *S* have contributed to the occurrence of *E*?'
2. **Objectual understanding**: 'How does *S* work?'
3. **Functional (purposeful) understanding**: 'What is the purpose of *S*?'

Understanding-why is the most specific kind of understanding, in that it relates to not only a system, but also a specific event which has occured or may occur. Objectual understanding is more general in that it is concerned with the system as a whole, or at least not with a specific scenario. One may speak of either 'understanding why the bus is late' or 'understanding vehicles and traffic'. The main difference here is the object of understanding: in the context of AI, understanding-why aims to understand *a decision of a model* while objectual understanding aims to understand *a model* (Páez, 2019, p. 452).

Apart from the object, these types of understanding provide similar perspectives: they concern understanding of the structure of the model and how it may perform in different scenarios. Functional[2] understanding has a different goal. It aims to give one an idea of 'why' a model is a certain way in terms of 'for what purpose'. For example, one may understand 'why there is a bus': because people need to go places, there are roads connecting places and it is efficient to transport multiple people at once over these roads. This type of understanding about an system's purpose may also be called *teleological understanding*, or as I will usually refer to it, *purposeful understanding*.

Related to the different kinds of understanding are different *types of explanations*. After all, as the goal of an explanation is to convey some understanding, when the understanding it

[2]'Functional' is meant here in the sense of 'purposeful', which slightly conflicts with **functional in the mathematical sense**, where it means 'in terms of functions (i.e. mappings)', which is not concerned with purpose. For example, someone with a computer science background might be inclined to think about 'functional' in this sense, as in programming contexts the term is used to describe a system which is constructed by combining (pure) functions which take some input and produce some output. This is not what is meant by 'functional' here. To avoid confusion, and to enable the use of 'functional' in the mathematical sense, I will usually refer to this kind of understanding as *purposeful understanding* instead.

aims to convey changes in a fundamental way, the explanation may need to do so as well.

The distinction between these different types of explanations goes back to the *Four Causes* model of Aristotle, which divides the ways in which something may be explained into four categories: *material, formal, efficient,* and *final* (Hankinson, 2001; as cited in T. Miller, 2019). The first two are concerned with the physical or conceptual properties of an object, and as such are concerned with objectual understanding (though at different 'levels' of the object[3]). An efficient explanation is concerned with *how something came to be,* or understanding-why. Finally, a final explanation is concerned with *what end or goal an object serves*[4]. As such, it leads to functional or purposeful understanding. This correspondence shows that these different types of understanding are indeed quite fundamental.

An account which nicely aligns with the distinction made by Páez (2019) between understanding-why and objectual understanding is that by Bermúdez (2005). When discussing psychological explanations of cognitive agents, he distinguishes between **horizontal and vertical explanation**:

> "*Horizontal explanation* is the explanation of a particular event or state in terms of distinct (and usually temporally antecedent) events or states. Horizontal explanations are singular and dated. That is, they specify relations between individual and identifiable events holding at a particular time. The paradigm is singular causal explanation – the explanation of the causal antecedents of a particular event." (Bermúdez, 2005, p. 32)

This notion of explanation clearly leads to understanding of the kind which Páez calls understanding-why: it explains why a certain event A (the 'output') occured by following its successive causes. On the other hand, we may then want to know *why* some event B causes event A. For that a more comprehensive understanding of the objects involved is needed, which is provided by vertical explanation:

[3]Of course Aristotle lived in a different time, but in the digital world it becomes much harder to distinguish between what a system is conceptually and its physical manifestation, and so it makes sense to group the two together for now. This distinction will return when multi-level accounts of explanation are discussed.

[4]Some things, most importantly human beings, are often considered to be *ends-in-themselves*, which makes purposefully understanding them a bit difficult. But at least for now and the foreseeable future, I think it's reasonable to call ourselves 'creators' of AI and assign purpose to them.

> “Suppose we ask why the window broke when it did. A horizontal explanation of the window’s breaking might cite the baseball’s hitting it, together with a generalization about windows tending to break when hit by baseballs travelling at appropriate speeds. [...] I can [continue to] ask why the window broke when the baseball hit it, [in which case I will not] be satisfied by having repeated to me the generalization that windows tend to break when baseballs hit them. [...] What I want to know is *why* those generalizations hold. I want to find out what features of the physical structure of glass make it the case that windows are fragile enough to be broken by baseballs [...] ” (Bermúdez, 2005, pp. 32–33)

As such, this distinction between horizontal and vertical explanation corresponds to the distinction between understanding-why and objectual understanding.

At this moment, it is quite clear what the form of a horizontal explanation might be: it should be some kind of a description of a sequence of causally related events. However, we have not looked at what it means to understand or explain an AI system ‘objectually’. What kind of explanations about such a system can be given, and what is their relation to each other? In the next section I will discuss some ideas about various *levels of explanation* which may be considered when trying to explain complex systems. These distinctions between ‘levels’ will help us to evaluate the role which different explanations play in explaining a system ‘as a whole’, and give us a framework which relates them to each other.

1.3 Multi-level accounts of explanation.

The combination of various explanations of a single system which each provide various levels of detail is called a **multi-level account** of explanation. There are multiple of these accounts, with many being similar but having subtle differences. To illustrate the utility of such an account, I would like to focus on one of the most prominent of these: the three-level account developed by Marr in the 70’s, which was later posthumously published in 1982. His research was focused on the human visual system and trying to understand it as an information-processing system. However, his ‘levels of explanation’[5] have gone on to be widely applied, as they are not so much concerned with the visual system specifically, but only require that the system which is being studied is somehow computational in nature or at least can be treated as such. Given the computational nature of AI, his ideas should certainly be relevant to the study of explaining AI systems.

His account is motivated by the observation that “almost never can a complex system of any kind be understood as a simple extrapolation from the properties of its elementary components” (Marr, 1982/2010, p. 19). We must instead “be prepared to contemplate different kinds of explanation at different levels of description that are linked, at least in principle, into a cohesive whole” (Marr, 1982/2010, p. 20). Additionally, he gives a definition for the notion of **representation**: “a formal system for making explicit certain entities or types of information, together with a specification of how the system does this” (Marr, 1982/2010, p. 20).

Using these ideas, an information-processing system *S* can be explained independently at three different levels. I will succinctly phrase these in terms of the questions they aim to answer, and then discuss their relationships in detail.

1. The **computational theory level (CTL)**[6]: *which computational task* does *S* perform and *why* is this suitable for the problem at hand?
2. The **algorithmic and representational level (ARL)**: *how* does *S* represent the information involved and which operations are applied to transform it from input to output?
3. The **implementation level (IL)**: *what happens* when *S* performs its task? In other words, how is *S* physically realized and which events play a causal role in producing the end result?

[5]An important aspect which is sometimes overlooked is that these levels are levels of *analysis* (Bechtel & Shagrir, 2015). These levels provide three different perspectives from which one can study and explain a system, but they do not give a blueprint of how a system *should* be structured. When developing a system which is multi-layered in some way, there is no need to stick to these three levels, and moreover there may not be a clear correspondence between the levels which exist in the system by design and these levels of analysis. Marr’s levels only serve to provide a general distinction between the different aspects of an information-processing system which may provide explanations of a different nature. However, if one’s aim is to develop a system which is easy to analyze using such a framework, it may be helpful to use these levels as a guideline in designing the system.

[6]The computational theory level is usually referred to as just the *computational level*, however I consider it important that the concept of a ‘computational theory’ is kept in tact.

Important to understanding the first level is the notion of a **computational theory**. This is a description of a computational system which consists of two parts, and Marr explains this using an example of a cash register (Marr, 1982/2010, p. 22). The first part is an abstract specification of the computation which the system performs, which in the case of the cash register is *the addition of prices for a set of items*. The second part is a justification for performing this computation: what goal does the system try to accomplish and why is this computation relevant? In the case of the cash register, the goal is to *determine the total price*, and we can enumerate a number of *constraints* which should be satisfied in this case. For example, buying nothing should cost nothing, and the order of items shouldn't matter. These constraints then naturally lead us to the mathematical operation of addition. This means the computational theory provides both a functional (in the mathematical sense) and a purposeful (i.e. why is this computation necessary?) explanation of a system. And as the name implies, these explanations of the system live at the CTL.

The ARL then gives a specification of the *representation* used. In the case of the cash register, how (i.e. in what format) are the prices stored? There are many ways to store numbers and there may be both theoretical reasons (informed by the CTL) and practical reasons (informed by the IL) for choosing one. Therefore, the ARL is 'linked' to both of the levels in that it should agree with them somehow, but it still provides a description of the system in a way which the other two do not. In addition to the representation, it specifies an *algorithm*: the operations which are applied on these representations to provide the result.

In some sense, if the goal of an algorithm is transforming data from one representation into another (i.e. into the one which is needed as output), choosing a representation and choosing an algorithm amount to two sides of the same coin[7]. The same applies in Marr's account: the algorithm and representation reside at the same level and may therefore

[7]When trying to develop systems of limited complexity, the relationship between algorithm and representation generally leads to an arbitrary delineation between the two. Much of the complexity of an algorithm may be transferred to the representation, by making the representation more complex in a way that requires less manipulations to be applied to it, and so simplifying the algorithm. However, the complexity does not disappear, it only shifts from one to the other. An example of this is the *time-space tradeoff*, where in many cases the running time of an algorithm can be shortened by making use of more memory (i.e. a larger representation). For this reason it makes sense to consider the two together when studying a computational process holistically.

be tightly linked. As such, the algorithm can be formulated using the properties of the representation. For example, if we want to add numbers which we know are represented using some positional notation, we may specify an addition algorithm which has rules for single digit numbers and for carrying. However, if we have Roman numbers such an algorithm would not work.

Finally, any formal representation may be stored in different ways, for example by writing it on a piece of paper of by storing it using computer memory. Similarily, an algorithm may be performed in different ways: a human can add numbers on paper using a carrying-algorithm while a processor might add the numbers in memory using essentially the same method. This leads us to the implementation level, where the actual realization of both the representation and the algorithm are specified.

Marr insisted that approaches which focus only on one level (usually, the IL, because it is the most easily observable) are lacking, and he motivated this in a way reminiscent of how AI researchers today motivate the need for explainability:

> "[Mechanism-based] approaches are genuinely dangerous. The problem is that the goal of such studies is mimicry rather than understanding, and these studies can easily degenerate into the writing of programs that do no more than mimic in an unenlightening way." (Marr, 1982/2010, p. 347; as cited in Cooper & Peebles, 2015)

If we are to understand "mechanism-based approaches" as those that aim to replicate certain (input/output) behavior, then the class of black-box machine learning models also falls under this label. In both cases, parts of the three-level description (such as the justification in the CT, or the representations used at the ARL) are neglected as long as they do not impact the performance of the system. Even at the time, Marr was surprised that this reductionist approach had been so prevalent in AI:

> "For far too long, a heuristic program of carrying out some task was held to be a theory of that task, and the distinction between what a program did and how it did it was not taken seriously.

As a result, (1) a style of explanation evolved that invoked the use of special mechanisms to solve particular problems, (2) particular data structures [...] were held to amount to theories of the representation of knowledge, and (3) there was frequently no way to determine whether a program would deal with a particular case other than by running the program." (Marr, 1982/2010, p. 28)

Somewhat disappointingly, some of these criticisms are still applicable to this day. The focus on validating many complex machine learning models by their empirical performance on test sets also leads to (3), in that it is difficult to verify that inputs which have not been observed, stored, and tested already cannot lead to undesirable outcomes. Additionally, there is often a barely any distinction between *what* and *how* a system is doing. The *Turing test*-inspired line of thinking, "if it seems to be performing well, we can assume it is doing the right thing", is still a prevalent way of going about things in contemporary AI. While Marr seems to be mostly concerned with the scientific ramifications of holding "heuristic programs" to amount to theories of physical systems, now that these programs are making decisions which affect human lives in very tangible ways the necessity of understanding both *what* a program does and *how* it does it is arguably more important than ever.

1.4 The role of Marr's levels in understanding.

Naturally, as Marr's levels are concerned with explanations, it makes sense to try to connect them to the types of explanations and understanding which we have seen before. Marr's levels aim to provide understanding about systems themselves, so they generally aim to provide *objectual understanding*. Additionally, the CTL and its concern with 'why' a system performs a certain task indicates that it also aims to provide *purposeful understanding* of a system.

In terms of the *Four Causes* model, an IL-explanation of a system can be said to provide explanations which are material (the physical realization of the representations used) and efficient (the physical realization of the algorithm) in nature. The CTL provides explanations which are formal (what is the system meant to do) and final (why does the computation the system performs fulfill its purpose). Taking these two levels together, it might seem that they give a reasonably complete account of the system's properties. This raises a question about the position of the ARL: what function does it have in this context, and how do its explanations differentiate themselves from those provided by the other two levels?

The ARL is special in that it is really only relevant to systems which are complex enough that we can meaningfully describe their structure at multiple *scales*[8]. When developing information-processing systems, there is almost always some use of *abstraction*, either explicitly (in the program design) or implicitly (by executing it on a multi-layered software platform). When abstracting, useful patterns which appear in several places in the system are identified and given a specification of their own, which may then be given an implementation in several ways. Bechtel and Shagrir (2015) refer to these patterns as *design principles*, and they argue that identifying this use of abstraction is in many cases essential to understanding the system properly, even if the system is not necessarily 'designed' as such. They cite examples from neuroscience and molecular biology, and conclude that:

> "[I]n any situation where the operations of the individual parts are organized in a complex manner, explaining how the mechanism works requires understanding the contribution of the organization. [...] [I]dentifying the relevant design principles and understanding what behavior they will generate under a range of implementations is different than determining what is actually performing the various roles in a specific mechanism." (Bechtel & Shagrir, 2015, p. 321)

They argue that the role of the ARL is non-redundant: some abstraction is necessary in understanding systems of high complexity, so the implementation level is not sufficient on its own. The computational theory level gives an abstract description of the system, but it may be highly idealized, and is only concerned with *what* a system does, not at all with explaining *how* a system functions. The ARL fills this gap by explaining the *how* of a system by abstracting over the bare implementation in useful ways.

1.5 A visual overview of explanation.

In this section, I will try to tie the various concepts I've discussed together as smoothly as possible. While I hope to not obfuscate any of the details, it is also important for further discussion to give a clear delineation of what I will consider to be the relevant objects of

[8]Marr himself refers to the micro- and macroscopic theories of gases when introducing the idea of a 'complex system' (Marr, 1982/2010, pp. 19–20).

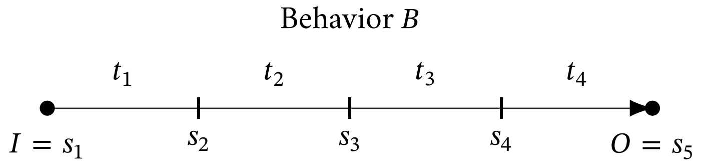

Figure 1.1: A single behavior B of a system S, consisting of a sequence of states $s_1 = I, s_2, ..., s_{n-1}, s_n = O$ and events $t_1, ..., t_{n-1}$ which cause the transitions between them, with here $n = 5$. This behavior is a kind of *horizontal explanation*.

explanation, the 'things' which may be explained about a system. To make this as clear as possible, I will try to describe this delineation in semi-formal language, and to make it understandable, I will present it using a visual representation.

Let S be some information-processing system which we wish to 'explain'. Suppose we execute S with some input I, and it produces some output O. The most straightforward way to 'explain' this process would be to construct a description of the events that take place: we inspect S and report on the steps which are taken to transform I into O. This aligns with what Bermúdez (2005) calls a 'horizontal explanation'. Formally, we will call such a description of the execution of S a *behavior* of S, and define it as a pair $B = (s, t)$. Here s is a sequence of 'states' of some length $n \geq 2$, which should start and end with (the state corresponding to) I and O respectively, and t is a sequence of $n - 1$ events which cause the transition from one state to the next. Such a behavior is visualized in Figure 1.1. As it describes a causal sequence of events, a behavior as defined here is in fact a horizontal explanation of a specific execution of S.

However, from arguments by Marr (1982/2010) and others we have seen that it may not be possible to give one definitive account of the states and events which make up a specific behavior. Instead, we can give different descriptions of this process at different *levels*. As such, we need to add another 'axis' to our space of explanations: one specifying the level. The first axis may be interpreted as 'process execution time', in the sense that it links chronologically ordered states of S together.

We now have a two-dimensional picture of the explanations we may give. If we accept Marr's framework of three levels of analysis, we may say that there are three possible

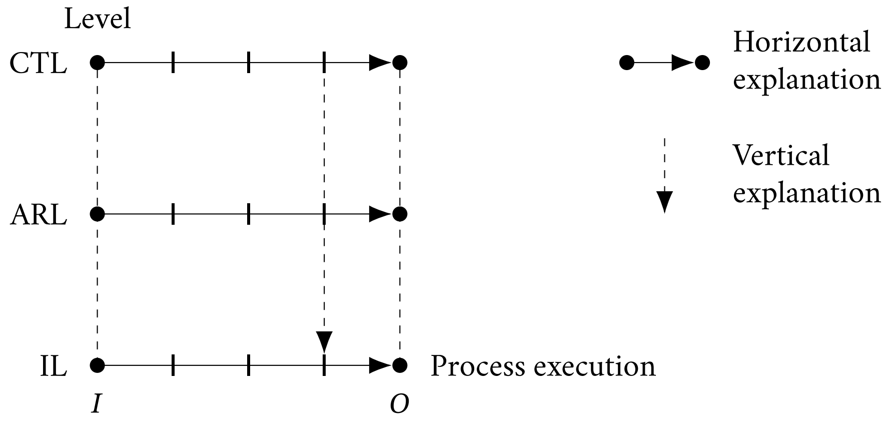

Figure 1.2: A two-dimensional picture of the types of explanations which may be given about a certain behavior. The horizontal arrows all represent descriptions of some behavior, but each takes place at a different level, which leads to a different interpretation of the states of the system and the way in which it transitions between them. A vertical explanation (shown by the vertical arrow) may look at one of these states or transitions and try to explain it further by 'zooming in' and relating it to lower-level descriptions.

values on the second axis. However, this idea of there being 'levels of explanation' can be used in a more general sense. Other authors have proposed frameworks which consist of more than three levels, leading to more possible kinds of explanations, and even then we may find certain explanations hard to categorize (they may fall 'between' levels). However, what these frameworks tend to have in common is that the levels are ordered, and that there exist 'lowest' and 'highest' levels. Therefore, any of these hierarchies of levels may be geometrically represented in a similar way, as long as we can determine given two levels which one is 'higher' than the other.

Now that we can account for different levels of explanation, we may also construct what Bermúdez (2005) calls 'vertical explanations'. Here, we look at one specific 'step' in the process from I to O, and investigate it at multiple levels to build a comprehensive explanation of that part of the process. These two kinds of explanations are visualized in Figure 1.2.

Finally, there is one more aspect that needs to be distinguished. Until now, this picture of the scope of explanation only takes into account a single 'run' of the system S, where it

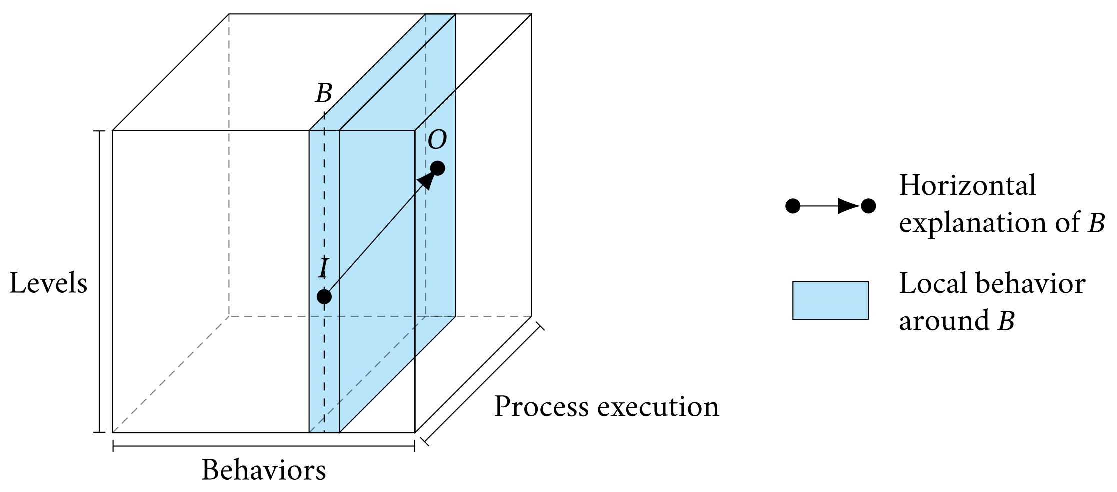

Figure 1.3: A three-dimensional view of the space of explanations. Here, the different ways in which a system may be described are differentiated on three axes: the behavior it shows (corresponding to the input provided, or the circumstances under which the system was executed), which stage in the process of the system execution is described, and the explanatory level at which a description is provided. The task of *fully explaining* a system according to this framework then becomes: finding explanations which together convey understanding of the system at every 'point' in this space, or every combination of the aforementioned three factors.

shows the behavior B of transforming input I to output O. Assuming S is deterministic and has no sources of information other than I, there will be a one-to-one mapping between the possible behaviors B and the possible inputs I. We may therefore also consider the space of *behaviors*, which can be identified with the space of inputs and therefore the space of *initial states*. This space may be arbitrarily large, but for now let us assume there is some mapping from this space onto either some finite set of discrete inputs, or some finite real interval. Then, we may display it as one more axis in our picture, leading to a three-dimensional *explanation cube* C. To define the scope of explanation, we will say that any explanation covers some part of C, and that to fully explain S is to construct a set of explanations which together cover the whole of C.

Now that we have a third dimension, we may view the space of explanations from two more angles. First is the *behavioral* plane (shown in Figure 1.4), consisting of the 'behavior'

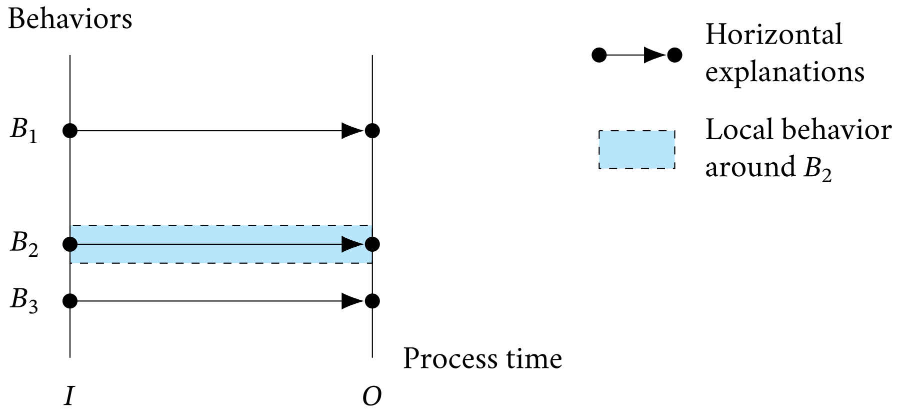

Figure 1.4: The 'behavioral plane', containing all possible behaviors of S. Each behavior corresponds to a different input for the system, and therefore (in most cases) different internal and output states. If we can calculate the similarity between inputs and map them onto a one-dimensional space in some way (in this case, such that the input for B_1 is more similar to that for B_2 than B_3 is), then we can interpret the slice around a behavior as a local approximation of the system around that behavior.

and 'process' axes. Here we can see all possible executions of S. If there is some way to quantify the 'similarity' of inputs I, it may be useful to imagine that different runs of S which are close in this picture are runs taking place under similar circumstances. This leads to a natural interpretation of a 'slice' of the behavioral axis as a *local approximation*: generally a model which tries to (reproduce and) describe the behavior of S, but which is valid for a certain range of inputs only.

The remaining face of this cube is the one which combines behaviors with levels of analysis. Here, the horizontal explanations around B which we've seen before exist in a 'slice' around the point on the behavioral axis which corresponds the input I associated with B. From this point, we can imagine trying to 'extend' these explanations along the behavioral axis, so that they do not only apply in the specific situation B which we considered before, but also in similar ones. This leads to a set of local approximations of S around B, one for each different level.

Additionally, we may attempt to generate explanations of S which apply to the system as a whole, not only to some particular (range of) behaviors. For example, a computational

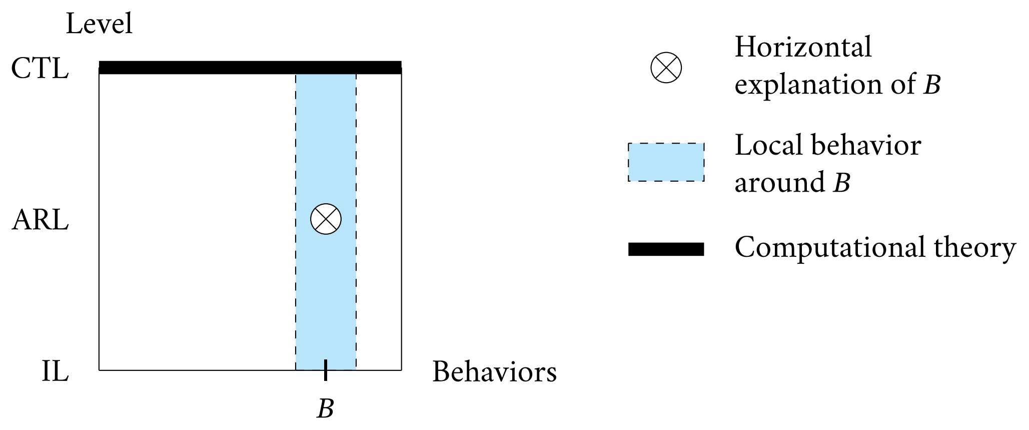

Figure 1.5: The 'static' view of the explanation cube. Here the 'temporal' axis is omitted, and as such horizontal explanations are reduced to points in the plane. On the other hand, here explanations may be easily distinguished by which of the system's behaviors they explain, and by the level they take place at. The functional, system-wide explanation given by the computational theory is shown by a thick horizontal line: it is independent of the specific behavior observed and as such it covers the whole system at that level.

theory as specified by Marr tends to encapsulate the whole of the system, because of its general nature. The task a system aims or is designed to solve is generally independent of which of the possible inputs it receives. Such an explanation therefore covers a vertical 'slice' of this space, where it explains *S* at a certain level for *all* possible behaviors comprehensively.

This kind of visual representation of the space of explanations will be useful later on, both to clarify what certain explanations explain about a system, and to argue for the importance of certain explanations by their 'coverage' of this space.

1.6 Explanation using the competence/performance distinction.

There are two specific 'categories' of explanations which I would like to describe using the terms we have developed thus far. These explanations are especially useful when describing agents performing cognitive tasks, which makes them relevant to our goal of explaining AI systems.

In linguistics, an influential account due to Chomsky distinguishes between two parts of linguistic cognition: **competence** on the one hand and **performance** on the other. The purpose of this distinction is to enable researchers to discuss the *ability* of a speaker to utter language separately from their actual *behavior* of doing so. *Competence* is concerned with this 'ability', and as such a description of linguistic competence is meant to answer questions such as "what kind of sentences *could* a speaker of English utter?" or "what makes a sentence a proper English sentence?"[9]. *Performance*, on the other hand, is concerned with the actual utterances, so a description of performance would instead be concerned with the more definitive "what kind of sentences *does* a speaker of English utter?". Additionally, the question of what constitutes a 'proper sentence' is not really relevant, as "a properly performed" sentence is not a function of the sentence but whether it is uttered and understood. This makes the notion of competence somewhat normative, while the notion of performance is not[10].

For Chomsky, this distinction is necessary because he posits that humans have an 'internalized grammar' which underlies their ability to construct linguistic utterances (Chomsky, 1965, p. 8). In order to discuss this internalized grammar, we need to divorce the *ability* for language from the *use* of language, hence the competence/performance distinction. However, in a more general sense these two ways of describing the linguistic 'faculty' (i.e. the part of cognition involved with language production and understanding) can be applied to other aspects of cognition as well. For example, one could consider the *ability* to recognize a certain object visually from the actual *performance* of recognizing it. These two perspectives provide a different kind of explanatory utility. In the theory of connectionist systems the distinction is also highly relevant: for example Smolensky believes that "[it] is likely that the view of the competence/performance distinction that arises from the connectionist approach will successfully heal a deep and ancient rift in the science and philosophy of mind" (Smolensky, 1988, p. 3).

[9]Note that these questions concern either the relationship between the task-related verb (to speak English) and the subject (the speaker) or between the verb and the object (the sentence).

[10]The concept of "a properly performed sentence" can be understood either pragmatically, as "a sentence which effectively conveys its message", or as "a sentence which is correct according to the competence description". In any case, the correctness is not determined by the description of the agent's performance alone, and so a performance description is not normative.

A **competence description** (or alternatively a **competence theory**) of a system articulates *what* is required of the system in order to perform 'well', and *how* the system is able to satisfy these requirements. Another way to phrase this is that the competence description articulates the **knowledge**[11] the system makes use of to achieve the performance, which may be either *implicit* or *explicit*. If this knowledge is explicit, the system 'is aware of' or otherwise explicitly puts to use the knowledge which the competence description describes, but if it is implicit (or *tacit*) the system does not need to be aware of nor explicitly use this knowledge as the competence theory describes it[12]. The complementary **performance description** then describes *how exactly* the behavior (or performance) of the system is realized and (usually) makes predictions about what behavior it will show.

We can try to place these two kinds of descriptions in the framework of Marr's levels. Marr himself refers to Chomsky's theory of competence as an example (in fact, one of the few) of a 'computational theory': "Chomsky's theory of transformational grammar is a true computational theory in the sense defined earlier" (Marr, 1982/2010, p. 28). However, if we follow the full conception of Marr's levels and the idea of competence, the correspondence seems to be slightly more complicated. According to Julius Moravcsik (1974)[13], the questions a description of competence aims to answer are threefold:

(a) What can be expected of someone who has the competence in question?

(b) Through what processes can the competence be manifested and what conditions are required by these processes?

(c) In view of answers to questions (a) and (b) what must be true of the agent having the competence?

[11]Here (and in most other parts of this thesis) the term 'knowledge' is used a bit more loosely than it generally is in epistemology. 'Knowledge' is roughly meant to mean 'information an agent uses to determine their actions'. This usage is common in psychological and AI literature. It can alternatively be read as 'putative knowledge' or even just 'information'.

[12]However, usage of the knowledge in the competence theory should of course at least be consistent with the behavior of the system for it to be a convincing description. The question of what makes it possible to be able to speak of 'implicit knowledge' is a complicated one, and is further treated in Section 2.5.

[13]Julius Moravcsik was a philosophical peer of Chomsky's, who in my opinion explicates Chomsky's ideas on competence in quite a clear and condensed manner. His discussion is also a bit more general than Chomsky's language-focused treatment.

Let us first try to answer these questions in the case of an example: the cognitively demanding task of *dishwashing*. Starting with question (a), what can be expected of someone competent in dishwashing? The answer is simply that, given a collection of dirty dishes (e.g. greasy) they will manage to clean them. Then for question (b): what kind of processes are necessary to accomplish this? Though there are many different ways to wash dishes, we can identify some common elements. A number of supplies are necessary: at least (1) water and (2) some kind of dish soap. Then, these supplies need to be used in a few different ways: (i) the dish soap must be applied to the dishes, (ii) the water must be used to rinse the dishes, (iii) the dishes must dry somehow. As for question (c), the agent in question must possess the knowledge and physical ability necessary to perform these tasks. It must know what dish soap is and how to apply it effectively. It must be able to rinse the dishes using water. Finally, it must have a way of drying the dishes, for example it must know that by simply waiting the dishes will dry and the task will be accomplished.

The facets of the process of dishwashing detailed above constitute a description of the *competence* of dishwashing. But, it is clear that not the whole process has been described here. There are many different ways to go about performing these steps: one could wash each dish one-by-one, or one could prepare some soapy water and submerge all the dishes in it. One could wash the dishes in various orders: start with the plates or start with the cutlery. These could be said to be 'algorithmic' differences in how to perform the task of dishwashing. Moreover, even a similar algorithm could be performed in different ways. One could use a brush or a sponge, and they could be cleaned by a human or some kind of robot (though 'dishwashing robots' generally use a pretty different kind of algorithm than humans do). Additionally, an experienced dishwasher will have a more refined technique in their handling of the dishes than an inexperienced one, and they will therefore be more efficient in their performance. These are further differences in the *implementation* of the dishwashing algorithm. Together all these varations make up a performance description of dishwashing: they are all instances of competent (i.e. acceptable) dishwashing, but they still differ wildly in their details.

In the general case, we can compare questions (a)-(c) to the kind of questions answered at each of Marr's three levels. At the CTL the relevant questions are *what* computation a system performs and *why* this computation is suitable for the task at hand. It makes intuitive sense to identify the 'competence' of the system as *being competent at the task which it aims to perform*. From that perspective, question (a) is clearly a CT-level question. An answer to question (a) would first need to illustrate what it means to be competent at the given task, which can be answered by giving a description of the properties the computation performed by the system must have, which is part of Marr's *computational theory*. For example, in the case of the cash register, an answer to (a) could be an enumeration of the properties of (successful) addition (e.g. it should give the same total price no matter the order in which the goods are supplied, etc.), which is the *what* part of the CT-level description of the cash register seen earlier. In addition, question (a) refers to the *purpose* or function of the system, as it asks 'what may be expected', which indicates someone external to the system requires something of it, indicating it has some *purpose*. This purposeful description is also included in the *why* of the CT.

Question (b) is of a different nature, as it discusses the *processes* which allow the competence to manifest. This differs from question (a) in two ways. First off, it makes specific reference to 'processes' and as such invite a mechanistic or otherwise causal explanation of the agent. Second, these processes should be (at least partially) internal to the agent and as such not included when giving a purposeful explanation of the agent. The computations or processes occurring inside the agent which are not externally observable may be exchanged by different processes which produce the same results, without straying from the CT-level description of the system. Rather than the CTL, an answer to question (b) would more likely be situated at the ARL, the level of *how*, where a description is given of the kind of operations the system uses to perform its task, or in other words, to manifest its competence. In the case of the cash register, one could say that a process which is required is an algorithm for addition, and the condition under which such an algorithm can function is if the numerical prices can be expressed in a compatible representation. In this case, the system is simple enough that there are no real sub-processes to speak of, but these would also need to be a part of the answer to this question if they are used to describe the overall process. For example, the process of dishwashing could be described as (1) scrubbing dishes with soap, (2) rinsing dishes and (3) drying dishes. In this case, as 'dishwashing' is described in terms of the three sub-processes, these need to either be elucidated as well, or deemed to be ‘fundamental’[14].

Finally, question (c) follows up on (a) and (b). Even if one takes (b) to be an ‘abstract’ specification of the processes involved (i.e. not an algorithmic one), (c) would then necessitate making it more concrete. If I possess a method by which I can add numbers, what must then be true of me? I must then be able to perform certain operations (e.g. carrying) and handle certain concepts (numbers).

From these observations, it seems that a ‘competence description’ is roughly equivalent to the combination of a CTL and an ARL description. If we assume that both Marr’s three levels and the combined notion of competence and performance provide ‘full’ descriptions of a system, the remaining parts of the two must also be equal. As such, the performance description would be roughly equivalent to the IL description. However, this equivalence is not strictly true. Unlike a full ARL description, a competence description is strictly concerned with what is *necessary* for the task to be (competently) performed. For example, one could say that the cash register, when dealing with positional numbers, *needs* to have a way to represent numbers and *needs* to be able to perform a carrying operation, or it cannot competently perform addition. But certain algorithmic details (for example, how it performs the carrying operation exactly given a certain numeral representation) which would be elaborated upon in a full ARL description might instead be relegated to the domain of performance.

Instead, I think the best way to relate Marr’s levels to the competence-performance distinction is to say that it ‘cuts’ the ARL in half. A competence description does not need to provide a full algorithmic description, but it should provide some sort of ‘blueprint’ for the algorithmic structure of the system. The details of the ARL will then be filled in by the performance description. One way to determine whether an algorithmic characteristic of the system belongs to the former or the latter, is by considering the following question: is this property *required* of the system *in order to perform its task*? For example, the system may *need* to use a certain *kind* of algorithm, but there might still be variability among the

[14]Of course, in any explanation the question of where one level of scale ends and another begins remains somewhat open to interpretation. The same applies when distinguishing between ‘an algorithm’ and ‘an implementation of an algorithm’ in Marr’s levels. If I read a pseudocode description of an algorithm and execute it by hand, is the implementation then me, the pseudocode, or both?

algorithms which satisfy this requirement.

A similar conception of a 'blueprint' of the ARL is given by Peacocke (1986). He proposes that in practice there exists a relevant level between the CTL (the highest level, so level 1) and the ARL (level 2), which he dubs "level 1.5". He characterizes this level as describing "the information on which the algorithm draws" (Peacocke, 1986, p. 101). This requirement of 'drawing upon' certain information does not specify an algorithm, but it does specify a "class of algorithms" (Peacocke, 1986, p. 112). As such, a level 1.5 description would give a similar kind of 'level 2 blueprint' as a competence description does. Instead of including another level between the CTL and ARL, I will instead just say that a competence description covers the CTL and *part* of the ARL[15]. Any algorithmic details which are not necessary to manifest the competence are relegated to the performance description, and only together do these provide a full ARL description.

We can identify that there are algorithmic concerns with regard to competence by setting up a sort of *Chinese Room*-inspired experiment. Suppose we have a competent adding machine *A* which works only for numbers up to a certain size (as is realistic for a physical computer). We could use this machine to construct a giant 'book' (e.g. a lookup table) of additions of pairs of numbers and their answer. Then, we could build another machine *B* which, when given two numbers, simply looks them up in the book and prints the corresponding result. Now a few different questions come to mind. Is *B* a competent adder? If so, is *B* competent in the same way *A* is competent? These questions do not seem trivial, yet from a CTL perspective *A* and *B* are equivalent. Interestingly, they do differ along in terms of the "the information upon which the algorithm draws" or at Peacocke's level 1.5. As such, there is a component to competence below the CTL, one situated either at a 'new' level, or in Marr's hierarchy, at the ARL.

The goal of this paragraph has been to embed the ideas of a 'competence description' and a 'performance description' in Marr's hierarchy. Note that e.g. a 'competence description' is a *description* of a process which takes place on a certain level, but not identifiable with a level itself. Also, a competence description is clearly meant to augment one's understanding of the system in some way (by answering questions (a)-(c)), so it is an explanation by our

[15]I think Peacocke wouldn't mind, as after all I am "drawing the same distinctions and making the same points, but in a different terminology" (Peacocke, 1986, p. 112).

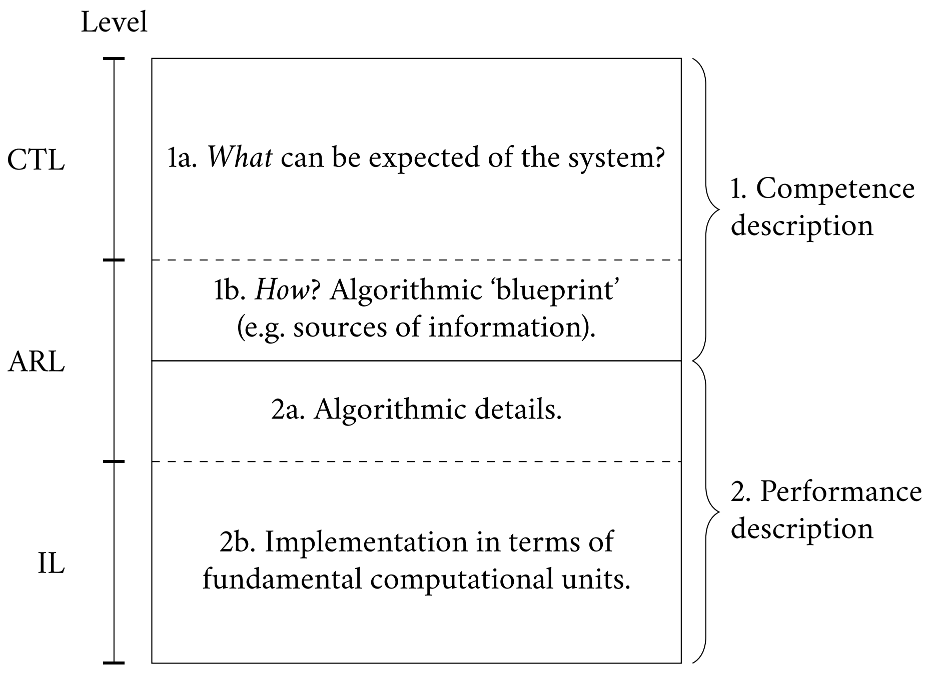
Level
CTL
ARL
IL
1a. *What* can be expected of the system?
1b. *How*? Algorithmic 'blueprint' (e.g. sources of information).
2a. Algorithmic details.
2b. Implementation in terms of fundamental computational units.
1. Competence description
2. Performance description

Figure 1.6: The competence/performance distinction situated in Marr's hierarchy of levels. The competence description (CD) is built up of two parts, 1a and 1b, and similarly the performance description (PD) consists of 2a and 2b. The CD provides the computational theory, and a 'blueprint' of the ARL. For example, it may provide an overview of the constructs involved and how they relate to each other, but will not provide an executable algorithm. The PD adds onto this by filling in the algorithmic details. One may imagine the blueprint to be more abstract (e.g. x and y are combined to form z) than the full ARL account (e.g. $x \in \mathbb{R}$ and $y \in \mathbb{R}$ are combined to form $z = \lfloor x/y \rfloor \in \mathbb{Z}$). Moreover, the PD also includes an executable implementation in terms of the fundamental units of the system. The border between the CD and PD lies somewhere in the ARL, and is determined by the question: 'Which algorithmic properties are necessary for the system to manifest its competence?'.

definition. Therefore we can place explanatory accounts of a system's 'competence' and 'performance' in the visual space of explanations, as seen in Figure 1.6. In later chapters, explanations of this nature will be revisited when we try to explain and understand not only the behavior, but also the *capabilities* of complex AI systems.

Chapter 2

Which explanations are needed in AI?

Now that we have some vocabulary with which to discuss various explanations, the next step is to identify what exactly are the kind of explanations we wish to obtain. For this a few questions need to be answered:

1. What needs to be explained? Looking at a single sequence of 'steps' made by an AI system to produce a result, do we need to give a description of this sequence at a single level (horizontally) or describe some of these steps in detail (vertically), or both?
2. To what extent should these explanations be generalizable to behaviors which have not been observed?
3. In what way should these explanations be delivered? What do people consider to be a proper answer when asking a why-question?

In a pragmatic approach to explanation, these questions should be answered on the basis of which explanations people report to provide them with the most understanding. T. Miller (2019) provides an extensive overview of research on explanation in the social sciences, and concludes the explanations people prefer tend to have certain characteristics. These will be summarized below.

Furthermore, I would like to argue that in many cases, vertical explanations are necessary to be able to form horizontal explanations of the desired kind. Specifically, the objectual understanding which is missing in contemporary AI systems is understanding *at the algorithmic and representational level (ARL)*, in a large part because of the inaccessibility of their *intermediate representations*. For many 'black-box' AI systems, we generally have a very high-level description (given by e.g. the data set it should reproduce), and a very low-level description (e.g. the implementation using a neural network). However, what is missing is understanding of the 'steps' taken from input to output, about which the high-level description provides no relevant information, and the low-level description provides too much irrelevant information.

Therefore, in order to answer certain why-questions about an AI system's behavior, we need to obtain some 'general' perspective on *the type of constructs* the system uses (representations) and *the ways in which they are put together* (algorithms). I would also like to argue that establishing such a perspective essentially comes down to building a description of the system's *competence*.

In the case of traditional software development, the notion of competence is usually fleshed out *before* an implementation is developed. Generally, there is a 'specification', and the implementation must follow this specification and ideally do nothing else. However, in machine learning the implementation is developed with only a very vague specification, which means the competence theory must be 'filled in' after the fact, and this makes the relationship between the competence theory and the implementation more complex and problematic. There are a number of ways to view this relation, and these lead to different 'attitudes' toward higher-level explanation of AI systems. These varying attitudes can be observed in different approaches in contemporary explainable AI research, of which I will review some examples.

2.1 An example black-box system and its levels of explanation.

To motivate the arguments in this thesis, I will occasionally refer to an example system which can be said to be somewhat typical of the kind of systems one might come across in contemporary AI applications. The objective of this system will be **to recognize birds by the sound of their call**. To start, we may try to specify the computational theory of this system: what does the system do, and why? We would like to establish 'what' in computational terms, and justify it by using the goal of "to recognize birds" as motivation.

In any case, we can be sure that the inputs for our system will be sound samples represented in some way. However, the outputs are more abstractly specified: what does 'to recognize a bird' mean? Let us suppose for now that we know that certain types of birds exist, and that we can enumerate these. Then, the objective of the system is to pinpoint which one of these birds a certain call belongs to. In other words, the system will be a *categorical classifier*. It performance can be measured by calculating (for example) its *accuracy*, or the percentage of its classifications which is correct. However, any further specification of the computational task is difficult without knowing more about the specific properties of bird calls.

One reason why the 'machine learning' approach is so popular is because this further information about the properties of bird calls is not necessary, at least not in principle[1]. The way to build such a 'learning' program would be to choose some numerical representations for the inputs and outputs, and then construct a neural (or in general, connectionist) network of some complexity which converts an input in the specified format into a meaningful output (that is, one that has an interpretation associated with it). Then, we can use an optimization algorithm (e.g. gradient descent) to tune the network in such a way that some function which quantifies the 'correctness' of the network outputs a maximum value[2]. After this optimization process, we have an implementation of a system which performs the required task to a certain degree of accuracy.

Note that, to develop a system using this method, we need some way to evaluate its 'correctness', so there needs to be another way to obtain an answer to the question it aims

[1]In practice, most implementations *do* make use of some more problem-specific information in choosing their approach. Parts of the approach may be justifiable from the context of the problem, such as the fact that bird's calls are often very short and so the system should able to classify an input based on a positive match in only a small part of the audio sample (Grill & Schluter, 2017). However, often a certain network structure is used only because it seems to provide good performance, and not motivated by the problem specification. Some choices may even be made that seem unsuitable on theoretical grounds. In many cases (for example all submissions in Stowell et al. (2018)), a transformation is performed which represents the audio using a structure motivated by human perception of sound (Mel-frequency spectrograms). The reason why this choice is made is so that methods for image recognition can be applied to audio as well, as these methods have generally had more development and so provide better general classification performance. However, this approach is again not informed by the nature of the problem: it simply tries to transform it into a different problem altogether.

[2]Most of the time, we have a cost function which quantifies 'incorrectness' and the objective is to find a configuration for which it outputs a minimum value, but the difference between these is only a minus sign.

to solve. Usually this will be achieved by constructing a 'training set' of input/output pairs by having human 'experts' determine what is the correct output for a given input. We will assume that such a training set is available for the task of bird call recognition[3], and as such the system is constructed by *supervised learning*.

To frame a system like this in terms of Marr's levels is not so straightforward, as Marr's levels were written with 'classical' computing in mind[4]. As such, there are various possible interpretations of such a system in terms of Marr's levels. Sometimes, the idea of 'implementation' is identified with 'physical implementation', for example in T. Miller (2019) where he matter-of-factly calls the IL the *'hardware level'*, or when Zednik (2019) says the IL "is concerned with the hardware components executing the program". However, I believe that it is in a more 'context-sensitive' manner that Marr intended his levels to be applied. Consider his example of a cash register, or computationally, an addition-machine. Two examples he gives for an 'implementation' of addition are a machine adding numbers via some algorithm, or a human adding numbers on paper using the same algorithm. But despite this being the lowest level, there are still many 'layers of abstraction' present in these implementations. It does not seem that Marr is interested in going down to the absolute lowest level of abstraction (in so far as that is even possible), but rather the level at which it becomes possible for *different implementations of the same algorithm to emerge*. Then, it seems somewhat arbitrary to place this boundary squarely between the concepts of software and hardware, or something like it.

Another way to look at it is that the implementation exists somewhat autonomously from the ARL, but is also subjugated to it. The ARL specifies some instructions which both the machine and the human *must* follow in order to constitute a proper implementation of the algorithm. However, the implementations are in some sense 'free' to do so in whichever way is convenient: how the machine functions internally or what kind of materials the human uses to write down the numbers does not have any influence on them being an implementation of the same algorithm, as long as it can convincingly be said that they

[3]Such a training data set does in fact exist. See https://xeno-canto.org (Vellinga, 2020) for a large collection of (labelled) bird sounds.

[4]Connectionism only really became relevant from 1986 onwards, while Marr's work was all produced before 1981.

do follow the instructions as specified at the ARL. This is not to say that the ARL must be determined *a priori*: we can also infer algorithmic constraints from a set of existing implementations. As long as the ARL and IL are consistent with each other in this way, they make up a valid multi-level analysis.

Therefore, for a connectionist system like a neural network, I will consider the implementation to be the set of neurons and weights which specify the network, and the operations the neurons apply. This not only clearly establishes the analogy between biologically implemented neural structures and digitally implemented ones, but also encapsulates all further irrelevant details at the lowest level. For example, suppose we have such a program, and we execute it on two different computers. These computers might wildly differ in how they function internally, but that has absolutely no bearing on our analysis of the system. This is comparable to, when Marr talks about 'a human' implementation of a calculation, he abstracts over the fact that no two humans are the same. Similarly, if we take two initializations of the same network, which both follow the same ARL constraints applied by their developers, and then 'train' them both, we will (supposing that there is some randomness in the training process) end up with *two different implementations of the same ARL specification*[5]. As such, when referring to this example system or another connectionist system at the IL, I mean the specification of the neurons and weights (which is then assumed to be executed in some standardized way), by the ARL I mean any further constructs which constrain their organization (such as a layer in a multi-layer network, or a self-contained component network such as an LSTM), and by the CTL I am referring to the functional and mathematical specification which determines the task of the network and when it is performing well.

Now, when we get this bird classifying system up and running, it might perform quite well on the bird recognition task in terms of our overall judgement. However, it is difficult to say that we 'understand' the system well. All we can confidently say about the system is that (1) inputs are entered into it according to our specification, then (2) it performs a lot of calculations, and then (3) the output signifies which bird the call belongs to. The vagueness of (2) illustrates that we have very limited objectual understanding. In addition, we would

[5]For example, the network structure (i.e. number of units) is an algorithmic level constraint, but the resulting connection weights between them exist at the implementation level.

not be able to answer a question such as ‘why did the system determine this call to belong to a falcon?’, which shows our understanding-why is also very limited. The only way in which we understand this system is functionally, because we programmed it using a goal-oriented description, but even then a difference between the specific numerical goal held by the system and our intended informal goal statement can lead to unexpected results. As such, while a system like this is a simple example, it already raises many questions about our (lack of) understanding of AI systems.

Because of the way the system was developed, we only have a partial understanding of it across the three levels (see Figure 2.1). There is a high-level functional specification of the system in terms of in- and outputs, and a way to numerically evaluate whether a system satisfies this specification via a cost function. Additionally, this cost function should be motivated conceptually by relating it to the real-world problem the system is trying to solve. Together these provide the *what* and *why* of the computational theory. In addition to this computational theory, we have specified a few algorithmic concerns: a certain network layout with a specific set of parameters which can be adjusted. However, the algorithmic details of the system are flexible, in order to enable automated ‘learning’ of the right computation. Then, we have an *implementation*, or a specific set of values for the parameters and a procedure for calculating the output given a network layout, an input, and the parameters. Generally, this implementation will be very large and complex with many ‘moving parts’, but it is accessible in principle (there is no ‘hidden’ information). To summarize, there is a very high-level functional specification (in terms of in/outputs) available to us, together with a very low-level mechanistic specification (in terms of neurons and connection weights).

Now, from our existing knowledge of the system, what kind of explanations can we give? First of all, there is no good way to describe the operations the system performs in higher-level language than that of the base units, e.g. neurons and connections between them. At a higher level, we can only talk about it in functional terms. Second, while we can provide explanations on the CTL, they are unsatisfactory because we cannot ground them via vertical explanation, due to there being no clear ‘path’ from the CTL to the ARL. For example, an answer to the question “Why did the system classify this sound sample as a hawk?” may be, “Because it was most similar (in some mathematical sense) to the

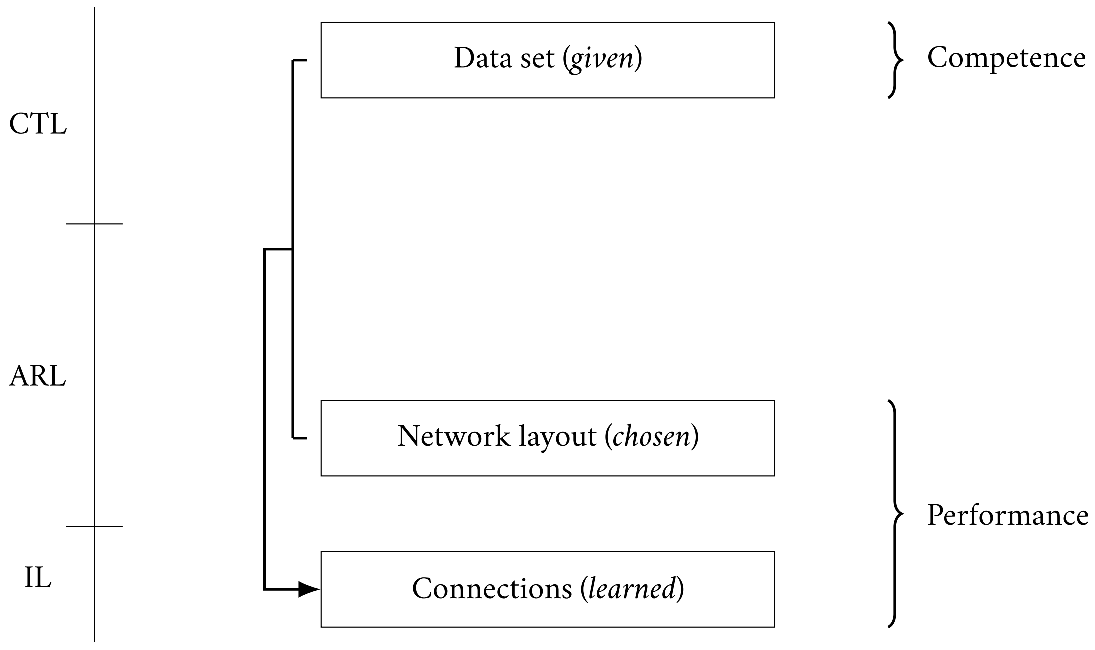

Figure 2.1: A picture which summarizes the extent of our understanding of the example bird-listening system introduced in Section 2.1. This is a supervised neural network, trained to recognize birds from sound samples. The system can be divided into three components: (1) the training data set (and an associated function to measure deviation from the data set), (2) the layout of the neural network and specification of the various parameters, (3) the values of the parameters (*connections* or *weights*). These components can be placed at the CTL, ARL and IL respectively. The block arrow on the left indicates that the connections are derived from the combination of the data set and the network layout. The competence description consists of the data set only (as it defines the problem which the system has to solve), and so the other parts constitute a description of performance.

hawk-samples in the training set." However, if one then asks "Why is it similar to those samples?" in the sense of, "what about this sample makes it similar to those samples", we have no real answer other than "crunch the numbers and you'll see".

As such, the kind of explanations we can 'easily' give are very limited. It is not possible to provide algorithmic (i.e. step-by-step) explanations which connect the implementation and the computational theory, while limiting the complexity to only include relevant details. For that reason, the field of explainable AI is essentially focused to trying to generate explanations which fill this 'gap'. In the rest of this chapter I will introduce and compare different approaches to this problem.

2.2 What kind of explanations do people desire?

In the review by T. Miller (2019), a number of observations are enumerated about the kind of explanations people prefer, and how they prefer them to be communicated. Some of these are emphasized as being particularly important, and I will shortly discuss each of them here.

1. People ask *contrastive* why-questions:

> "Research shows that people do not explain the causes for an event *per se*, but explain the cause of an event *relative* to *some other event* that did not occur; that is, an explanation is always of the form "*Why P rather than Q?*", in which *P* is the target event and *Q* is a counterfactual contrast case that did not occur, even if the *Q* is implicit in the question." (T. Miller, 2019, p. 9)

This observation indicates that, when people are searching for understanding-why, their understanding can be improved by presenting explanations which compare different events. It is related to the idea that people are mostly interested in explaining *unexpected* or *abnormal* behavior (T. Miller, 2019, p. 28). From a computational point of view, this is somewhat advantageous, because it allows the system to focus on explaining specific parts of its behavior, namely those which differ between the case of *P* and the case of *Q*. It also aligns nicely with the second observation, namely:

2. People prefer *simpler* and *more general* explanations to *complicated* and *more accurate* ones. When presented with a number of related events and different explanations for them, people seem to prefer a 'root cause'-explanation, which explains all events somewhat, to a more complicated explanation which lists the specific causes for each event (T. Miller, 2019, p. 25). The first explanation is more general, in that it predicts multiple events at once, but also simpler in that it only consists of a singular cause. However, it is less informative in the sense that the specific connections to the events observed are more vague than in the second explanation. This means that, when answering a question of the form "*Why P rather than Q?*", it is doubly important to focus on the aspects which make *P* different from *Q*: leaving in irrelevant events will degrade the usefulness of the explanation. In addition, people expect explanations to be *cohesive*, or consistent with their prior beliefs. This implies something about the consistency between different explanations as well: if two related explanations (e.g. for "*Why P rather than Q?*" and "*Why P rather than S?*") seem somehow inconsistent, they will be judged to be less reliable. As such, the generality of explanations is also important in that more general explanations can be identically applied in different situations, and so the event-specific explanations resulting from their application will be more consistent amongst themselves.

Some questions which may arise at this point are, "How can we determine what the contrast case *Q* is if it is left implicit?", or, "How can we determine which factors are relevant to this particular explainee?". The answer is that it is not necessary to determine these things *a priori*, or that it would even be better *not* to do so, because:

3. Explanation is a form of *social interaction*. Earlier, we defined explanation to be a communicative act, but in fact in practice it is not a single act, but rather a process of communication (T. Miller, 2019, pp. 32–33). The explainee may ask a why-question, and the explainer may answer with an explanation, upon which the explainee can ask another question to further improve their understanding. It is important to view explanations in this way, because it implies that fortunately, *singular explanations may be limited in their scope*. In fact, given point 2, it may be preferable to '*when in doubt, withhold information*' when explaining, and only divulge it if the explainee expresses discontentment with the less detailed explanation. This also implies that a good way to establish a contrast case *Q* would be to simply ask the explainee to provide one.

Note that, while the idea of 'explanation as communication' might seem a bit of stretch for common AI systems such as neural networks, 'communication' does not imply a use of language. In fact, these kind of interactive explanations are more common than one might think. Consider a diagram which presents a schematic view of some data set, split into categories. Now suppose that there are too many categories to show at once, so similar categories are grouped together to provide a global overview. Additionally, when the user selects one of these groups because they are interested in it, the categories contained in this group are displayed in full. In some sense, this is an interactive 'explanation': it gives the user insight into an otherwise inaccessible source of information by incrementally providing them with more details about the aspects in which the user expresses interest.

Now, what do these observations mean for explanations in AI in terms of the concepts treated in Chapter 1? First of all, it shows that there is little need for 'complete' explanations of the system, as people are generally focused on having explained to them what is happening in specific situations. As such, we can focus on generating horizontal explanations for certain behaviors under study. However, these horizontal explanations should not be at a too low level, for they then lose the desired properties of simplicity and generality. In Marr's terms, it seems that the implementation level may be too low, while the computational theory level may be too high: it is mostly focused on a functional understanding of the system, and idealizes the system to a point where it would be hard to discuss the differences between specific behaviors. As such, it seems the horizontal explanations we are looking for should be located around the ARL. However, the requirement of coherent explanations makes the CTL relevant as well, as it provides the overarching theory which connects different explanations at the ARL and so can ensure their consistency. From the perspective of the competence/performance distinction, the desire that explanations are simple and general indicates that it is important to explain the system in terms of its competence, rather than the 'messy' details of its performance.

2.3 How would we like to understand AI systems?

Based on the observation that the kind of understanding people are generally looking for is an understanding of specific events, it seems reasonable to suggest that the efforts of explainable AI should focus on answering questions about these events. However, much research in explainable AI seems to focus on 'model-based' explanations, of the kind scientists apply, rather than the kind of explanations which are applied in 'ordinary' situations. This is observed by Mittelstadt et al. (2019):

> "[T]he vast majority of work in [explainable AI] produces simplified approximations of complex decision-making functions. We argue that these approximations function more like scientific models than the types of scientific and 'everyday' explanations considered in philosophy, cognitive science, and psychology." (Mittelstadt et al., 2019, p. 1)

They argue that the focus of explainable AI should be on question-answering, rather than modelling behavior. In other words, while the scientist is generally concerned with objectual understanding, the majority of "stakeholders" in the application of AI are concerned with understanding-why. Moreover, the two are incompatible in the sense that answering a 'understanding-why'-question with a behavioral model can be misleading:

> "It is not enough to simply offer a human interpretable model as an explanation. For an individual to be able to trust such a model as an approximation, they must know over which domain a model is reliable and accurate, where it breaks down, and where its behaviour is uncertain. If a recipient of a local approximation does not understand its limitations, at best it is not comprehensible, and at worst misleading." (Mittelstadt et al., 2019, p. 3)

As such, they explicitly point out the gap between understanding-why and objectual understanding. Partial objectual understanding of a complex model might lead an observer to try to answer why-questions themselves, but the result may be incompatible with the actual system. Partially, this is a question of how to effectively communicate these limits of approximations, but it also raises an interesting question about the relationship between the different types of understanding.

An opposing viewpoint is motivated by Páez (2019). He considers understanding-why to be "a localized variety" of objectual understanding (Páez, 2019, p. 454), which means that the construction of local models is in fact the best way to answer why-questions about specific events. Relatedly, he argues that "the connection between explanation and understanding in AI is not comparable to that same connection in the social and natural sciences", and motivates this by the claim that "[the] use of arbitrary black-box functions to make decisions in machine learning makes it impossible to reach the causal knowledge necessary to provide a true causal explanation". But what exactly makes a machine learning model different from a natural complex system such as a fluid is not so clear. In principle, there should be a (possibly very extensive) causal chain which can be followed for every decision[6]. Therefore, his argument does not only equate understanding-why and objectual understanding in some sense, it also *devalues* them in that he supposes that understanding of either kind is essentially limited. This seems like a point which can be investigated further, to make clear *why* exactly understanding of black-box AI systems should be so limited.

In some sense, I agree with both of them, in that the two types of understanding are related but not equivalent. Why is that? An answer may be formulated in terms of Marr's levels of analysis (or another multi-level account), because while we have full knowledge about the system at *some* level, we have almost no understanding at other levels. I would argue that understanding-why is the end goal of explainable AI, and this understanding can be provided by horizontal explanation at an appropriate level. However, that does not mean objectual understanding of the system is unimportant, as we must be able to *ground* these horizontal explanations vertically, by connecting them to lower levels. For that reason, it is necessary to look into the relationships between the various levels of complex AI systems.

An example: the feature-possession model. Let us consider an example to further illustrate where the 'gap' in our understanding of AI systems lies. In T. Miller (2019), an example is presented of an 'idealized' explanation of a classification system, so let us assume that this is the kind of explanation we are hoping to achieve in the end. This example has been reproduced in Figure 2.2. It shows a fictional conversation between an artificial agent (the explainer) and a human observer (the explainee), and illustrates how one might

[6]At one point, Paéz mentions that "Many types of black box models, like deep neural networks, are stochastic (non-deterministic)" (Páez, 2019, p. 445). However, this seems to be a misunderstanding. While neural network models have some randomness in their learning process (for example, their initial weights), given any specific instantiation of a network and a decision made by it, the network is (usually) fully deterministic, and we *can* indeed retrace the computations it has made completely. Furthermore, even if there were actual randomness in the decision-making, this will still generate events with causal connections. Imagine a person following a list of instructions, of which one is "roll a dice, if it comes up 3 or less make coffee, otherwise make tea". If we then ask the person why they made coffee, "I followed the instructions and the dice came up 2", that seems like a valid causal explanation.

Type	No. legs	Stinger	No. eyes	Compound eyes	Wings
Spider	8	✗	8	✗	0
Beetle	6	✗	2	✓	2
Bee	6	✓	5	✓	4
Fly	6	✗	5	✓	2

(a) A simple lay model for distinguishing common arthropods.

Person:	"Why is image J labelled as a Spider instead of a Beetle?"
ExplAgent:	"Because the arthropod in image J has eight legs, consistent with those in the category Spider, while those in Beetle have six legs."
Person:	"Why did you infer that the arthropod in image J had eight legs instead of six?"
ExplAgent:	"I counted the eight legs that I found, as I have just highlighted on the image now." (ExplAgent shows the image with the eight legs counted).
Person:	"How do you know that spiders have eight legs?"
ExplAgent:	"Because in the training set I was trained on, almost all animals with eight legs were labelled as Spider."
Person:	"But an octopus can have eight legs too. Why did you not classify image J as an octopus?"
ExplAgent:	"Because my function is only to classify arthropods."

(b) Example Explanation Dialogue between a Person and an Explanation Agent.

Figure 2.2: An example classification model and an interactive explanation of an agent's decision. Table (a) contains the features the agent uses to make its decisions, while (b) contains a possible conversation between the agent and an interested third party. Reproduced from T. Miller (2019, p. 4).

imagine contrastive, social AI explanation taking place. The system uses observational data of arthropods to classify an arthropod as one of four 'types'. Presumably trained on a large data set, it deduces a number of *features* which determine the type of anthropod it is dealing with. The types (outputs) and their features are listed in the accompanying table. This kind of model is commonly found as a way of reasoning about AI systems[7], and I will refer to it as a **feature-possession model**. This is a model in which an input is represented in terms of a number of **features** (a variable which has a specific value for the given input, e.g. 'it has three eyes' or '*eyes* := 3') and where a decision is made based on which features the input 'possesses' (e.g. 'if it has 8 legs, it is a spider' or 'if *legs* = 8 then *spider*').

The example in Figure 2.2 is suggestive in that, given this table of types and their features, one can intuitively see how it should be possible to give answers to why-questions in a way similar to those presented in the conversational example. However, what makes explanation of connectionist systems difficult is *obtaining* such a table in the first place. Think back to our bird example: if we have a table which lists for each type of bird a numbers of features of its call, it seems like we could reasonably answer questions about the classifications the system makes. But, the only information we have is a number of sound samples, and an associated 'type of bird' the system has assigned to it[8]. We know nothing about either (1) the kind of features these bird calls have according to the system, (2) the kind of operations the system applies to these features to produce a result, or even (3) whether there are any delineable 'features' present in the system at all. Another way to phrase this is that a feature-possession model should clearly be situated at the ARL: it defines constructs (i.e. representations) and how to combine them (i.e. via an algorithm). However, as we have seen before, for a typical black-box model we only have a 'computational theory' and an implementation to guide us.

This is the central challenge in explaining connectionist systems: we have no knowledge of their internal *representations*, and as such also no knowledge of the operations applied

[7]As an example of this terminology, Blum and Langley (1997) named the "selection of relevant features" as one of the central problems in machine learning, and describes the way in which a machine learning program can learn about *concepts* in terms of features: "[Concept learning is] deciding which features to use in describing the concept and deciding how to combine those features". It is also prevalent in contemporary approaches to explainable AI, as will be shown in Section 2.6.

[8]Or more likely a probability distribution over bird types, but this is not relevant to the argument.

to them (their *algorithms*), at least not above the level of the fundamental computational units (e.g. single neurons). In Marr's terminology, our understanding of the ARL is severely lacking. As argued in the previous section, ARL explanations are generally the kind of explanations which aid people the most in understanding these systems. Therefore, this is what makes connectionist systems difficult to explain: in addition to generating then explanations themselves, it is already a challenge to build an ARL model of the system (such as the table in Figure 2.2a) which can be used to generate these explanations.

It should be clear that without any understanding of the system at the ARL, it is not possible to answer questions in the way presented in the example dialogue, *even in principle*. To produce an answer along the lines of, "because it has eight legs", the system must first be able to recognize the concept of *having legs*, and we need to be able to identify it as recognizing such a concept. Therefore, the challenge of explainable AI is as much about *discovering ways to articulate the system's representations and algorithms* as it is about finding ways to present these in the form of explanations which people can easily parse and learn from.

This issue raises a number of questions. Is it possible to give an explanation on the ARL of such a black-box system, and if so, what does that explanation actually say about the system? In the next section, I will look at a number of ways to view these connections between the different levels of connectionist systems. After that I will look at a number of existing methods in explainable AI, each of which aims to articulate the representations and algorithms used by connectionist systems for the purpose of providing explanations, and show that they relate the various levels of such systems to each other in different ways.

2.4 The competence and performance of connectionist systems.

In traditional, 'classical' computing, there is a clear relationship between the different levels of Marr's hierarchy. At the CTL, it is specified what the program computes, in (semi-)mathematical language. Another way to look at it would be that it specifies the *constraints* the program has to satisfy. Normally, one would then try to derive an algorithm which gives a procedure that is guaranteed to satisfy the constraints of the CT, in effect specifying the ARL. As such, the ARL is tightly linked to the CTL in that it is designed to satisfy the

CT's constraints. Then, an implementation can be developed using whatever computing machinery is available. Here again at the IL, the program is designed to fit the specification of the ARL, and so the CTL as well. Due to the 'downward' motion of development here (CTL → ARL → IL), this has sometimes been called the *classical cascade* (e.g. by Clark (1990), after Dennett (1987, p. 227)). This is a natural approach from an engineering perspective: one often builds a list of 'requirements' for a system and then constructs an implementation which satisfies those requirements.

However, in connectionist networks (which are roughly identifiable with 'black-box machine learning systems'), this is not the way in which programs are developed. Rather, as seen in the bird example, one starts with a functional description, which contains a mathematical specification (e.g. a utility function) but is otherwise quite general. Then, a few constraints of the implementation are chosen (such as, the type of network and the number of neurons, etc.), and it is 'trained' to accomplish the goal specified at the CTL. Now, the question is, because there is barely any *a priori* description at the ARL, can we even describe the system at this level, and if so, how do the descriptions at the various levels relate to each other?

To construct an ARL description of such a system, it seems that it is necessary to approach it from either the CTL or the IL. The CTL is focused on specifying *what* task is to be performed while both the IL and ARL are concerned with *how* it is accomplished. As such, it seems that there should be more to learn from the IL than from the CTL. Andy Clark (1990) calls this 'inversion of movement' from the lower levels towards the higher levels "Marr-through-the-looking-glass":

> "Under Marr's influence, Cognitive Scientists are likely to expect some high level understanding of a task to *precede* and *inform* the writing of algorithms. Classical competence theoretic specifications aim to do just that job. The connectionist, however, effectively inverts this strategy. She begins with a minimal understanding of the task, trains a network to perform it, and *then* seeks, in various principled ways, to achieve a higher-level understanding of what it is doing and why." (Clark, 1990, p. 220)

Here Clark talks about 'competence theories', which have been discussed in chapter 1. According to Clark, they are "meant to be suggestive of the processing structure of a class of mechanisms of which we are a member" (Clark, 1990, p. 200), and as such he places them on a similar level (between the CTL and the ARL) as I do (shared across the CTL and ARL). Additionally, Clark places the functional specification of the network (e.g. the objective function) at an even higher level than the computational theory level ("level 1"): he frames it as more of a "level .5 task analysis" (Clark, 1990, p. 214). I will instead just call such a specification a computational theory, but indeed a very loosely specified one (i.e. one which contains very few constraints). Then, as far as I understand it, his "task analysis" is the same as what I place at the CTL, and his idea of a competence theory is mostly concerned with what I have called the 'blueprint' of the ARL[9]. So his idea of a competence theory roughly aligns with the 'competence description' as I have introduced it in chapter 1.

Clark presents two ways in which the connection can be made between competence and performance in a connectionist system. First is what he calls the 'Newtonian competence' view, in comparison to the relationship between Newtonian (classical) physics and quantum physics (Clark, 1990, p. 204). In this view, any higher-level description than the IL may be *descriptive* of the system's performance, but not in any way *suggestive* of the actual processing involved (Clark, 1990, p. 208). In terms of explanation, this means that a higher-level explanation should describe the behavior of the system well, possibly in terms of higher-level intermediate representations and operations similar to a classical program, but it does not say anything about any intermediate representations or such *actually existing* in the system.

Depending on the demands we place on AI explanations, this may be sufficient, in which case it nicely separates the problem of high-level explanation from the problem of extracting information about representations used in the system. This idea of explanation in connectionist systems leads to the approach of *approximating models*, some examples of which are discussed in the next section. However, for some questions it may also seem insufficient to only have a description which fits, but not necessarily corresponds to what the system is actually doing. Is it sufficient for a system to only 'look like' it is not being unlawfully discriminatory in its decisions, or should it somehow *actually* not be?

[9]Additionally, he places what I call the 'implementation' on the algorithmic level: "the 'algorithmic' specification, for a connectionist, must be a specification of (a) the network configuration and (b) the unit rules and the connection strengths" (Clark, 1990, p. 216).

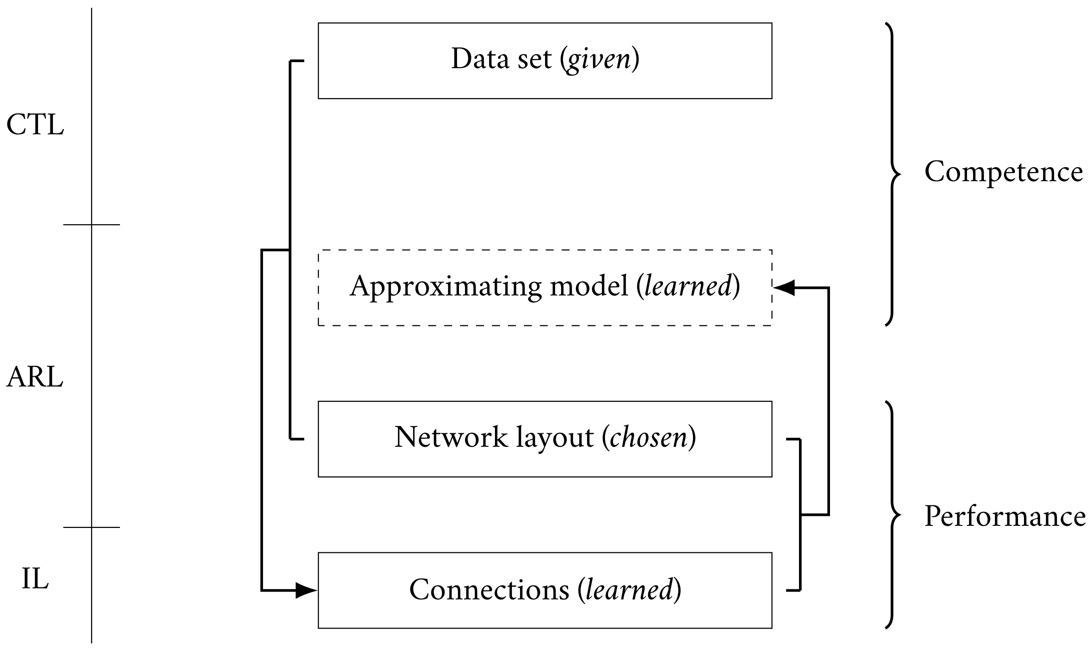

Figure 2.3: A schematic depiction of Clark's (1990) view of *Newtonian (descriptive)* competence, using the example system from Figure 2.1. In this view, the competence description may be descriptive of the system's performance, in the sense that it produces (approximately) the same outputs for a representative set of inputs. To explain the system's behavior, the competence description is extended with a 'simpler' system (dotted box) which approximates the black-box system's performance in this way.

The second view of explanation which Clark presents he calls 'rogue competence' (Clark, 1990, p. 208). The idea of this view is that the connectionist system is *not actually an instance of what is described by the competence description*, but it is simply a heuristic tool which gives answers that agree with it most of the time. In some sense the connectionist system is the approximation here. For example, the bird network as specified above cannot be said to actually recognize birds by their calls, it can only approximately 'predict' what a proper recognizer would answer. As such, the only way in which this network can be linked to the idea of 'recognizing birds' is by some *external resource* which evaluates it as doing so (Clark, 1990, p. 209). This external resource would be another system which *does* confirm to the competence description in the traditional sense, in that its processing is reflected in it.

What is interesting here is that the 'external resource' may be part of same agent in some way. Clark gives an example of language learning:

> "An obvious and related advantage of the rogue approach concerns the psychological plausibility of so-called supervised learning algorithms. These are procedures for training connectionist networks which rely on the back propagation of error messages, and hence rely on a *teacher* [...] which looks at the system's output and tells it what the output *should* have been like. [...] Such set-ups have often appeared deeply psychologically unrealistic. For example, when we learn a language, we can do so by being given *positive* examples only (as Chomskians are fond of pointing out). Whence, then, the teacher and the error messages?
>
> The possibility which rogue models open up is that a separate system stores a set of input-output pairings (*e.g.* a set of observed print-phoneme pairings) and uses these to train a connectionist network. The negative instances are thus generated and spotted by the brain itself, rather than by other agents." (Clark, 1990, p. 210)

So, what this idea of competence postulates is that there is an *actual* competent system somewhere out there, which interacts in some way with the connectionist system. For example, in the scenario of the bird network, this would be the learning algorithm and the data set it uses to 'train' the system. These are competent resources, because they constitute a successful realization of what we consider to be a competent demonstration of

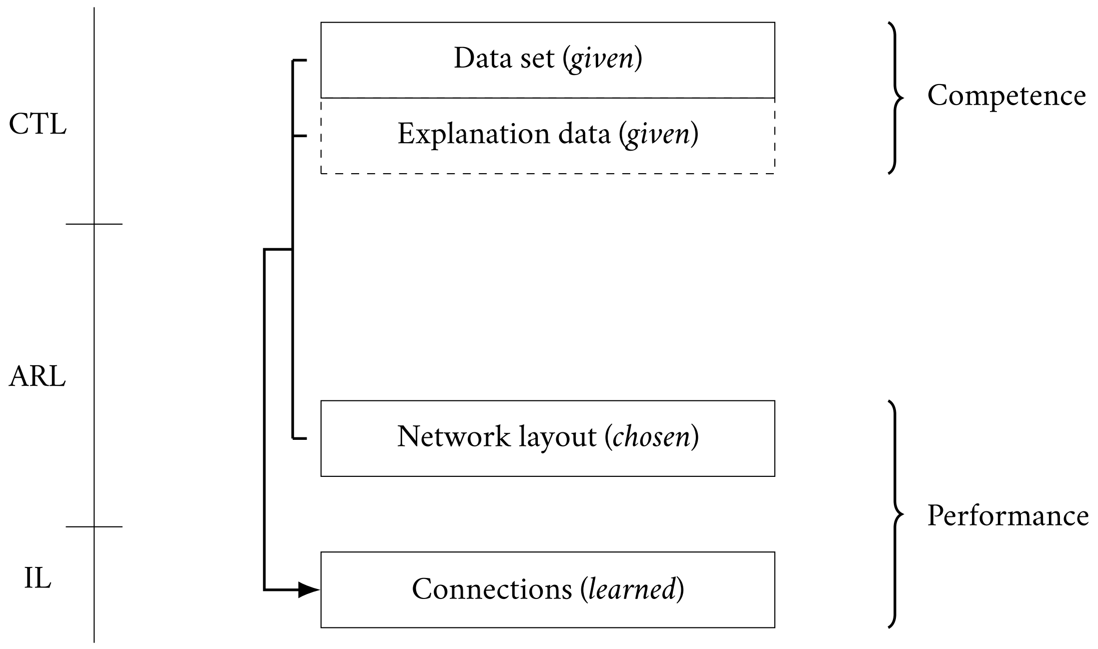

Figure 2.4: A schematic depiction of Clark's (1990) view of *rogue (top-down)* competence, using the example system from Figure 2.1. Here, the competence description is entirely separate from the performing system, and so any notion of competence in the system can only be enforced 'from the top down'. Concretely, this means that for a (supervised) learning system, we train it to generate explanations in addition to its usual outputs. The competence description is therefore extended with a data set containing example explanations (dotted box) from which the system can learn.

'recognizing birds'. By assumption, the training data is right, and therefore competent with respect to the task at hand. However, this idea of competence extends further in that the competent resource may be more complex than just supplying samples from a data set. The example Clark mentions in the quote above, where the brain 'trains' its linguistic faculties by generating examples of language itself, is somewhat similar to contemporary *generative adversarial networks (GANs)*, in which two systems 'challenge' each other repeatedly to improve their performance[10].

This 'rogue competence' view of connectionism implies that a full description on all levels cannot be given for a system such as the bird network without taking into account the external resources it interacts with. As such, when we try to give a description across all levels, we are actually describing multiple interacting systems instead. Additionally, this means that a competence description again says nothing about the connectionist implementation itself, but is rather just an analysis of the competent resources. For example, we may be able to extract some set of features from the data set of bird calls, but there is no telling whether the connectionist implementation also makes use of these features. The way to provide further understanding about the implementation would therefore be to *extend the interaction* between the connectionist implementation and these competent resources somehow. In the next section we will see some examples in explainable AI that in some sense aim to do just that, for example by including the constraint of 'providing explanations' in the learning process.

In both of these scenarios the classical idea of a 'competence theory', where the performance is an *instance* of the competence being demonstrated, is disconnected from the connectionist implementation, and therefore Clark concludes that "the connectionist

[10]An extensive introduction can be found in Goodfellow (2016). It characterizes GANs as follows: "The basic idea of GANs is to set up a game between two players. One of them is called the **generator**. [...] The other player is the **discriminator**. [...] The discriminator learns using traditional supervised learning techniques, dividing inputs into two classes (real or fake). The generator is trained to fool the discriminator. We can think of the generator as being like a counterfeiter, trying to make fake money, and the discriminator as being like police, trying to allow legitimate money and catch counterfeit money. To succeed in this game, the counterfeiter must learn to make money that is indistinguishable from genuine money, and the generator network must learn to create samples that are drawn from the same distribution as the training data." (Goodfellow, 2016, pp. 17–18). Here the discriminator plays the role of the 'competent resource' (by definition) and the generator the role of the 'rogue system' who tries to live up to the standard of competence set by the discriminator.

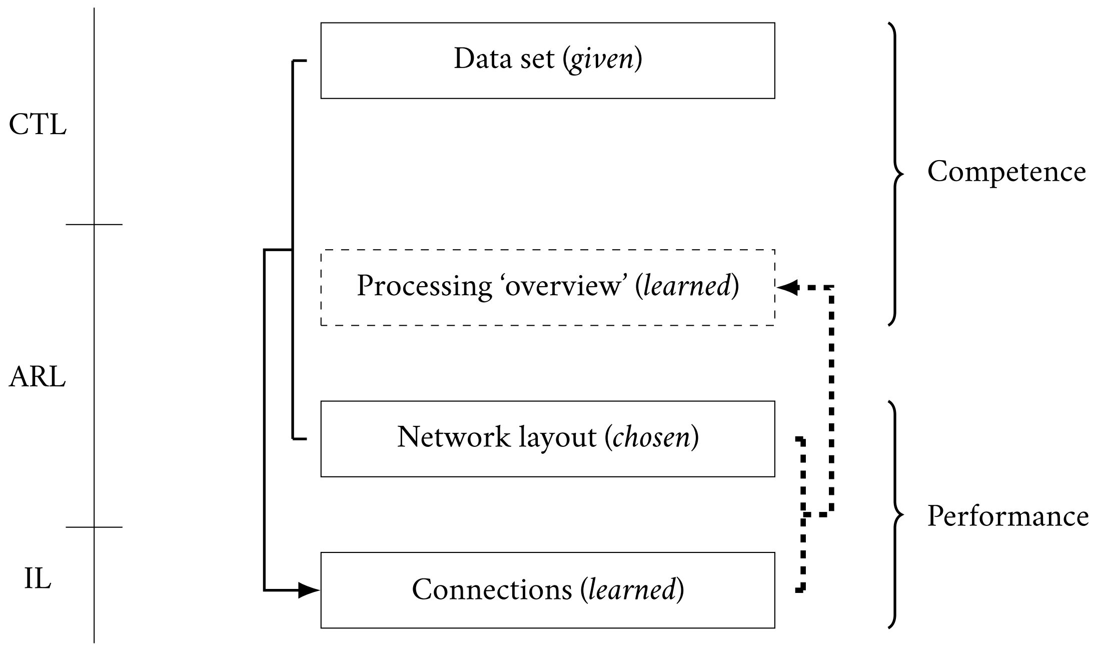

Figure 2.5: A schematic depiction of Clark's (1990) view of *bottom-up* derivation of a competence description, using the example system from Figure 2.1. This looks similar to the descriptive view in Figure 2.3, with the difference lying in the nature of the relationship between the performance description and the 'processing overview' (dashed line). This should be somehow more in-depth than in the descriptive case, in that it also considers the internal structure of the system. A way in which this relationship may be construed is discussed in Section 2.5.

must distance herself from the details of the classical competence model" (Clark, 1990, p. 220). Instead, he advocates in favor of techniques which analyze the implementation to extract some kind of 'overview' of the processing it performs[11]. To him this kind of *bottom-up* analysis, where the implementation is investigated in order to construct a higher level description is different from the 'classical' idea of competence, because in the classical picture the competence theory determines what is a valid performance, not the other way around.

To summarize, according to Clark there are three different ways to view the relationship

[11]He particularly focuses on *cluster analysis*, a technique similar to modern day *representational similarity analysis (RSA)* (Kriegeskorte, 2008), which is used to perform experiments concerning the ways in which the human brain represents information.

between a competence description and a performance description of a given system:

- The *top-down* (rogue) approach (Figure 2.4): the competence description specifies constraints which the performance description, and so the implementation, should satisfy. However, other than that the implementation is free to be constructed and to function in any way which is convenient. The main purpose of the implementation is to be a useful 'tool' in order to generate performances, which may not be entirely accurate to the competence specification.
- The *descriptive* (Newtonian) approach (Figure 2.3): the competence and performance descriptions *agree with* each other, but neither has 'authority' over the other. A competence description can be said to *model* the system, even in a rigorous (e.g. falsifiable) manner, but its account of the system's internal state cannot be said to be any more 'correct' than another description which is equally effective in modelling the system's behavior.
- The *bottom-up* approach (Figure 2.5): the competence description is *derived* from the implementation. This makes the performance description the primary account.

Clark calls the first two approaches 'classical' conceptions of competence, while the third is something new, something which is necessary in the paradigm of connectionism (and therefore explainable AI). Still, there are a few questions which are left unanswered:

1. Even in the traditional picture of competence and performance, there is no 'fixed' direction of inference in which theories should be developed. It seems to me that, in any case where a competence has been formulated, this has always been preceded by investigation of 'messy' performance. According to Chomsky, "the problem for the linguist [...] is to determine from the data of performance the underlying system of rules that has been mastered by the speaker-hearer and that [they put] to use in actual performance." (Chomsky, 1965, p. 4), and "[c]learly, the actual data of linguistic performance will provide much evidence for determining the correctness of hypotheses about underlying linguistic structure" (Chomsky, 1965, p. 18). How then, is it possible to distinguish between top-down and bottom-up approaches to competence?

2. How exactly is the 'bottom-up' construction of a higher-level description different from the 'Newtonian' view, where competence is simply an account of the system which is *approximately* valid, in the sense that it gives the same results? Can a bottom-up competence description in any way be *more* than 'merely' descriptive of the actual system?

The first point casts doubt on the division of competence theories as either 'classical' or not. Even the 'original' competence theories by Chomsky are partially derived from actual performance. While not explicated by Clark, a difference between classical and non-classical higher-level theories seems to lie in their attitude towards to *idealization*. In the classical view, we can derive some account of an 'ideal performance' according to the competence theory, while in the bottom-up view we cannot: here the implementation is primary. Another difference is somewhat metaphysical: in the first case the specification of a competence, though abstract, is still treated as 'real' and existing separately from the implementation (or even as physically real, as with Chomsky's generative grammar), while in the second case it is treated as only a different kind of description of the actual implemented system.

The second question is the topic of the following section. Here we will look at a way in which a bottom-up account can be said to be 'more' than only a descriptive model of the system's behavior. The general idea behind this is that there needs to be some kind of *structural correspondence* between the lower- and higher-level descriptions, which allows us to make stronger claims about the processing the system performs.

2.5 The bottom-up approach: tacit knowledge in connectionist systems.

We have seen that for connectionist systems, what is generally missing is the 'algorithmic' part of competence, and that this can be built in two ways: either top-down, by adding to the competence specification of the system, or bottom-up, by extracting algorithmic information about the connectionist implementation. In Clark's three approaches to describing the competence and performance of connectionist systems, he discusses a top-down 'rogue competence' approach, which places competence as external to the connectionist system, and a bottom-up 'Newtonian' approach, which places competence as simply approximating the performance of the connectionist system. However, he also recommends one 'distances themselves' from these conceptions of competence, and advocates for another bottom-up approach, which depends on investigation of the algorithmic structure in the connectionist implementation. This notion is a bit difficult to clearly grasp: for example, how is this different from the Newtonian 'approximative' approach? One answer is that the Newtonian approach is mostly concerned with functional approximation (i.e. reproducing in-output pairs) while the 'new' approach is more concerned with *structural* approximation (i.e. uncovering the mechanisms which produce these in-output pairs). In this section I would like to discuss this difference a bit more thoroughly.

An important notion to describe the kind of relationship between levels we are dealing with here is **tacit knowledge**. This is a notion which often comes up in the context of competence, as it is essential to Chomsky's account of language. The motivation behind it is that, because the language people use displays some regularities, and people can make use of these regularities to make judgements whether a sentence is 'correct' or not, there must be some underlying internal representation of these regularities or **rules** which the speaker possesses. However, when a speaker is asked about these 'rules', they will generally not be aware that they know of any, nor be able to state them explicitly. This is why Chomsky says that they *tacitly* possess knowledge of the rules.

But what exactly is the difference between (1) knowing a rule, (2) tacitly knowing a rule, and (3) not knowing a rule but simply 'happening' to perform as the rule would specify? This is a relevant question when discussing connectionist systems. For example, given a feature-possession model, if we would like to say the system 'outputs *O* because input *I* has feature *F*', we clearly cannot claim this in the sense of (1), as the rule linking *F* to *O* is not explicitly present in the implementation. However, if we simply mean this in the sense of (3), we are back to the 'Newtonian' conception, where the rule simply 'fits' the behavior of the system. Therefore, interpreting this in the sense of (2) would be a more convincing option, but when exactly can we reasonably do so?

Moravcsik (1974) tries to isolate the context in which tacit knowledge appears by distinguishing different kinds of intelligent behavior. On the one hand, we have what he calls *strategy*, where one tries to achieve a certain goal, and applies *corrections* to their behavior in order to achieve this goal (Moravcsik, 1974, p. 302). Learning connectionist systems certainly fit this description, as they adjust their connections in order to produce the desired behavior. On the other hand, we have *conscious rule-following*, where one explicitly follows a rule in order to determine their behavior. In between these two lies the notion of 'tacit rule-following'. Given an agent *A* and a rule *R*, Moravcsik distinguishes these types of behavior using a number of criteria (Moravcsik, 1974, p. 303):

(i) The rule *R* fits the relevant aspects of *A*'s conduct.

(ii) The rule *R* provides a reliable basis for predicting *A*'s future conduct.

(iii) *A* must have beliefs with regard to what does and what does not constitute a violation of *R*.

(iv) When presented with *R* they must recognize it as the rule they are following.

(v) *A* must have formulated *R* consciously[12] some time before their application of it and this event of formulation is causally related both to the events of application and to *A*'s satisfying (iii).

When all criteria are satisfied, we can clearly speak of *A* following rule *R*, and we are in scenario (1). When only (i) and (ii) are satisfied, *A* is only *acting in accordance* with *R*. As such we are in scenario (3), where the rule is simply a 'Newtonian' approximation of the behavior. However, in the case where (i-iii) hold but not (iv) and (v), we are somewhere in between: the rule seems to be present, but *A* cannot articulate it. This is where Moravcsik places 'tacit rule-following', of which linguistic competence is one example (Moravcsik, 1974, p. 306). This means that the important criterion which determines whether tacit knowledge of rules is present is (iii): *A* must have beliefs about rule violations.

Of course, this notion of "having beliefs" about rule violations is difficult to apply to an artificial computational system. For that reason, let us look at another conception of tacit knowledge, by Martin Davies:

[12] Moravcsik is only talking about humans here, so in our case we should replace 'consciously' by something like 'explicitly'.

"[F]or a speaker to have tacit knowledge of a particular articulated theory, there must be a causal-explanatory structure in the speaker which mirrors the derivational structure in the theory." (Davies, 1989, p. 544)

There is quite a bit to unpack here. First of all, his usage of 'theory' implies a logical formalism, consisting of axioms and rules, which one can use to derive propositions which are true. This derivation is a process of asserting axioms, applying rules to them, and formulating conclusions. The structure of this process must then be *mirrored* in the speaker, which means there must be some correspondence between the different states the speaker goes through (i.e. the states as used in a horizontal explanation) and the states in the derivational process. To put this definition in more familiar terms, one could substitute 'implementation' for 'speaker', 'ARL description' for 'theory' and 'algorithmic structure' for 'derivational structure'. This leads us to the following definition:

For **tacit knowledge** of the components of a certain *ARL description* to be present in a certain system, there must be a *horizontal explanation* (i.e. sequence of states) at the *implementation level* which mirrors (i.e. can be connected via vertical explanation to) the *algorithmic structure* of the *ARL description*.

As an example, consider the feature-possession (FP) model discussed before. Using such a model, we will be able to deliver explanations of the form "the output is O because I has feature(s) $F_1, ..., F_n$" for some $n \in \mathbb{N}$. Such an explanation amounts to a horizontal explanation at the ARL, consisting of $3 + n$ consecutive states:

(1) The input is I.
(2, ..., $n + 1$) I has feature $(F_1, ..., F_n)$.
($n + 2$) Features $F_1, ..., F_n$ are characteristic of output O.
($n + 3$) The output is O.

This horizontal description describes a rough algorithmic structure by which the system decides on output O given I. To strengthen this 'hypothesized' description, we would like to ground it *vertically* in the implementation. In other words, we would like to say that the implementation *tacitly* (or implicitly) makes use of the features $F_1, ..., F_n$ in the input to determine the output. Using the definition above, the way to do this would be to show some **correspondence** between the 'internal states' (2, ..., $n + 2$) which appear

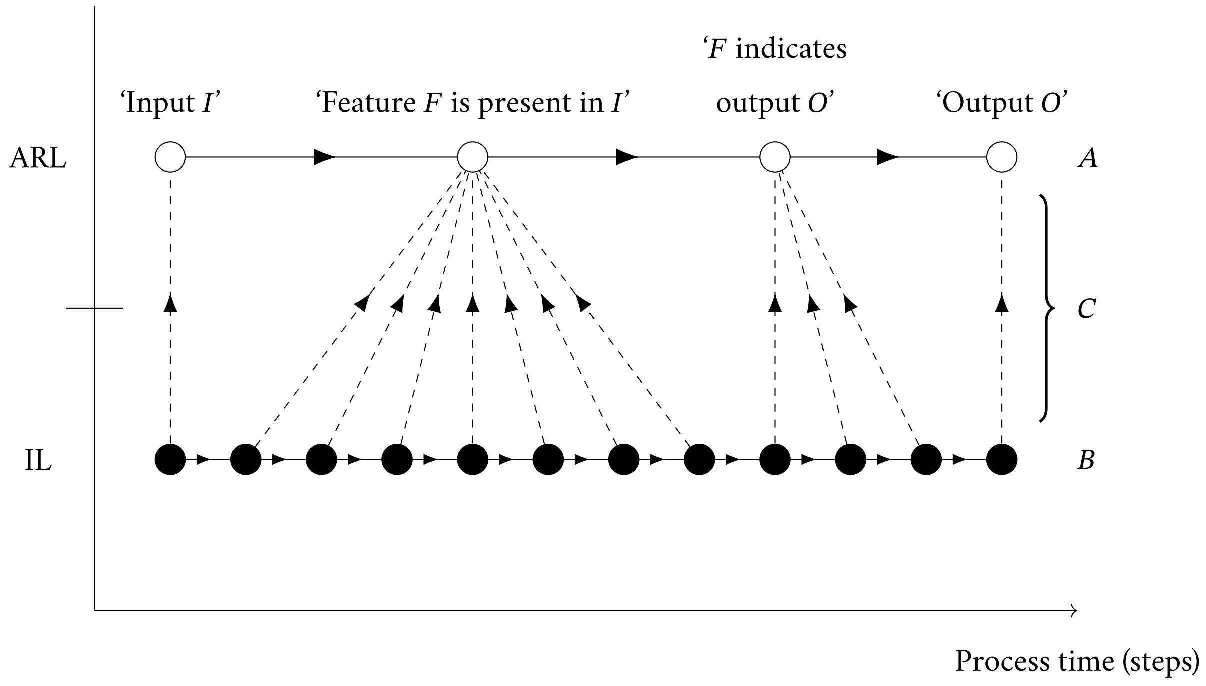

Figure 2.6: An example correspondence between an IL and ARL description of a single behavior, where the input *I* is classified as *O* based on a single feature *F*. The horizontal explanations *A* and *B* consist of sequences of system states at their respective levels. The horizontal axis shows that these states follow each other sequentially in the system's processing. The correspondence *C* between them is given by a function mapping states from *B* onto states from *A*. This correspondence obeys the requirements specified in the text: each state in *B* maps onto one in *A*, and the order of any two states is preserved after mapping.

in this description, and one or more states which appear in a horizontal description at the implementation level. An example of a correspondence with $n = 1$ is visualized in Figure 2.6. For such a correspondence to hold, we need to have that each possible (set of) state(s) in the implementation clearly maps to the presence of a *specific* (set of) feature(s) F_i (or of course the input/output states $(1, n+3)$). We cannot have a mapping which is context-sensitive (i.e. where the states used to identify a certain feature are dependent in their identification on other states) or otherwise undetermined (where some states in the implementation do not map deterministically onto a feature), or else we cannot say that the implementation (tacitly) makes use of the feature as presented, as then the (deterministic) causal structure implied by the reasoning 'input → features → output' is not preserved.

To formalize this notion, we can say that given a sequence of ARL states $A = (a_1, ..., a_n)$ and a sequence of IL states $B = (b_1, ..., b_m)$, there needs to be a correspondence $C : B \rightarrow A$ which maps IL states onto ARL states. To preserve the causal structure, we must also have that if $C(b_i) = a_j$, $C(b_k) = a_l$ and $i \leq k$, then $j \leq l$. This ensures that the order of states remains the same when mapping to the ARL (i.e. arrows cannot cross each other when displayed as in Figure 2.6). In other words, each IL state must 'belong' (be mappable onto) a certain ARL state (or C is not a function), and we need to be able to order the IL sequence so that all states belonging to a certain ARL state are 'grouped together', meaning that we can find a subsequence of B containing all of and only those states[13].

This definition of tacit knowledge can be related back to the "intuitions about rule violations" definition by Moravcsik. In this context, we would like to interpret his definition as the implementation having 'intuitions about algorithmic-level states and operations'. But because we obviously cannot *ask* the system about its 'intuitions', the best we can do is probe its state. If we have a clear mapping from the IL to the ARL, we can probe the implementation's state, translate this into the associated ARL state, and

[13]A weak point of this formalism is that a single IL state must be decisively mapped onto a specific feature. However, it might be the case that multiple states need to be taken together to distinguish between features. For example, if we use neuron A and neuron B *both* firing to determine the presence of feature F, we cannot map "A is firing" onto "F is present". The way to circumvent this is to adjust the IL description to consider "A and B are firing" as a single state instead. Nonetheless, there is most likely a better way to formalize it that incorporates this complexity into the definition of the correspondence instead of the IL description, but this one is sufficient to illustrate the idea.

from that deduce whether or not the system considers a certain 'rule' (in the sense of a certain algorithmic operation) is being followed (performed). In other words, if we have a correspondence there will be a set of states $\mathbf{b} \subseteq B$ the observation of which can tell us whether a certain feature (state) $a \in A$ is present. If we do not have such a mapping, we cannot necessarily identify the presence of a, in which case we cannot 'elicit' any response of the system about a 'rule' which exists at the algorithmic level of description, and so cannot claim that the system 'responds' to the rule or concept represented by a.

As such, we need two things to speak of an implementation making tacit use of an algorithmic concept: (1) we need to be able to *localize* this concept in the implementation and (2) when this concept is present, it must have a *consistent* manifestation in the implementation. It is often argued[14] that this is a central challenge for connectionist networks, as it is very unlikely that any *a priori* determined semantic concept will have such a localized and consistent representation. But this, as I understand it, is exactly the reason why Clark (1990) advocates for a bottom-up approach: if we cannot find 'our' concepts in the implementation, we must instead take the implementation's concepts and make them ours. As a simple example, suppose we take a neural network and identify a single neuron N_F with an output value in $[0, 1]$ to signify the presence of some feature F. We can then clearly say whether the implementation judges F to be present or not by looking at the value of N_F. As such there is a simple correspondence between the new ARL state "the feature F is present" and the IL state "the neuron N_F is firing". Because there is this well-defined correspondence, the knowledge of F seems to exist in the system, but it does not seem to *explicitly* exist in the system, as it is only treated as a 'distinctive' piece of knowledge because of our interpretation as such. To the system, N_F is just another neuron without any special meaning. As such, it seems reasonable to say that the system has *tacit knowledge* of F.

However, we don't yet know anything about what the feature F *means*, so to discover this,

[14]From the conception of connectionist systems they have been viewed as difficult to interpret, the reason being that the 'features' a connectionist network makes use of have no clear semantic interpretation. For example, Smolensky asks: "What kind of subconceptual features do the units in the intuitive processor represent? Which activity patterns actually correspond to particular concepts or elements of the problem domain? There are no systematic or general answers to these questions at the present time; seeking answers is one of the principal tasks for the subsymbolic research paradigm" (Smolensky, 1988, p. 7).

we must investigate the role of N_F in the implementation. This leads Clark to propose we point our attention towards approaches such as *cluster analysis*, which look at the implementation to discern ‘interesting’ states, and builds a higher-level analysis from those. These bottom-up methods form the third category of methods in explainable AI, and we will see some examples of these as well in the following section.

To summarize, it can be said that the main difference between the two approaches to ‘bottom-up’ explanation is that we have (1) the approach which is descriptive of the system’s (in/output) behavior and (2) the approach which is descriptive of the system’s behavior *and* the internal *structure* of the system. Clark argues this ‘structurally descriptive’ approach is necessary, and I tend to agree: if the goal is to ‘clarify’ a black-box system, it will not be enough to stick to methods which only aim to approximate the same in/output behavior. A black box system is, after all, a system only defined by its in- and outputs, so using these kind of explanatory models we are essentially exchanging one black box for another, though possibly a very simple (and badly approximating) one. This adds very little to our understanding, and more importantly sidesteps the actual problem of trying to understand black-box models. In that sense, an approximating model which also *structurally* mirrors the implementation has a much stronger epistemic status: it actually *uncovers* information about the ‘inside’ of the black box, which before was inaccessible.

2.6 Contemporary approaches to AI explanation and their limitations.

In this section I will look at a few publications in explainable AI research which present methods to generate explanations of the behavior of these systems. There already exist many overviews of explainable AI research[15], so I will focus on a perspective which frames them using the distinctions I have just introduced: specifically, Clark’s three views on the interactions between various levels of explanation.

The Newtonian view, where higher-level explanations are only descriptive of the system’s (in/output) behavior, is common in explainable AI applications. Oftentimes, a specific behavior of the system is clarified by using methods which express *which ‘parts’* of the

[15]One good example is the article by Zednik (2019), which relates some common methods in explainable AI using similar concepts as are used in this thesis.

input are used to produce the output. *Saliency maps* (Montavon et al., 2018) for example are often used when the input is an image: the pixels in the input image are colored or otherwise highlighted in a way which shows whether they contributed positively or negatively to the output decision. These provide some idea of where the system's 'attention' is focused and may be helpful in identifying certain causal relationships. An example is given by Zednik (2019):

> "Consider a well-known historical (albeit probably apocryphal) example in which a neural network learns to visually distinguish enemy tanks from friendly ones. Although the network quickly learns to categorize images of tanks, it does so by tracking an accidental correlation between tank allegiances and weather patterns: whereas the images of friendly tanks were all taken on a sunny day, the enemy tanks were photographed under cloud cover. For this reason, although the system correctly classifies images (*what*), its reasons for doing so (*why*) have nothing at all to do with tanks!" (Zednik, 2019, p. 13)

On the other hand, while knowing where the system's attention is focused may show it is doing the wrong thing, it *cannot* show that it is doing the *right* thing. An example is given by Rudin (2018): different classifications of an image may be 'looking' at the same spot simply because there is the most information there, but then we have no explanation why the system chose one classification over another. Similarly, if the classification is wrong but the system is looking in the right spot, we have no explanation why it is wrong at all. As such, these methods can only provide a very limited type of explanations.

More generally, there are a number of techniques which train explanatory models to approximately reproduce the system's behavior (locally, i.e. only for inputs 'close to' a certain input). A common technique is *LIME* (Ribeiro et al., 2016) which approximates the system as a linear function mapping inputs onto outputs. If the input is specified as a list of features, the influence each feature has on the output of the system will then be given by (the size of) its coefficient in the linear function when it is written as a linear combination of features. If necessary, it also subdivides the input into an amount of features suitable for human inspection: for example, an image will be divided into a number of pixel regions (e.g. using some kind of edge detection algorithm) such that the effect of each 'feature' region on the output can be measured.

These methods have in common that they provide a simplified description of the system's behavior, but in no way do they look 'inside', and so they remain silent on whether the actual processing involved is anything like the approximate model suggests. As Rudin (2018, p. 4) puts it: "even an explanation model that performs almost identically to a black box model might use completely different features, and is thus not faithful to the computation of the black box". As such, explanations generated using these methods are susceptible to Clark's critique of 'Newtonian competence': the explanation may be descriptively and predictively accurate, but it cannot elaborate on what kind of processing is *actually* going on inside the system.

The 'rogue' view of competence is less commonly found in explainable AI. From this perspective, the implementation is only a heuristic 'tool' which delivers the same 'answers' (input-output pairs) as specified the by higher-level descriptions. As such, the implementation does not contain any more explanatory value with regard to the problem we are trying to solve than the higher-level descriptions can give us. An approach towards explainable systems therefore needs to incorporate the task of 'giving explanations' into the system's specification of 'competence' or the CT-level specification. For a typical neural network, this means that it must be *trained* to produce explanations in the same way it is trained to produce classifications or predictions. As such, a training data set should be built of not only input-output pairs, but of input-output-explanation triples.

Some examples of this approach are those by Codella et al. (2018) and by Ehsan et al. (2017). Here, the data set used for training consists not only of performance data, but also human-generated explanations of these performances. For example, Ehsan et al. (2017) had human agents play a game and vocalize their thoughts simultaneously, producing a data set relating the environment, actions and 'rationalizations' of the actions to each other. The system is then trained to produce both the actions and the rationalizations given the state of the environment as input. Because the system is trained to produce actions and reasons for them simultaneously, one might expect it to discover connections between the two, it the same way it discovers relationships between the input and output in a regular machine learning setting. Moreover, because the processing for generating action and explanation is shared, one might expect that an 'efficient' configuration of the system would be one in which (some of) the same computations used to generate the action are also used to generate the explanation, leading to the two sharing a common causal factor[16].

Finally, we have the investigative 'bottom-up' approach, which discards the competence theory as a primary source of information about the system, and instead analyzes the implementation to extract some kind of 'overview' of the processing it performs. Clark argues in favor of this approach, as in both of the other scenarios the competence theory is explicitly *not* a description of the connectionist system, but of something else, leading to weaknesses in the higher-level explanations. Instead, one should work from the implementation upwards, generating an explanatory model which does not only functionally (in/output-wise) agree with the connectionist implementation but which also *structurally* agrees with it (as discussed in the previous section).

These types of investigative analyses are also common in explainable AI under the moniker of *feature visualization*[17]. For example, in a typical neural network such a method may try to find *what kind of input* (not necessarily encountered in practice) causes a certain neuron to respond through optimization, i.e. it simply calculates the numerical input values which maximize the neurons output, similarly to how neural network is trained normally to minimize the loss function. Note that, while a single neuron is very simple (a nearly linear function ot its inputs), if we consider a neuron a few layers deep in the network it will be provided with the results of all prior processing, and so the output of this neuron is in fact a very complex function of the inputs. Therefore inputs which maximize it may contain many kinds of structure, possibly encoding higher-level concepts.

For example, the research by Olah et al. (2017) has found ('late' or 'deep') neurons in image classification networks which roughly respond to specific human-recognizable concepts such as 'an upward curve' or 'a dog's fur'. However, while these examples are informative, single neurons are not *guaranteed* in any way to be parseable. Olah et al. (2017) also find a number of examples of this, which they call 'polysemantic neurons'. In later research, they

[16]This is only a speculative argument: I don't think anything like this has been verified to happen in practice. In fact it seems it would be quite difficult to do so, as one would first need a good understanding of the neural network to discover whether the two outputs are indeed causally related, putting us back where we started.

[17]'Visualization' implies a visual representation of the feature, which is often the appropriate because many black-box systems of interest deal with image processing. However, the idea of trying to extract a description of a feature from the implementation is of course not necessarily limited to the visual modality.

have further developed this idea, looking instead at *combinations* of neurons and trying to establish relationships between them (Olah et al., 2018), even going so far as to speculate this is a *universal* way of describing neural networks at the ARL:

> "Three Speculative Claims about Neural Networks
>
> **Claim 1: Features** Features are the fundamental unit of neural networks. They correspond to [...] linear combination[s] of neurons in a layer. Often, we find it most helpful to talk about individual neurons, but [...] there are some cases where other combinations are a more useful way to analyze networks - especially when neurons are "polysemantic." [...] These features can be rigorously studied and understood.
>
> **Claim 2: Circuits** Features are connected by weights, forming circuits. A "circuit" is a computational subgraph of a neural network. It consists of a set of features, and the weighted edges that go between them in the original network. Often, we study quite small circuits - say with less than a dozen features - but they can also be much larger. [...] These circuits can also be rigorously studied and understood.
>
> **Claim 3: Universality** Analogous features and circuits form across models and tasks." (Olah et al., 2020)

Basically, on top of the neural network, they try to build another kind of network, one which isolates identifiable 'features', leading to a higher-level understanding of the processing going on. This is clearly an approach which incorporates the idea of 'bottom-up' investigation as formulated by Clark (1990). Such a method undoubtedly provides valuable knowledge about the processing structure of the implementation, and makes it possible to effectively *ground* ARL descriptions of the system in the implementation via vertical explanation. However, the approach by Olah et al. cannot be completely universal with regards to explanation in AI: it is still specified in the specific language of 'layered' neural networks. If a different kind of black-box model is used, this conceptual framework will not be applicable and something new must be developed. Moreover, looking at explanations of the kind as idealized by T. Miller (2019) in Figure 2.2, they do not make any reference to such a computational mechanism. In other words, if we would like to produce these kind of human-like or *intentional stance*-explanations, these methods are not sufficient on their own. They only produce a higher-level model, not explanations. The computational description at the level of 'circuits' still needs to be interpreted, and this remains up to the researcher.

We have seen that there are contemporary implementations for each of the three attitudes toward explaining connectionist systems, and that there are different kinds of explanations they produce. They either shine a light on the functional relationship between the output and specific parts of the input, they uncover higher-level structure in the internals of the system, or they derive associations between input-output pairs and explanations by learning from examples. However, none of them are on their own able to produce human-like explanations of the connectionist system: they either produce a model which needs to be interpreted to form an explanation, or they produce an explanation which is disconnected from the actual connectionist implementation. Where to go from here? It does not seem wise to look for *another* 'new' kind of approach towards explanation, as there are already quite many, and they all provide some explanatory utility. Instead, I would like to argue that if we wish to produce satisfactory explanations without analysis and interpretation by an expert, it is necessary to use them *together*. I will elaborate on this suggestion in the next section.

2.7 A suggestion: integrative explainable AI.

In the previous section I have presented a number of methods which are part of the contemporary field of explainable AI. I hope that from my examples and arguments it is clear that, while each of the methods deliver *some* explainability, none really delivers the 'right' or 'full' explanation in any scenario. Perhaps this is to be expected: given the variety of systems and the different possible ways in which to understand them, it is maybe naive to expect there to be any one-size-fits-all method of explanation. A better approach might be to take the different methods in explainable AI as 'pieces of a puzzle' which have to be fit together properly in order to explain a system. This is not an easy task for a number of reasons, however, these same reasons also illustrate why it may be an important one.

First, as we have seen before, people expect explanations at relevant levels of detail, which may differ between situations. However, they also expect explanations given in different situations to be *consistent* with one another. This means that, if we have two behaviors B_1 and B_2, and generate horizontal explanations E_1 and E_2 for them independently, we have to make sure the overall picture of the system given by (E_1, E_2) is consistent[18]. As such, explanations at the same level of detail are not independent. Similarly, explanations at different levels of detail are not independent either: E_1 and E_2 might display the same high-level inferences, but be vertically grounded by explanation of two different low-level mechanisms A and B. Alternatively, E_1 and E_2 might display different high-level inferences, but both be vertically grounded using mechanism A. In that case, the appearance of both E_1 and E_2 necessitates the further explanation of A to show why it leads to different inferences between the two situations.

This interaction between explanations at various levels or parts of the system is, in my view, underexplored in explainable AI research. Sure, it is reasonable to have an initial focus on trying to extract at least *some* information which can be used for explanation (one needs some pieces before a puzzle can be built), but even then it is important to have a clear conception of what goals these efforts are trying to reach. In any case, the idea that among the various efforts in explainable AI *one* will emerge as the 'definitive' explanation generator, is arguably unrealistic and should at least be contrasted with a different conception of what explainable AI is aiming for.

For example, looking at the current research on explainable AI methods, Clark's two bottom-up approaches are clearly present, but a top-down interaction is rarely taken into account. This could be seen as a limitation if we compare the explanatory capabilities of AI systems with those of human agents, as humans may often *adjust their (low-level) behavior*

[18]Existing methods may fail this requirement, even for similar inputs. For example Alvarez-Melis and Jaakkola (2018) found that models based on saliency or LIME (among others) are not *robust*: "[Alvarez-Melis and Jaakkola] conducted experiments where they slightly perturbed input values that were entered into the black-box model. They found that the [approximate models] and the original black box models produced stable output values in response to the perturbed inputs, but that the perturbations often caused the *explanations* provided by LIME and Shapley to change drastically. In short, explanations generated by LIME and Shapley values are not robust to small changes in input values, even when the outputs of the [approximate models] are" (Hancox-Li, 2020, p. 3).

based on some high-level explanation they have given, in a process known as **rationalization**:

> "Rationalization occurs when a person has performed an action and then concocts the beliefs and desires that would have made it rational. Then, people often adjust their own beliefs and desires to match the concocted ones." (Cushman, 2019, p. 1)

Despite their 'fabricated' nature, we still accept these kind of rationalized explanations, even though we cannot tell whether they really correspond to *what actually drove someone to act*. In the Newtonian view there would be no problem with this, as high-level explanations only need to be reasonable descriptions, without any deeper connection to the actual reasons for an action. However, the fact that people then adjust their behavior shows that there is something else going on as well. When giving an explanation such as this, one is also making a certain *commitment*, namely that in the future their behavior will be consistent with these reasons (Summers, 2017). This indicates that the explanation also partially determines the behavior, which is only possible in the 'rogue competence' view. As such, to give explanations with the same properties to those we expect from human agents we need *both* bottom-up and top-down interactions, which means that an integrated view of explainable AI is necessary.

Second, a skilled AI researcher might be able to use the tools currently at their disposal to build a relatively complete understanding of specific system. However, given the increasing amount of AI systems used in practice, the interpretation of which can differ a lot between problem domains due to their flexibility, it would be useful to *automate* (or build-in) as much of the work of explaining systems as possible. This becomes even more relevant when we consider an interaction between a system and a 'regular' user. For example, as argued by Mittelstadt et al. (2019), model-based explanations may provide a 'tool' to discover something about the system, but these are not usable for a non-expert user. As such, the user remains dependent on the researcher to answer any questions about the system they might have, and it is not possible for the user to gain further understanding of the system by interacting with it directly. To make this possible, there should be an automated way of at least generating an acceptable, cohesive explanation.

Therefore, having the tools to obtain explanatory pieces only solves part of the problem. If we want *practical* explainable AI, the task of putting together the puzzle is just as important, and also just as difficult. As there is no single 'best' explanation of a specific event, there is most likely also no single 'best' way to construct an explanation from several parts.

Of course, the idea of integrating explanations has been proposed before, but it is not nearly studied as much as 'fundamental' explanation generation. Wang et al. (2019, p. 12) found that "users [employ] a diverse range of XAI facilities, to reason variedly. Therefore, while much work has focused on developing good specific XAI algorithms, more work is needed to [integrate] multiple explanations into single explanations". Like Mittelstadt et al. (2019), other authors have also placed emphasis on the 'barrier' between an explanatory model and an actual explanation which is suitable for a user. For example, Doran et al. (2017) emphasize the importance of *reasoning* in explanations:

> "[Existing] approaches are lacking in their ability to formulate, for the user, a line of reasoning that explains the decision making process of a model using human-understandable features of the input data. Reasoning is a critical step in formulating an explanation about why or how some event has occurred [...]. Leaving explanation generation to human analysts can be dangerous since, depending on their background knowledge about the data and its domain, different explanations about why a model makes a decision may be deduced. [Explanatory] models thus enable explanations of decisions, but do not yield explanations themselves[19]." (Doran et al., 2017, p. 7)

An example which puts this into practice is the work by De et al. (2020), which combines a 'structural bottom-up' approach with functional approximation: it performs clustering on the internal states and uses this to construct a decision tree which approximates the decisions of the system, in order to output logical inferences which describe the reasoning for a particular decision. An example which focuses on human-friendly presentation is the work by Weitz et al. (2019), who incorporate a virtual agent that delivers the reasoning in intentional terms.

In addition to these approaches, I believe that it is worthwhile to investigate this problem of integrative explanations in AI further, and semi-independently from the general problem

[19]This statement seems confusing, as we *are* generally looking for explanations which explain the system's decisions (see Section 2.2). What I think they mean by this is that the models enable certain horizontal explanations (about behavior) which are however not grounded using vertical explanations (about the system itself).

of producing explanations. By that I mean that explainable AI should not consider connecting various methods of explanation only as an ‘iteration step’ in the search for better explanation generation algorithms, but rather attempt to find different ways of conceptualizing the relationships between various explanatory methods as a somewhat separate venture, with its own challenges and interesting research questions. This leads me to my proposal for explainable AI, which is threefold:

1. To treat the problem of *integrating* explanations in AI as a separate research direction from the problem of generating (partial) explanations.
2. To view existing methods in explainable AI as components of a to-be-devised *framework of explanation*.
3. To look at integrated theories of cognition or *cognitive architectures* as a source of inspiration for such a framework.

These first two points I will not argue for extensively, but I hope the last few sections have given some context as to why integration in explainable AI is a problem worth considering. The rest of this thesis will focus on the third point, taking cognitive architectures specifically as a starting point for developing these kind of integrated explanation systems.

Chapter 3

How might cognitive architectures aid explanation in AI?

After looking at explanation in general and explanations of AI systems in particular, we come to the topic of **cognitive architectures (CAs)**. The term “cognitive architecture” has been historically used to describe “the structure of cognition”, specifically the structure of the human mind. The definition of “cognitive architectures” used here is a newer, more specific one. In contemporary cognitive science literature, a cognitive architecture is generally understood to be “a computational formalism for expressing computational models of cognitive phenomena” (Varma, 2014, p. 1). By specifying certain constructs and mechanisms as a basis for cognition, these formalisms embody certain theories about the mind, and attempt to show how interactions between these constructs lead to an ‘intelligently’-behaving system. Experimentally, cognitive architectures “attempt to provide evidence what particular mechanisms succeed in producing intelligent behavior and thus contribute to cognitive science” (Kotseruba & Tsotsos, 2018, p. 5). They can be used to validate certain theories of cognition by showing that the behavior generated by an implementation of the theory in such an architecture matches observations from human experiments. Usually, cognitive architectures are ambitious in their aim: they try to provide a comprehensive model of human cognition, or a *unified theory of cognition* (Newell, 1990; Varma, 2014).

Of course, what is interesting for this thesis is not so much whether one of these architectures is a ‘correct’ description of the way human cognition is structured, but rather, how they can be used to *explain* cognitive phenomena. Which aspects of cognition do these architectures explain exactly, how do they arrive at their explanations, and are the explanations they deliver suitable for the purposes of understanding an AI system in the ways outlined in Chapter 2? The hypothesis here is that embedding complex AI systems into such a comprehensive architecture may provide new insights into ways of explaining their behavior.

To evaluate whether cognitive architectures may be useful for the purpose of explaining AI systems, we must first have a clear picture of what cognitive architectures are and what they are not. Especially *the ways in which they explain cognitive phenomena* need to be clarified. Using the ideas from previous chapters, such as different *levels of explanation*, the different *kinds of understanding* and the difference between *performance* and *competence*, we can make an assessment of the explanatory utility of cognitive architectures and their relationship to explainable AI.

3.1 The origins of cognitive architectures.

First, it is worthwhile to look at the context in which cognitive architectures initially emerged. From the 1950's onwards, the field of *cognitive psychology* took a leading role in psychological research. Cognitive psychology is perhaps most easily described by contrasting it with its predecessor: behaviorist psychology or *behaviorism*. The idea behind behaviorism is that psychology should not try to make claims about concepts 'inside' the mind, such as mental states, as those are unobservable from the outside and can therefore not be scientifically studied. By "changing the subject to the study of behavior, psychology could become an objective science based on scientific laws of behavior" (G. A. Miller, 2003, p. 141).

Cognitive psychology was a departure from behaviorism, and was established when the latter's limitations started to show:

> "As Chomsky remarked, defining psychology as the science of behavior was like defining physics as the science of meter reading. If scientific psychology were to succeed, mentalistic concepts would have to integrate and explain the behavioral data. We were still reluctant to use such terms as 'mentalism' to describe what was needed, so we talked about cognition instead." (G. A. Miller, 2003, p. 142)

In other words, the main tenet of cognitive psychology is that, while it is important to be scientifically grounded, it is also necessary to talk about (possibly unobservable) mental concepts such as *memory* or *perception*. As such, the progression from behaviorism to cognitive psychology is one moving from treating the mind as a only 'a black box which produces behavior' towards a description which includes *internal* concepts as well. Just like in any other science, these concepts may then be precisely and formally described using *theories*, which may be experimentally validated. And like in all sciences, these theories have a number of functions, one of which is *explaining* something about the object studied: in this case, the mind.

Taking this as the defining quality of cognitive psychology clearly establishes the relevance of the work of Alan Newell (1927 - 1992), which culminated in his 1990 monograph *Unified Theories of Cognition*. He argues that, if the primary objective of cognitive psychology is establishing theories of the mind, it makes sense that a possible 'end goal' could be a universal 'theory of everything' with regards to cognition, just as it is in a science like physics. His idea of a **unified theory of cognition (UTC)** is relevant to this thesis because it encapsulates (and in practice, is essentially equivalent to) those things which are more commonly called 'cognitive architectures': comprehensive computational specifications of cognitive functioning.

Therefore, to establish what constitutes a 'cognitive architecture', a good first step would be to look at what Newell had in mind when talking about a UTC. A *theory* according to Newell is a "body of explicit knowledge, from which answers can be obtained to questions by inquiries" (Newell, 1990, p. 13). This is a broad definition, but he also attributes them more specific properties. First, a theory should be objective: the answer obtained to a question should not depend on the person who is asking it. In addition, a theory is always approximate, and therefore always wrong, but "how [it] is wrong is carried along as part of the theory itself" (Newell, 1990, p. 14). Finally, theories are "things to be nurtured and changed and built up" (Newell, 1990, p. 14). A useful theory, even if known to be 'wrong', may still be useful, either because it can be changed to agree with experimental knowledge, or because it is known *how* it is wrong, so it may live on with a more specific domain of application.

Note that this notion of a theory is seemingly not entirely compatible with what we require of our explanations. Given such a theory, one may generate an explanation for a certain behavior, but as understood here the explanation cannot be dependent on the needs of the person who asks for it. This means that the social aspect of explanation is hard to align with such an objective theory, and raises an obstacle which must be overcome in order to utilize a UTC for the purposes of 'pragmatic' explanation. One direction in which this may be resolved is by considering the position of *usefulness* as a core indicator of a good theory, and the idea that a theory is *dynamic*: it may change if the circumstances require it. As such, while a theory should be objective, it is not immune to being judged and influenced by its environment either.

Given this idea of a theory, Newell then names a 'unified theory of cognition' as being a "single set of mechanisms for all of cognitive behavior" (Newell, 1990, p. 15). This is a bit more restrictive than the name suggests: a UTC is not just a body of knowledge about cognition, but it is focused on an explanation of cognitive *behavior* using *mechanisms*. This means that explanations generated by a UTC would involve a description of mechanisms and their interactions, and should be clearly relatable to the behavior which is generated by them. This second point is emphasized when Newell clarifies that a UTC is not a what he calls a (somewhat condescendingly) "content-free" description:

> "A unified theory of cognition does not mean a high-level theory that gains its unification because it backs off to vagueness and platitudes. Calculations and simulations count." (Newell, 1990, p. 16)

If we view this in the context of Marr's levels, it becomes clear that a UTC is a theory which encompasses not only as much of the 'behavior space' as possible, but also one which connects the higher and lower levels of explanation. It needs to specify the overarching principles of the system, yet also enable one to perform a simulation of behavior. For example, a 'computational theory' as Marr defines it would not be sufficient as a UTC, even if it were applicable to all cognitive behavior, because it is *vertically incomplete*. One could not build a simulation of it without further specifying the system at lower levels.

This is one aspect of UTCs, and by extension cognitive architectures, which make them interesting for explanatory purposes. Not only do they connect different behaviors under a single theory, providing a sense of *generality* and *consistency* to various explanations, they also *explicitly connect different explanatory levels* of the system. As we have seen in Chapter 2, a large obstacle in explainable AI is the fact that contemporary AI systems are often 'underspecified' at the algorithmic and representational level. UTCs are interesting because they provide 'general' (i.e. applicable to all cognitive behaviors) formalisms which can be used to substantiate this intermediate level.

It is key that a UTC should not be an entirely 'new' theory of cognition. Rather, it provides a way "to integrate and synthesize knowledge within the existing paradigm [...] , not to propose new questions and radically different answers. [...] The function of [a UTC] is to precipitate and consolidate [...] the saturated solutions of empirical results and successful microtheories that currently exist" (Newell, 1990, p. 395). As such, a main goal of UTCs is to relate existing theories to each other and integrate them into a cohesive account of cognitive processes.

In other words, a clear advantage of UTCs is that they are *unified*. They can be said to (aim to) provide a universal 'language' for describing any kind of cognitive process, and so if we take AI systems to be a subset of cognitive processes, a language for describing any kind of AI system. This language is not a language of the spoken kind, but rather one expresses through descriptions which consist of combinations of certain computational mechanisms. In addition, these descriptions should connect high-level functional specification (*what* and *why*) and low-level behavioral performance. However, one question remains: should we expect that such a 'universal' language will be suitable for the purposes of intuitive, 'everyday' explanation as is sought after in a field like explainable AI? This question of the *explanatory value* of cognitive architectures is treated in the following section.

3.2 The explanatory value of cognitive architectures.

Now that we have an idea of the purpose with which cognitive architectures were initially proposed, the question remains whether they are relevant to the field of explainable AI: in other words, do they help us to explain the black-box AI systems we are trying to understand, in terms which are accessible to researchers as well as non-scientist users?

A look at Newell's motivation for the proposal of UTCs does not provide us with much of an answer. This is because Newell is mostly concerned with scientific explanation of human cognitive phenomena, so he places primary importance on agreement with human experimental data. His reasons for developing UTCs (Newell, 1990, p. 18) are concerned with the increased ability of such a theory to explain and predict experimental results such that they are *identifiable* and *unifying*. Identifiability means that the theories must produce results which correspond to experimental results in such a way that they cause us to prefer the UTC over other candidate theories, and they must be unifying in that they provide a *single* paradigm which explains many different kinds of experimental data in this way[1]. However, there is not much motivation for the characteristics of these explanations: how are they structured, which aspects of the system do they emphasize, and how are they communicated? We know they take on the form of a mechanism-based description, but this is only motivated by their usefulness in achieving goals of unification, identifiability, etc., not by the utility of mechanism-based explanations in terms providing understanding.

Instead, we must look at the results (the cognitive architectures themselves) and try to judge *in what way* their explanations help to explain cognition. One such analysis is performed by Varma (2014), who distinguishes between the *objective* and *subjective meaning* of cognitive architectures, along various levels of explanation. Unfortunately, the terminology used here is not very clear: what exactly constitutes the 'meaning' of something is a bit vague, and it is not clearly defined by Varma (nor does there seem to be a common established definition which he makes use of). Nonetheless, we can try to get some sense of what *he* considers it to mean[2].

By his account, the difference is that an *objective* analysis of CAs is concerned with "how it supports models that correspond to the phenomena of cognition", while a *subjective* one describes "how it helps cognitive scientists understand cognition" (Varma, 2014, p. 1). In other words, while an objective analysis is concerned with concepts such as measurable performance (i.e. agreement with experimental data), the subjective analysis is more concerned with *understanding*. This seems like an reasonable distinction to make, and appropriate to the context of explainable AI as well. Analogously one might say that, while the *objective* utility of AI systems is their ability to accomplish some task, their *subjective* utility lies in how they help researchers understand the task and how it is solvable. In AI we often see 'objectively' well-performing systems whose internal structure is inaccessible

[1]Newell predicts the problem of identifiablity will be solved by trying to find a UTC: "The way to solve the identifiability problem is by bringing to bear large, diverse collections of quasi-independent knowledge [...] that finally allow the system to be pinned down. The more unified the theory, the better the chances of identifiability, because then the chance of being able to bring diverse sources of knowledge to bear is greater" (Newell, 1990, p. 22).

[2]Indeed, very ironic.

so that they provide very little 'subjective' value. As such, explainable AI is also concerned with a 'subjective' analysis of AI systems. Varma's subjective analysis states that cognitive architectures provide researchers with a specific understanding of cognition, which forms when they "learn to see cognition through the architecture's *worldview*" and internalize "the distinctive information processing style it attributes to cognition". This happens through the process of becoming familiar with the architecture (e.g. "a year or more of sustained effort") (Varma, 2011, p. 1333). This nicely aligns with the idea of CAs as providing a kind of 'language' one can use to discuss cognitive phenomena. When one learns to 'speak' this language, it provides one with a new way to understand the cognitive processes at hand. For example, cognitive psychologist John Anderson recalls how the ACT-R cognitive architecture took the role of a *lingua franca* among a community of cognitive researchers:

> "[The language of the theory] became a language spoken among all members of the community, rather than a language spoken by authors of the theory to readers of the theory. Sharing the language among so many forced a greater standardization and consistency which in turn made it possible for a wide range of researchers to contribute to the theory." (Anderson, 2007, p. 41)

In other words, CAs have been used not only for the purpose of accurate prediction of experimental data, but also for purposes of *knowledge sharing*, such as discussing cognitive processes and exchanging ideas.

Any specific cognitive architecture will provide a different perspective on what constitutes cognition and a different way to express the same cognitive phenomena. By looking at the "worldview" or "paradigmatic principles" (Varma, 2011, p. 1333) a CA articulates, one can establish an account of which kind of understanding it provides, which provides a way to evaluate the utility of CAs as scientific explanatory frameworks. Varma (2014) gives such an analysis, divided over Marr's three levels. At each level a CA provides different constructs which help one to describe cognitive phenomena:

1. At the implementation level, CAs postulate a set of *cognitive primitives*: the smallest discernible objects which are necessary to describe how cognitive behavior emerges. For example, logical propositions which can be combined in order to construct complex reasoning behavior.

2. At the algorithmic level, they provide certain *idioms*: "combinations of cognitive primitives that solve [recurring] problems" (Varma, 2014, p. 6). For example, a certain component of the architecture might describe how memory is handled, and this same component can then be applied in various cognitive scenarios.

3. At the computational theory level, they postulate a *processing style*: they give the general principles which drive the system to function (*what*), and the motivation for where these principles originate (*why*). For example, a 'rational' optimality-based processing style may lead to a system which tries to calculate the best possible outcome in order to decide on an action.

Different CAs will substantiate each of these levels differently. In the case where we are trying to extend or model our example neural network as in Chapter 2, we can already establish a few constraints which such a system should satisfy. At the lowest level, we know that the *primitives* which are used are the neurons in the network and the operations they perform. Clearly, these cannot be substituted, though they may be joined by different kinds of primitives, e.g. ones which carry additional information used for purposes of explanation. On the highest level, we know that the behavior of the system is directed towards optimization of a certain numerical goal, such as accordance with a certain data set on which it has been trained. This should also be reflected in the 'processing style' of any architecture which encapsulates it. On the algorithmic level, there are generally few constraints, but commonly a neural network is assembled into *layers*, which means there is some sort of hierarchical or sequential processing going on. Each layer has (only) the output of the previous layer as input, so one could say the information is iteratively 'shaped' to correspond to certain features in the input, as is commonly assumed in e.g. applications of feature visualization[3].

From these constraints, we can already look at some examples of cognitive architectures to see how they might be able to model or extend our example system. However, there is another way in which to clarify the explanatory value of cognitive architectures, which is

[3]By this I mean the assumption that neurons in 'later' layers correspond to features which are more complex than those in 'earlier' ones, and that these complex features are somehow constructed by assembling them from the simpler features.

especially useful in this case as it uses the same terminology of *competence* and *performance* as we have seen before. I will discuss this view of cognitive architectures in the following section.

3.3 Cognitive architectures as competence descriptions of cognition.

As we have seen, from their origins cognitive architectures are principally designed with their predictive qualities in mind. They must give a picture of human cognition which is as broad as possible, and as accurate as possible. In other words, cognitive architectures are generally focused on explanation of *performance*, as is measured in psychological experiments. However, there is another way to view cognitive architectures or UTCs, which is interesting for their explanatory role: as descriptions of *competence*.

In a review of *Unified Theories of Cognition*, Fehling (1993) critically evaluates Newell's ideas on why and how a UTC should be developed, and what we should hope to gain from it. Fehling's main criticism of Newell is that he, in fact, seems not to stray very far from the behaviorist principles which cognitive science is supposed to supersede:

> "Newell's approach seems quite compatible with the behaviorist doctrine in its absolute commitment to a UTC as a system for simulating human performance. Newell, like [John] Watson before him, seems to feel that a psychological theory is meaningful only in terms of its ability to describe or predict human performance. [...] [M]any behaviorists such as E.C. Tolman have been as willing as Newell to postulate mediating processes in their accounts of human behavior. [...] [F]or both Tolman and Newell, the validity of a mediating construct [...] derives from objective measurement of observable behaviors." (Fehling, 1993, pp. 309–312)

In other words, to Newell the central qualification of both a theory and the constructs it contains comes from their utility in making predictions about human behavior, such as reaction times, and this puts him very close to the behaviorists. This is however not a criticism of his idea of UTCs. To the contrary, Fehling sees certain advantages of the theoretical constructs Newell introduces which his focus on performance prediction does not do justice. He proposes an alternative view of UTCs as "competence models", which according to him "better exploits the significant strengths of Newell's theories of knowledge systems and intelligent cognition" (Fehling, 1993, p. 317).

The concept of a 'competence theory' has been introduced in Chapter 1, but Fehling's treatment of it has some notable details. First he introduces the familiar picture of a competence theory as an *idealized* description of the *knowledge* necessary to perform a task, which may be *tacit* or implicit, and the structural relationships between different pieces of knowledge (Fehling, 1993, pp. 298–299). He then talks about theories of **cognitive competence**, which according to him are theories of *knowledge-based agency* (Fehling, 1993, p. 317). In other words, they describe the knowledge (and relations between pieces of knowledge) which enable an agent to act in a certain way. This makes his idea of a cognitive competence theory explicitly intentional-stanced: it is not simply a statement of causally related events, but rather a description of how an agent came to act. This naturally leads to descriptions of behavior that make use of mentalistic concepts, such as saying the agent 'believes x' (i.e. it possesses a piece of information that says x is true) or 'did y because it desires z' (i.e. it has (1) a goal z and (2) a piece of information that specifies y leads to z, and used this information to deduce it should y in order to reach z). Therefore, if we have a way to interpret CAs as theories of cognitive competence, this provides a way to discuss them using an intentional stance: as descriptions of an thinking, acting agent, rather than a mechanistic process.

In addition, a cognitive competence theory should consist of *generative* processes that can produce the *potential actions* an agent may take (Fehling, 1993, p. 318). A competence theory is generally not 'strongly' predictive, in the sense that e.g. a theory of linguistic competence will not predict what people will say given a state of the world. However, it may be 'weakly' predictive, in the sense that it may predict what they *might* say or *likely will not* say. It is concerned with an abstract, idealized description of language, leading to questions such as "is this an English sentence?". Similarly, a cognitive competence theory would try to answer questions such as "how is the agent able to x?" rather than "how did the agent x?".

In the same way, while CAs may predict which behaviors are possible or not, they are too general to give a definite predictive account of an agents internal state. This is partially a practical (measuring) problem, but even then it seems unreasonable to expect that any observable state of an agent corresponds to an exact state of the CA which models it.

Rather, its utility is much more obvious when one uses it to investigate what kind of behaviors an agent *may* show, and to give an account of how these *may* arise. In this way, the *restrictive* nature of CAs is emphasized, which is just as important: just as it is easy to make a grammar which generates every possible sentence, it is also easy to specify a mechanism which can perform every possible computation. The ways in which a CA is limited make it more interesting than, for example, a Turing machine[4]. This emphasis on restrictions is also found in contemporary research on *bounded rationality*, where it is studied how the *limitations* of human cognition lead to certain behaviors.

Furthermore, according to Fehling a *unified* theory of cognitive competence should clearly consist of multiple distinct competences. This is because the kind of competence which is required depends on the situational context. This fits with the modular structure of CAs: one does not give a single description of the cognitive system, but rather a description of distinct components and the ways in which they may interact. The behavior which may result from these interactions is minimally constrained, as to provide the largest possible coverage of phenomena while using only a few constructs. This relates a 'module' in a cognitive architecture to a partial competence or 'subcompetence' of the cognitive system.

The idea of cognitive architectures as competence descriptions can also be found in more recent accounts, when one 'reads between the lines'. For example, John Laird, who worked together with Newell on one of the first cognitive architectures, has a recent publication (Laird, 2012) in which he presents his conception of cognitive architectures. He stresses that a cognitive architecture on its own, despite Newell's commitment to measurable performance, is not a complete description of an agent:

> "A cognitive architecture provides a framework for encoding knowledge, but on its own it doesn't generate behavior. Moreover, representations of knowledge without an architecture are like programs without a computer - they do nothing. Representations of knowledge need the architecture to retrieve them and combine them with other representations, whereas an architecture without content is like a computer without software - it is an empty shell.

[4]Newell (1990, p. 21) also mentions this idea: "[s]howing that a theory implies universal computation destroys its usefulness, because that makes anything possible", referring to Peters and Ritchie (1973).

> It is the combination of knowledge and the architecture that leads to purposeful behavior." (Laird, 2012, p. 18)

Here Laird states that a cognitive architecture only provides a *basis* for a working agent, but that in the end it is the information stored and processed using the architecture which determines their actions. This is similar to the function of a competence description: it describes the essential capabilities and limitations of an agent, but the way in which these capabilities manifest themselves as *performance* can differ wildly between different instances of the same competences. Laird elaborates on this when discussing the difference between *architecture-based capabilities* and *knowledge-based capabilities*. The difference between these two is whether a capability is *explicitly* implemented in the architecture, or whether it is meant to *emerge* via the combination of specific knowledge with more general processing mechanisms[5]. He names an example of numerical addition:

> "For example, basic arithmetic, [...] can be done using knowledge in Soar through pure symbolic operations in which individual numbers are represented as individual digits, which are then added together, with carrying and so forth. The time required to add two numbers is linear in the number digits, and requires using the agent's knowledge about arithmetic. If these operations are done in the architecture, with appropriate technology, they can be done in constant time as part of a single operation, independent of the number of digits, up to a very large number of digits." (Laird, 2012, p. 15)

I would like to compare this to the role of competence theories in language processing. When one talks about the kind of capabilities a speaker of a language has, one may distinguish between the capability to recognize a sentence as being grammatically correct, and the ability to grasp the meaning of a specific sentence. In principle, these capabilities are independent: one may recognize the functions of words and the validity of the sentence structure without knowing what the words mean exactly, and one may grasp the meaning simply by recognizing a few 'key' words without having a full grasp of the grammar of the language. We may (if we for example adhere to Chomsky's theory of generative grammar) view such a speaker as an agent running on an architecture which has the inherent capability to process certain structured inputs (i.e. sentences). But even if one can

[5]He also compares this distinction to the difference between RISC and CISC computer architectures.

process the structure of a sentence, certain knowledge is necessary to grasp the meaning of a sentence and act upon that meaning. In this case the grammatical understanding is provided by an architecture-based capability, while the interpretation of the sentence is a knowledge-based capability.

This distinction by Laird shows that even though the architecture provides a basis, it can differ how extensive this basis is: most of the operations needed to perform the task may be implemented in the architecture, or the architecture may be left very general, and the necessary operations must be 'learned' by the agent. This, of course, makes a difference to the expected performance: if a capability is explicitly implemented, one can make stronger guarantees about its properties than when it needs to be learned autonomously. This means the architecture specifies the *essential* capabilities of the agent: the ones it necessarily must have. These are then roughly identifiable with the competences the agent possesses, as is the case when the capability to recognize grammatical structure is often deemed 'essential' in competence theories of language.

From the description of cognitive architectures as competence theories of cognition, their explanatory value becomes quite clear, especially in the case where we already have an artificial system which performs the behavior we seek. They provide a mechanistic description of the competence(s) of a cognitive system. Moreover, they aim to be generally applicable, so given a system which we aim to explain, it is likely that there will be (part of) an architecture which 'fits' the existing constraints of the system. Then, such an architecture points us towards parts of the system which may be 'missing', i.e. which do not have an explanatory equivalent in the system under analysis. This then allows us to pinpoint *where* the lack of understanding in the system may be located, or in other words, where it may be augmented in order to provide further understanding. Alternatively, we may use this to 'place' existing explanatory methods in the context of a larger framework which helps to clarify which kind of 'competences' of the system they explain and how they accomplish this.

It is not the purpose of this thesis to suggest new methods of explaining black-box systems, so while approaches found in CAs may inspire entirely new methods, I will not speculate about these too much. Instead I will again discuss the explainable AI methods treated in Chapter 2, and try to view them as components of such an architecture, in order to better

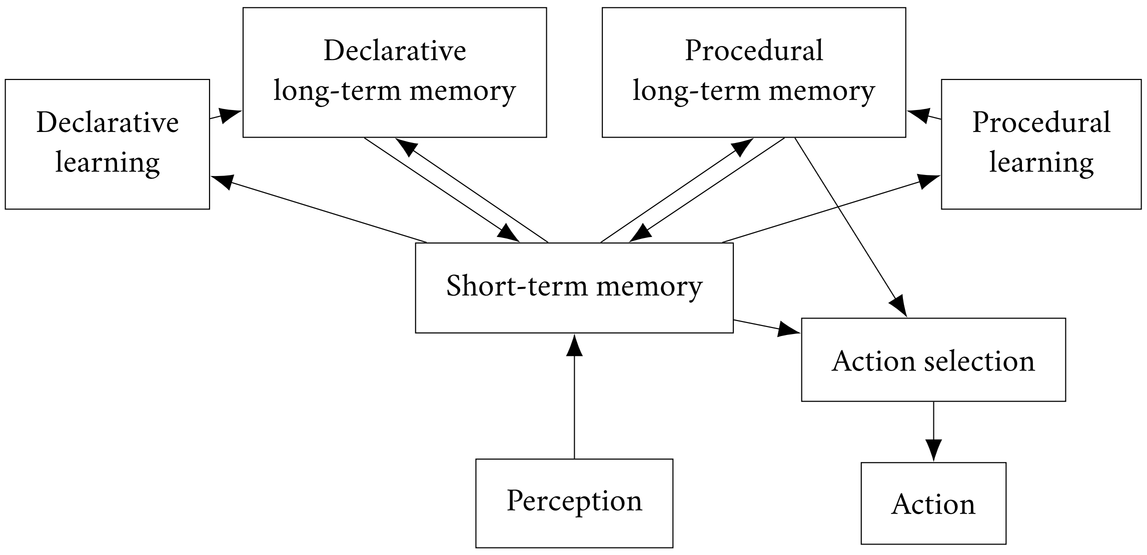

Figure 3.1: An abstract overview of a 'prototypical' cognitive architecture. Adapted from Laird (2012, Figure 1.3, p. 12).

understand their function and the interaction between different kinds of explanations. But first, I will look at a few well-known cognitive architectures and their (mainly high-level) structure, in order to see what kind of component systems we encounter and how they are meant to interact.

3.4 Examples of contemporary cognitive architectures.

In this section, we will look at a few examples of common cognitive architectures, and evaluate them from an 'explainable AI' perspective, using the terminology by Varma and that of competence theories.

A 'typical' pseudo-architecture. As a starting point, we can look at the "prototypical cognitive architecture" presented by Laird (2012), reproduced in Figure 3.1. This gives us a general overview of the kind of components which are often included in cognitive architectures. That these commonly shared components exist is most likely because most CAs attempt to model human cognition to some extent. This 'pseudo-architecture' can be viewed as an extension of the regular perception-reasoning-action cycle which is commonly applied in agent programming. The main difference is that the 'reasoning' step is split into a number of different 'memories': components which each store information, but in a different way and for a different purpose.

First, a 'perception' module converts sensory information into some internal representation. This is then provided to a short-term memory (STM), in which it can be directly stored. This STM is connected to two kinds of long-term memory (LTM), a *declarative* and a *procedural* type of memory. The difference between the two lies in their representation: declarative memory is meant to store *facts*, for example as logical statements, while procedural memory is meant to store *processes*: think 'muscle memory'. This is closely related to the philosophical concepts of *knowing that* versus *knowing how*[6]. When information is stored in the long-term memories, the process of *learning* takes place. This makes sense: the STM is used to store information for a single 'interaction' with the environment, while the information in the LTM is stored for a longer time so that it can be recalled later. This means that *the way in which information is being selected and stored in the LTM* determines *what* the agent is 'remembering', or learning. Finally, information from all the memories is combined to determine which *action* to take at the current time.

This pseudo-architecture provides a skeleton which different concrete CAs substantiate in their own way. They may differ in the types of memories they use, the way in which they interact, how perceptive information is represented, and on what basis the actions are chosen. I will look at three examples of prominent cognitive architectures in order to illustrate some possibilities in which these aspects may be substantiated.

Soar (Laird, Newell, Rosenbloom). Soar is one of the oldest CAs which is still being developed today, and as the architecture pioneered by Newell (among others) it is the prototypical example of what constitutes a 'unified theory of cognition' according to him. The unifying property of Soar is that it treats all cognitive processing as *problem space searches*. The idea is that all processing is meant to achieve certain goal states starting from some initial state. Actions should be selected in order to follow a *path* from the initial state to a goal state. As this path has to be found, it views the 'problem' a cognitive processor

[6]Moravcsik (1974) discusses this difference between *knowing how* and *knowing that* in the context of competence.

tries to solve as a search problem: given a space of possible (world) states and *operators* (actions) which can be applied to transform from one state to another, the problem one aims to solve is to find a sequence of operators which moves from the initial state to the goal state (also called a *plan*). However, such a path may not be able to be found easily: one might encounter a state in which the operator to be applied can not be determined conclusively. Here Soar introduces the concept of *chunking* or hierarchical problem spaces. If the next operator cannot be determined given the 'knowledge' one has right now, we can establish a new sub-goal: to determine the right operator to apply. This sub-goal can then similarly be achieved by problem space searching, establishing more sub-goals in the process if necessary. This idea of problem-space search can be viewed as the 'language' in which Soar expresses all cognitive computation.

Soar has been around since the early 1980s (Laird, 2012, p. 19), and as such has undergone many revisions, each of which has added additional complexity. However, a few general properties of Soar have remained constant over time. One relevant property is that Soar is mostly *symbolic*. Sensor data (which may be non-symbolic) arrives in a perception module and here "symbolic structures are extracted" (Laird, 2012, p. 17) and added to the working memory. This memory and all others store symbolic information. For example, procedural knowledge (knowledge of how to do something) is represented by *production* or *situation-action rules* (if x then y). Because all information stored after the perceptual stage is symbolic, the processing Soar performs in order to make decisions mostly takes place in a symbolic context[7].

What does this say about Soar's relevancy to the problem of explainable AI? We are trying to explain the inner workings of a black-box system. Such a system has no place in Soar's symbolic world, and thus the most we can accomplish is to try to 'emulate' the system's behavior using Soar. This means that we build a Soar agent which performs (roughly) the same as our non-symbolic decision-making system, but gives us access to a symbolic internal state. However, for this to be meaningful there needs to be something which

[7]In recent versions, some kinds of non-symbolic processing is performed as well (Laird, 2012, p. 21), however this seems minimal. Examples are numeric preferences used to decide between rules when there are multiple possibilities, or the addition of 'mental imagery', a sort of perceptual feedback loop. However in the end all information about the environment must still be represented symbolically before it can take part in the reasoning process.

suggests a deeper connection between the two systems (the black-box and the Soar agent). In other words, we seem to be stuck in Clark's 'Newtonian' conception of explanation, where the explanatory model (the Soar agent) merely 'fits' the behavior. We will merely have two models which express the task of 'recognizing bird sounds' (or any other) in two different ways, without being able to make the claim that one explains the other.

Soar provides us with a general agent architecture and an associated kind of 'language', which tries to frame any kind of cognitive task as *goal-directed hierarchical problem-solving*. But even if the 'Soar language' of the problem-space search turns out to have additional explanatory value for the task at hand, it would be difficult to use Soar to explain a given black-box model. Instead, if one wishes to describe the computation using Soar, it would be better to entirely replace the black-box model by a Soar agent trained on the same data (or in the same environment), not to claim that one model explains the other.

CLARION (Sun). Like Soar, the cognitive architecture CLARION can also succinctly be motivated by the hypotheses about cognition it aims to support. From early on in its development, an important aspect of CLARION has been the distinction between **explicit and implicit knowledge**[8]. As explained by Ron Sun:

> "Generally speaking, implicit processes are less accessible and more "holistic", whereas explicit processes are more accessible and more crisp [...]. This dichotomy is closely related to some other well-known dichotomies in cognitive science, for example, the dichotomy of symbolic versus subsymbolic processing [...]. The dichotomy can be justified psychologically by the voluminous empirical studies of implicit and explicit learning, implicit and explicit memory, implicit and explicit perception, and so on [...]." (Sun, 2015, p. 2, references omitted)

This combination of two different 'fundamental' types of processing (symbolic and subsymbolic) has led to CLARION (and others like it) being called a *hybrid* architecture.

Similarly to the 'prototypical' architecture presented earlier, CLARION's structure is mostly informed by the various types of memory which it distinguishes. It consists of four main components ("subsystems"), each of which has a "dual representation", in that

[8]For example, a precursor to CLARION is Sun's (1995) research on combining *rule-based* and *similarity-based* reasoning. These two types of reasoning make use of explicit and implicit knowledge respectively.

it contains both an explicit part (implemented as e.g. production rules) and an implicit part (implemented as a type of neural network). This leads to both two different types of reasoning which interact and also two types of representations which do so: a localized, explicit representation and a distributed, implicit representation.

For the kind of system we are trying to understand, this distinction is an interesting one, because it establishes a link between the 'implicit' connectionist system which we would like to explain, and the 'explicit' rule-based model (such as a feature-possession model) we would like to use to explain it. CLARION[9] incorporates both kind of systems into a single 'unit', which means it provides new possibilities to establish the relationship between the two: one need not merely be an 'Newtonian approximation' of the other, but there can be a more extensive interaction.

In Hélie and Sun (2010) this interaction between the explicit and implicit systems is presented in detail. The two parts are referred to as the *top level* (explicit) and *bottom level* respectively. At both levels, different information may be stored, but it is expected that they store some information *redundantly*:

> "Often, concepts are (redundantly) encoded in the bottom level (using distributed representations [...]) and in the top level (using localist representations) of CLARION. If only the top-level representation is activated, a top-down signal may be sent to activate the corresponding representation in the bottom level (known as *implicitation*). Likewise, if the bottom level is activated, a bottom-up signal may be sent to activate the corresponding representation in the top level (known as *explicitation*). [...]
> [I]t is often the case that the equivalent forms of knowledge [...] will be able to access or activate each other [...]. When mutual activation happens, complex insights are more likely to happen [...]." (Hélie & Sun, 2010, p. 1003)

This two-level system is depicted in Figure 3.2. Computationally, the bottom level is a

[9]To be precise, a single subsystem: the non-action-centered subsystem or NACS, which is essentially the 'declarative memory' of CLARION. The others (for example, the motivational subsystem) are not so relevant in the limited scope of our examples.

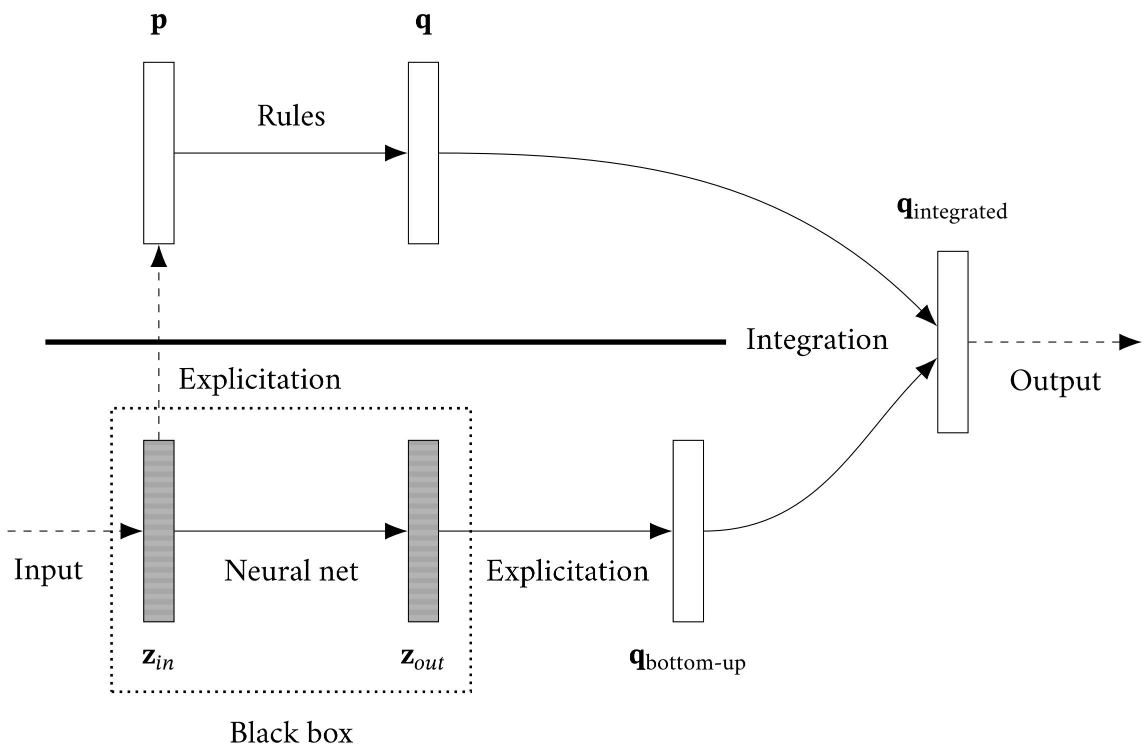

Figure 3.2: A schematic overview of the explicit-implicit distinction in the NACS (non-action centered subsystem) in CLARION, in the particular case where the input to the system is subsymbolic and the output is symbolic. The top and bottom level are separated by a thick line. In the top, a rule-based process is used to transform a symbolic state $\mathbf{p}$ into $\mathbf{q}$ (both binary vectors). In the bottom level, a neural network transforms a subsymbolic state $\mathbf{z}_{in}$ into a new state $\mathbf{z}_{out}$ (both real vectors). Then, through an explicitation transformation, the subsymbolic state is transformed into an equivalent symbolic state $\mathbf{q}_{\text{bottom-up}}$, and the two $\mathbf{q}$-states are combined into a single output $\mathbf{q}_{\text{integrated}}$. In this particular case, the input is subsymbolic and so is placed into $\mathbf{z}_{in}$, after which it must be explicitated into input $\mathbf{p}$ for the top level.

‘black-box’ type of neural network[10], while the rule-based processing of the top level is modelled by a linear binary neural network consisting of two layers, where each node represents an atomic proposition p. If we have a vector of binary (0 or 1) input variables $\mathbf{p} = (p_1, p_2, ...)$ and a binary matrix V, the output vector $\mathbf{q}$ is then calculated as $\mathbf{q} = NV\mathbf{p}$, where N is a matrix which normalizes the values in $\mathbf{q}$ to lie in the interval $[0, 1]$. This equation implements simple ‘rules’ which relate each ‘output proposition’ q to a number of input propositions $p_i, p_j, ...$, and q has a ‘fuzzy’ truth value (or alternatively, a probability of being true) which is determined by the proportion of $p_i, p_j, ...$ which are true. This can be straightforwardly interpreted as a feature-possession model: if every q refers to (the probability of) a certain output judgment (e.g. type of bird) and each p refers to a certain feature of the input (e.g. pitch of the bird call), then V attaches a set of input features to each output judgment q, and the more of these features are present, the more likely the output judgment q is made.

Both the top and bottom level processes can be run in parallel. When both have produced results, the bottom-level result vector $\mathbf{z}$ is *explicitated*[11] or transformed into a vector of output propositions $\mathbf{q}_{\text{bottom-up}}$ by another configurable (linear) transformation[12]. These two output vectors are then *integrated*: they are combined into a single vector $\mathbf{q}_{\text{integrated}}$ using a kind of *OR*-operation[13]. This means the final output is produced by both an ‘implicit’ black-box system and a more ‘explicit’ rule-like system.

As such, CLARION provides interesting ideas about how to combine a black-box system and a more ‘interpretable’ system in a way such that interaction in both directions is

[10]In Hélie and Sun (2010), an *attractor neural network* is used. This is a kind of recurrent neural network that has a single ‘layer’ of state, which changes over time according to some configurable interaction rules (‘weights’). As a simple analogy, one can compare such a network to a cellular automaton such as Conway’s Game of Life. For each ‘computation’, this state is evolved in time until a ‘steady’ state is produced: one which (almost) does not change on application of the rules. This is then the output state.

[11]Perhaps a more specific term to refer to the current characterization of explainable AI would be *explicable AI*, in the sense that to a large extent we aim to explicitate the (implicit) concepts used by a black-box system and the relationships between them.

[12]Using another transformation of this kind, the implicit knowledge about the input can also be explicitated into a vector $\mathbf{p}_{\text{bottom-up}}$. In addition, explicit knowledge (e.g. the contents of $\mathbf{p}$ or $\mathbf{q}$) can also be *implicitated* by transformation into a state vector $\mathbf{z}$ for the bottom-level neural network.

[13]To be precise, a max-function is applied, which is a fuzzy-logic equivalent of *OR*.

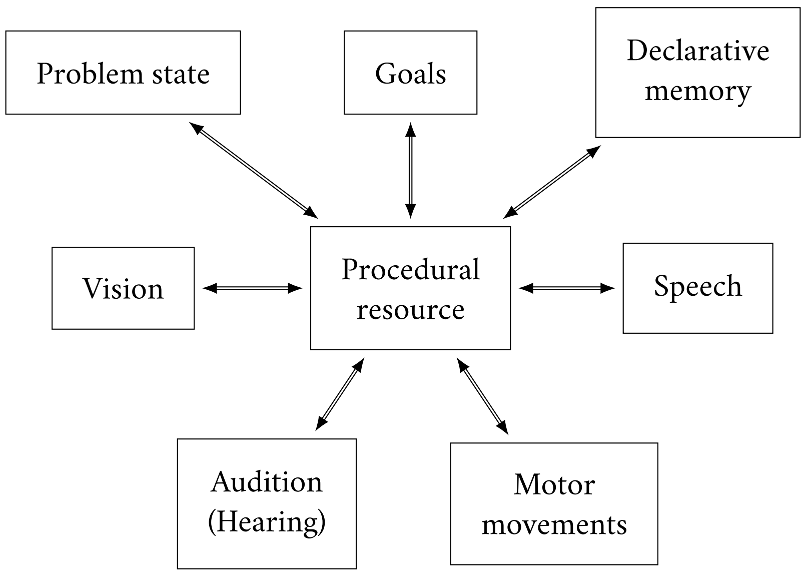

Figure 3.3: An abstract overview of the ACT-R cognitive architecture. Adapted from Salvucci and Taatgen (2010, Figure 2.2, p. 30). Each box constitutes a *resource* or *module* which handles a single type of task. At the center is the *procedural resource*, which manages the interaction between different resources.

possible, and *both* systems influence the final output. This means that there is no 'primary' system of which the other one is a descriptive approximation, but that both subsystems are necessary components in producing the integrated system's behavior, which is then observed. It can therefore be reasonably said that this is an example of a 'stronger' or more substantive connection between a black-box system and a more explanatorily useful model, in the sense advocated for by Clark (1990).

ACT-R (Anderson, Lebiere). Unlike the generic but abstract task of 'problem solving' which architectures like Soar and CLARION attempt to solve, ACT-R takes an approach which is more concretely inspired by the specific functionality of the human brain, and its conception of modularity is quite different as well. Many agent architectures contain bottlenecks which limit the kind of processing to a question-answer style of computation. For example, they may have a single entry point where 'perception' happens, which provides a representation of the environment to various computational mechanisms. These then work together to produce a single 'answer' in the form of an action to be performed, which provides feedback to the environment. In other words, many agent architectures stick to a single percept-reason-action loop. ACT-R is more *situated*, in the sense that its components correspond to certain capabilities (such as a sensory capacity or a motor function) which can independently interact with the environment, and perform computations in parallel[14]. This is due to ACT-R's commitment to capture the *multitasking* capabilites of the mind: according to its hypothesis, the mind provides various *resources* which can be used to perform cognitive functions, and it may be possible to work on two problems at the same time, as long as these problems require different resources. The resources and the way they interact with each other is schematically shown in Figure 3.3.

In the example of a simple bird-recognizing agent, we only need a single 'perceptive' source (the audition module), as such an agent is not really multitasking in the sense described above. However, the interplay between vision, memory, goals, and eventual output[15] is still nontrivial, and functions different from how it would in another architecture such as Soar. This interaction is managed by the *procedural resource*, which maintains and applies "production rules that match patterns in particular resources and, if the match is successful, perform actions in other resources" (Salvucci, 2014, p. 3). Like other CAs, ACT-R contains two kinds of 'knowledge': "*declarative knowledge*, which incorporates a network of interrelated facts, and *procedural knowledge* in terms of production rules" (Salvucci, 2014, p. 3). Like Soar, this knowledge is essentially symbolic: in addition to the symbolic production rules, the declarative knowledge is processed in terms of *chunks*, which are

[14]In an abstract sense, one may compare the difference between an architecture such as Soar and ACT-R to the difference between von Neumann-type and Harvard-type computer architectures. While single-loop agent architectures can process only one 'instruction' (input) at a time, the structure of ACT-R allows it to perform a kind of *pipelining* so that it can process multiple different kind of inputs in an interleaved fashion.

[15]We do not really have to have the agent relay the output to the environment using a special module: as long as it can construct and recall a declarative statement such as 'input x is a bird of type y' we can say the classification process has occurred.

discrete 'units' of knowledge comparable to the features we have discussed before[16]. This makes it difficult to establish a link to the subsymbolic black-box implementation, but it remains an interesting example of a different way in which such an architecture may be structured. In tackling the *integration* of explanations in explainable AI, it is informative to consider this idea of 'modules' which run in parallel to each other, and interact via a central coordinator.

3.5 A architectural view of explainable AI.

Now that we have seen some examples of cognitive architectures, the question remains how exactly these ideas may inform the practice of explainable AI. We can find an answer if we look at each architecture's computational paradigm, 'worldview' or 'language', and imagine what happens if we translate this to the context where we have to structure a system whose aim is not to perform cognitive tasks, but to *explain a black-box system*. We would like such a system to be similarly general yet capable of providing explanations in many different contexts. For each of the cognitive architectures discussed before, I will take one of their distinctive characteristics and connect it to a specific approach to explainable AI, in order to see what how they might inform each other.

Explanation as part of the problem (like Soar). An obvious approach we can take to address the new task of generating explanations is to incorporate it into the CT-level problem statement. In the context of (supervised) machine learning, that means we extend the data set which the system is trained to emulate to also include explanations of behavior. This unified approach which treats all processing as problem solving through computation is similar to the philosophy of Soar. In some sense, we simply add additional 'goals' which our system tries to fulfill simultaneously as the goal of providing good

[16]There is some notion of subsymbolism in ACT-R, but rather than occurring on a lower level than the symbolic processing (i.e. the algorithmic-level symbols corresponding to implementation-level subsymbolic patterns), the symbols are treated atomically but have some subsymbolic processing applied to them. To be precise, the declarative chunks get assigned numerical *activation values* which determine which chunk gets used ('recalled') in a certain situation, and similarly the production rules have *utilities* which allows for a decision between them to be made in case of conflict (Anderson, 2007). However, there remains an underlying assumption that some symbolic knowledge can be correctly identified, but this is one of the main problems we aim to solve in explainable AI.

prediction or classification performance. The question which remains is then whether the explanation provided is really reflective of the processing the system performs, or if it is just a reasonable enough rationalization of the decisions made.

If we look at Soar as inspiration, we should ask what gives it the ability to provide a clear 'story' of the processing it performs. The answer is that it is strongly *hierarchical*. If we look at a single input-output pair (or in Soar terms, an initial state and goal state), we should be able to provide a horizontal account of the processing performed in terms of a *path* between the two states consisting of the *operators* applied. If along the way substantial sub-problems are encountered, they will be horizontally processed in a similar fashion (by Soar's *chunking* mechanism).

This suggests a way to strengthen the explanatory value of learning rationalizations: by adding hierarchical structure. If we were to try and explain a Soar agent's processing, we would have to lay out the chain of operators, and for each operator provide a partial explanation. By identifying the operators with some linear hierarchy present in a neural network (e.g. the layers), we see that for a similar kind of explanatory account to the Soar paradigm we should be able to separate the rationalizations produced into parts, and associate each of these to one of the 'processing stages' (the layers). This means that each explanation provided in the data set should be separated into its logical constituents. As a simple example, an rationalization of the form 'A because B' may be divided into parts 'B', 'B implies A' and 'A'. Then, these small parts or 'micro-rationalizations' should be derivable from successive processing stages[17]. This lends further credibility to the claim that various parts of the rationalization given are indeed causally connected, as the micro-rationalizations are derived from the processing stages, which are constrained to have a certain causal relationship. This mirrors the way in which a Soar account of a process divides the computation into a set of operators which are constrained to be applied one after another. If one can assign definite explanations to each operator (given context) in Soar, the full horizontal explanation of a process clearly follows. Therefore, if

[17]Note that for this example to work it does not have to be the case that stage 1 of the system needs to be a 'B detector' and stage 2 a 'B to A converter'. The only thing which is required is that if one micro-rationalization is applied after the other, the related processing should also take place after the other. But if we think of layers in a neural network, there is no hard constraint on what tasks layers may perform, only that it must be possible to retrieve the micro-explanations using information from a specific stage only.

we subdivide the rationalization into smaller parts, it should be sufficient to check whether these parts stand up to scrutiny. Because these parts then respectively correspond to smaller parts of the system, it may be more easily verifiable that their specific realization (i.e. the corresponding states in the implementation) agrees with the explanation given.

Implicit processing and explicit explanation (like CLARION). Perhaps the main issue which makes it difficult to give concrete explanations of the behavior of black-box systems (especially connectionist systems), is that there is no good way to explicitly articulate the concepts the system uses to represent the world and perform its computations. This is a problem which CLARION provides some interesting perspective on, in its coupling of an implicit (black-box) and explicit (rule-based) system such that representations may be translated between the two. In the model from Hélie and Sun (2010), a correspondence is established between a recurrent neural network which evolves from an initial state to a stationary output state, and a simple rule-like network which infers an atomic proposition (or a probability distribution over proprositions) from a set of input propositions. In addition, both take part in the learning process as they jointly generate the output (so during e.g. backpropagation both will be adjusted) and because of the correspondence both may influence each other as well (depending on what is 'given' and what is inferred, i.e. is the external input symbolic or subsymbolic?).

Comparing this to the output of a feature visualization method, we see that they are actually quite similar. In feature visualization, generally a sort of 'overlay' network of features is constructed, onto which the neurons of the original network may be mapped. The aim is then that both the interpretation of the features and the connections between them is somehow 'simpler' than the original network, leading to a more accessible representation of the internal state. This translation between the 'low-level' and 'high-level' networks is similar to CLARION's interplay between the top level and bottom level of the subsystems. In fact, in Olah et al. (2020) the definition of 'feature' used is "a linear combination of neurons in a layer". This is exactly the same way in which explicit representations (propositions) are linked to implicit representations in CLARION: by a linear transformation applied to the bottom-level neuron activations. The main differences are that CLARION's implementation is (or can be) bidirectional, whereas the high-level network of features is completely generated from the low-level internal state, and that CLARION's networks do not involve multiple layers. However, both aim to establish the same kind of correspondence between a set of symbols (features or propositions) and activation patterns in a subsymbolic network.

This indicates that there may be some fruitful exchange of ideas possible between the two different contexts. The approach of feature visualization by optimization provides a general way to establish which bottom-level states correspond to top-level symbols, e.g. which kind of inputs strongly activate a certain 'feature detector'. In CLARION's case, this is simply given by the (linear) transformation between top- and bottom-level states. By applying this transformation, the top-level propositions p can easily be related to specific subsymbolic patterns, which allows one to assign a meaningful interpretation to the propositions (assuming they have not been given one a priori).

However, suppose one wants one to extend the top-level model to not only link a set of input propositions to output propositions, but to incorporate intermediate propositions (e.g. complex features) as well, so as to lead to a more detailed view of the 'reasoning' performed by the system. This creates a problem, as when top-level propositions correspond to 'hidden' neurons in a multi-layer network, their interpretation becomes more difficult. Luckily, this is exactly the problem feature visualization aims to solve. Because subsymbolic patterns in the 'middle' of the network have no meaningful interpretation, we may instead look at inputs which (strongly) generate these patterns. Feature visualization as in Olah et al. (2017) accomplishes this by adjusting the input through an optimization process so that it 'activates' (generates) the desired 'feature detector' (subsymbolic pattern). As such, the feature visualization approach may inform CLARION in the sense that it provides a flexible way to construct interpretations of learned explicit representations.

On the other hand, a significant contribution of CLARION is the more extensive role it assigns to the top-level system. In the approach by Olah et al. (2020), the subsymbolic system performs all computation, and the symbolic 'overlay' is purely descriptive. Their research is focused on showing how an analysis like theirs might lead to insights about the inner workings of neural networks. However, this means that they do not (aim to) establish *how* exactly such an analysis is constructed: for example, how does one choose which linear combination of neurons constitutes a 'good' feature? They seem to view that as the work which is to be performed by the scientists, possibly over a extensive time period:

> "What if we treated individual neurons, even individual weights, as being worthy of serious investigation? What if we were willing to spend thousands of hours tracing through every neuron and its connections?" (Olah et al., 2020)

While interesting from a research perspective[18], for the more 'everyday' kind of explanations this kind of time investment is obviously not feasible. I have emphasized the importance of accessibility of explanations multiple times, and to require every machine learning program to be extensively studied by experts does not provide a way to make them easily understandable to users. Instead, there needs to be a way to automate this process of analysis; ideally the system in question would be extended with the capability to locate the relevant features (and connections between them) and relay them to the user *by itself*.

CLARION tries to achieve something like this by incorporating the top level into performing the computation. If the symbolic 'overlay' derived from the bottom level plays a role in producing the output, it can also be trained via a similar process as the rest of the network. This means that the explicit representations can be determined through learning, in the same way that the implicit representations are. Additionally, this allows for 'strengthening' of explanations by giving them *consequences*. As mentioned before, in human communication we are aware that people often cannot be entirely sure of what motivated them to act, and may only provide rationalizations. However, one reason we accept those rationalizations is because we expect the person providing it to adjust their behavior to act in accordance with it, as to not act hypocritically. This kind of behavior adjustment can only be realized if there is some influence from the features used in the explanation over the outputs produced. In a bidirectional model such as CLARION's, that is achieved by constructing the output using both systems, not only the subsymbolic one. This may also allow for the possibility of systems to accept feedback on their explanations, for example to adjust the learned explicit representations to agree more closely with concepts easily recognizable by humans.

[18]Such a thorough analysis of single networks seems especially interesting from the perspective of scientific research on cognition: for example, in trying to find and understand general computational mechanisms that may be used to perform specific kinds of visual processing which are not yet well understood.

Threaded explanations (like ACT-R). As discussed earlier, one problem in the field of explainable AI is that there are many different techniques which all extract some explanatory value from the system being investigated, but whose explanations do not necessarily agree with each other or complement each other well. For example, we might have a saliency map which highlights a certain part of our input, yet have a feature-visualization model whose features do not neatly correspond to the areas highlighted by the saliency map. Or there might be two local explanatory models which give entirely different explanations in similar cases[19]. One solution to this problem is to only ever use one method of explanation at a time, and to treat separate explanations as independent analyses. This is comparable to the one-size-fits-all approach of an architecture like (old) Soar (or to a certain extent, CLARION), where all processing happens in one direction, and all processes are treated as homogeneous (e.g. problem-space search). And just as one researcher might come to a different conclusion as another, independent explanatory algorithms might come to generate different explanations of a system. However, this retains the requirement for a 'researcher's perspective', meaning that the explainee is able to evaluate the contesting explanations and incorporate them into a cohesive picture for themselves. This leads to low accessibility of the explanations for regular users of a system.

Taking ACT-R's point of view, we are provided with more possibilities for integration. In ACT-R, different modules can be treated heterogeneously, and their computational properties may differ. However, the procedural resource ensures that there exist causal connections between them, which are highly intelligible (being if-then rules). In a similar way, one could place various explanatory 'modules' which generate different types of explanations next to each other, and include a coordinating module which assembles the explanations into a cohesive 'story'. This could be causal or time-related in the same way that ACT-R accounts of processing are, but other ways to relate them can for example be found back in the analysis in Chapter 1. Different explanations could have (1) a horizontal (causal) relationship (a-then-b), (2) a vertical (grounding) relationship (a-because-b), or (3) a 'similarity' relationship (a-like-b), corresponding to each of the three explanatory dimensions: respectively (1) time, (2) level of analysis and (3) behavioral similarity.

[19]These examples are hypothetical: one could argue these are simply shortcomings of the methods used, but my goal here is to provide another perspective which does not rely on finding a caveat-free or 'universal' method.

Let me propose an example: suppose we have a black-box model trained to classify images from a certain data set, and we try to explain its behavior using a saliency map and a feature visualization model. For each input presented to the system, the saliency map will point to certain areas of the input which it deems to be highly relevant. The coordinator module might then use this information to train or adjust the feature visualization model to try and choose features such that they highlight similar areas (by optimization), leading to a set of features which indeed is relevant to the computation the system performs. In this way, the saliency map can inform the feature visualization model.

Additionally, the features selected by the visualization model can be used to give context to the output of the saliency map: instead of highlighting certain areas, the salient parts of the input can also be described in terms of features, which allows one to (1) generate examples of these features, showing what the system 'thinks' it is seeing when looking at a certain area, and (2) to follow path of these features through the overlay network, which gives an more detailed idea of how exactly the features are used and what kind of influence they have on the computation. In this way one may establish a causal relationship between the two kinds of explanations (features are selected because of their saliency) and a grounding relationship (the 'meaning' or informational content of salient areas is substantiated by the feature visualization model).

One can also think of other connections: for example, if a system can generate rationalizations based on a data set, it may be desirable to ground them by connecting them to the feature visualization model as well. Using optimization, one could try to find parts of the system which respond to both the presence of a certain feature in the input, and the usage of a certain concept in the associated explanation. In this way, a correspondence between the concepts used in a rationalization and the processing structure can be established, which leads to a more reliable explanation.

Conclusion

In this thesis, I have taken an extensive look into the explainability of black-box systems, taking a more holistic perspective than is commonly done in explainable AI. I have emphasized the importance of the interaction between the person receiving the explanation and the system providing it, endorsed the view that good AI explanation is 'human-like' explanation, but also tried to cover the complexity of giving explanations about black-box systems in the first place. I have tried to create an accessible overview of this complexity through various distinctions:

- First, the distinction between different levels of analysis on which explanations may be given (Section 1.3),
- Second, the distinction between different kinds of explanations: horizontal or temporal explanations, vertical or 'grounding' explanations, and explanations of different (but possibly similar) behaviors, leading to 'local' or 'global' explanations (Section 1.5),
- Third, the distinction between descriptions of competence and descriptions of performance, which is similar to the distinction between levels, but draws a different (and sometimes more relevant) boundary (Section 1.6),
- Fourth, the distinction between different ways to view the relationship between a description of performance (the black-box system) and competence (roughly, an explanatory system) as presented by Clark (1990) (Section 2.4).

Using these distinctions, I have provided an overview of and some critique on contemporary methods of explanation of black-box systems (Section 2.6). Additionally, I have tried to illustrate the explanatory utility of cognitive architectures, and to establish a parallel between this utility and the needs of contemporary explainable AI. To wrap up this discussion, I have chosen a few interesting questions which I will try to concisely answer:

1. What is difficult about explaining AI systems? The main challenge for explainable AI that is often presented is to find the right balance between *accuracy*, which leads to complexity, and *transparency*, which is hampered by complexity. Rudin (2018) considers this a false dilemma, as many 'simple' models may provide adequate performance in most cases. This may be true in general, but if we assume our aim is to clarify (the decisions of) a black-box model, there is something to this idea of finding a 'balance', as when constructing an explanatory model for this purpose one has to look for such a balance as well. Specifically, the explanatory model must be accurate enough that it can be considered representative of the black-box model, yet transparent enough that it can be understood better than the original black box. This means that a challenge in explainable AI, as in 'the field researching how to explain (the decisions of) black-box models', is to *find the right level of detail* for the explanations (cf. explanatory models) it produces. I believe this challenge is recognized by current research in explainable AI, and it provides the reason why for example Mittelstadt et al. (2019) argues against model-based explanations (too low-level, hard to interpret correctly) while Páez (2019) argues for them and against explanations focused on specific behaviors (too high-level, disconnected from the system).

A second challenge for explainable AI is, in my opinion, to *clarify what exactly the explanations produced tell us about the black-box system*. This is a topic which seems to be less prevalent in the literature, and to find analyses of this nature one is better off going back to the late 80's/early 90's (as I have done), where the idea of connectionist systems was still new and there was much skepticism about their proper integration in the production of scientific knowledge. Nowadays, I think this has become less of an issue, possibly because (1) connectionist models have 'proven themselves' in many tasks, and as such the skepticism about them has mostly faded, and (2) because of the statistical nature of how connectionist systems achieve their performance, this means that an affirmation of connectionism automatically brings with it an affirmation of the underlying philosophical basis, namely that all cognitive tasks are essentially statistical pattern matching. This then leads researchers embedded in the connectionist paradigm to develop similar conceptions about what makes a good explanatory model: one which is statistically accurate to the system it explains. As such, a descriptive notion of competence (Section 2.4) is readily accepted.

One thing I hope is made clear in this thesis is that there are other perspectives on this relationship between a black-box system and its explanations, and that these alternative perspectives are no less relevant today than they were 30 years ago. The big questions posed by people such as Marr on 'mechanism-based approaches' (Section 1.3), Clark on the conception of competence in black-box systems (Section 2.4), and Smolensky on numerous aspects of interpretation of connectionist systems (referenced throughout, see Smolensky (1988)) remain unanswered, and whatever the answers may be, they will likely not be of a statistical nature. Moreover, I hope to illustrate these are not merely theoretical (i.e. analytic) concerns, but may also have practical (i.e. empirical) implications: if the relationship of an explanation to the system is not clear, it becomes more difficult for an 'ordinary' user to trust the system's decisions, and it also makes it difficult to produce multiple explanations (of different behaviors, or different types of explanations) which still allow for a cohesive picture of the system as a whole to form.

2. What makes an explanation 'good'? Next to the structural properties an agreeable explanation seems to have (see the discussion on T. Miller (2019) in Section 2.2), one may also ask the question of what makes the content of an explanation reliable. We have seen that, while scientifically there might be very rigorous notions of what constitutes an explanation (Section 1.2), when talking about the human context many kinds of explanations are often accepted which are not demonstrably correct, such as rationalizations (Section 2.7).

This makes it difficult to establish strict criteria for AI explanations in general, in a way that they are accessible to different types of users of a system. There should be a balance between accuracy and complexity, which is the problem of finding the right level of detail. The difference is that, even if we have an explanation at the right level of detail, we cannot always be confident in what exactly it explains about the system at hand. This can be seen in Clark's critique of the descriptive (Newtonian) and top-down (rogue) accounts of competence (Section 2.4).

One recurring idea throughout this thesis has been the importance of *correspondences* between a system and its higher-level explanations. We have seen Clark advocate for this idea when he suggests methods such as 'cluster analysis' as promising alternative ways of explanation. It also returns in recent literature, such as in *feature-visualization* methods of explanation (e.g. Olah et al. (2017)), which establish a correspondence between features and certain neurons in a network (Section 2.6). Finally, it is reflected in Davies' definition of *tacit knowledge*: implicit knowledge which can be made explicit through a structural correspondence (Section 2.5). It seems that, when considering multi-level descriptions or explanations of a system, a possible indicator for proper integration of the multiple levels is this existence of a correspondence between them.

In the most abstract sense, this approach of grounding explanations through correspondences can be said to amount to an application of the **correspondence theory of truth**, which says that the truth of a claim lies in whether it corresponds to some underlying reality (David, 2016). While such theories are not without criticism (e.g. to establish a correspondence one first needs to agree on what constitutes 'reality', which is also difficult), it is more feasible to apply such a theory in this context, because it establishes the truth of an higher-level *explanation* only in terms of its correspondence to the low-level *implementation*, which is taken as an known truth. In other words, the black-box implementation provides the underlying 'reality' which an explanation then describes on a higher level.

Given the arguments in this thesis, I feel like the most concrete answer I can give to this question of reliability is that, in addition to the more common criteria of statistical agreement, this idea of *correspondence* is also important for giving reliable explanations of black-box systems. To give a satisfactory explanation, one should take these two factors into account, and try to establish both statistical agreement and structural correspondence between the black box and an explanatory model as strictly as the context in which the explanation is given requires.

3. What can researchers in explainable AI learn from research on cognitive architectures? One thing we have seen is that explanation is unequivocally a matter of *interaction* between systems. Viewed from the outside, there is the interaction between an explainer and an explainee. But internally, there is also interaction, at least between the black-box system which is being explained and the 'explainer system' which tries to explain it. Additionally, because explanations are multi-faceted, the explainer system must almost necessarily have some non-trivial internal structure of its own. Due to their extensiveness, the study of these interactions might at least somewhat be performed separately from the study of black-box systems and explanatory models *themselves*, on which explainable AI has mostly focused until now.

These various interactions differ with regard to the nature of the interacting agents. The interaction between explainer and explainee is a human-computer interaction, and therefore mainly poses questions for the research field of the same name. The interaction between explainer and 'explanandum' (what is being explained, i.e. the black-box system), is highly similar the same as the interaction between a scientist and its research subject, which relates it to some very difficult and long-standing issues in science and philosophy. This interaction is both scientific in the sense that one needs to rigorously study the black-box system to be able to extract any kind of explanation from it, but it is also philosophical in the sense that one needs to think about what questions are worth asking in the first place.

One can view the area of research referred to as 'explainable AI' in two ways. The first is as the field in which researchers play the role of explainer, and black-box systems are the explananda, in the same way as how biology has biologists as the explainers and organisms as the explananda. This is how the field of explainable AI generally seems to view itself. But, as I have argued, due to the parallel role of artificial and human agents, it is important to make AI explanation somewhat *autonomous*, in the same way that a human agent provides an explanation for its own behavior and does not need a researcher to investigate it (extreme cases notwithstanding). In that case, we come to the second view of explainable AI, in which computational black-box systems are the explananda and *some other* computational systems are the explainers. In this case, the role of researchers is to try and build these 'explainer systems' in such a way that they help the researchers and others understand the black-box systems. If we move from the first to the second view, we see that the interaction between explainer and explanandum transforms from a human-computer interaction to a purely computational interaction.

This is where cognitive architectures become relevant. Cognitive architectures are formally very similar to general computational architectures, in that they both describe interactions between computational systems. This means that they study the kind of interactions which play a role when considering AI explanation to consist of multiple interacting systems. However, cognitive architectures do so chiefly in the language of (semi-)human cognition, and with the aim of describing cognitive tasks, which lends them explanatory utility for the aim of describing computational agents in an intentionally-stanced manner. Common black-box models use mostly mathematical or biological language, talking about neurons and loss functions. Conversely, cognitive architectures have an inherent preference for anthropomorphic language, with terms such as 'goals', 'knowledge', 'memory' being prevalent. This kind of language in turn helps one describe a computational artificial agent using more recognizable human terms.

Just as cognitive psychology grew to a point were interactions between different processes in the brain became important, leading to a desire for 'unified theories of cognition' (Section 3.1), so is explainable AI starting to grow to a point where we have multiple ways of describing a black-box system at a higher level, leaving us with the increasingly difficult question of how to come to a cohesive explanation which convincingly explains the system's behavior. Hopefully the parallel is clear: in the same way as cognitive architectures study the interactions of cognitive systems with the environment and with each other, we can imagine a kind of 'explanatory architectures', which describe the interactions of various explanatory systems with a black-box system and with each other, in order to provide integrative explanations of a black-box system's behavior.

To close, I would like to thank my supervisor Brandt for the wonderful guidance these previous months, and my friends and family for their continual support. I hope that this thesis provides an interesting perspective on the field of explainable AI, mainly to those involved in the technical side of AI development, and gives a clear motivation for what I consider to be promising yet underexposed ways of thinking about how we (could) understand the complex computational systems we coexist with.

List of concepts

Bibliography

Achinstein, P. (2010). *Evidence, Explanation, and Realism: Essays in Philosophy of Science*. Oxford University Press.

Alvarez-Melis, D., & Jaakkola, T. S. (2018). On the Robustness of Interpretability Methods, arXiv http://arxiv.org/abs/1806.08049.

Anderson, J. (2007). *How Can the Human Mind Occur in the Physical Universe?* Oxford New York, Oxford University Press.

Bechtel, W., & Shagrir, O. (2015). The Non-Redundant Contributions of Marr's Three Levels of Analysis for Explaining Information Processing Mechanisms. *Topics in Cognitive Science*, *7*(2), 312–322. https://doi.org/10.1111/tops.12141

Bermúdez, J. L. (2005, April 28). *Philosophy of Psychology*. Taylor & Francis Ltd.

Blum, A. L., & Langley, P. (1997). Selection of relevant features and examples in machine learning. *Artificial Intelligence*, *97*(1-2), 245–271. https://doi.org/10.1016/s0004-3702(97)00063-5

Chomsky, N. (1965). *Aspects of the Theory of Syntax*. Cambridge, M.I.T. Press.

Clark, A. (1990). Connectionism, Competence, and Explanation. *The British Journal for the Philosophy of Science*, *41*(2), 195–222. https://doi.org/10.1093/bjps/41.2.195

Codella, N. C. F., Hind, M., Ramamurthy, K. N., Campbell, M., Dhurandhar, A., Varshney, K. R., Wei, D., & Mojsilovic, A. (2018). Teaching Meaningful Explanations, arXiv http://arxiv.org/abs/1805.11648v2.

Cooper, R. P., & Peebles, D. (2015). Beyond Single-Level Accounts: The Role of Cognitive Architectures in Cognitive Scientific Explanation. *Top Cogn Sci*, *7*(2), 243–258. https://doi.org/10.1111/tops.12132

Cushman, F. (2019). Rationalization is rational. *Behavioral and Brain Sciences*, *43*. https://doi.org/10.1017/s0140525x19001730

David, M. (2016). The Correspondence Theory of Truth. In E. N. Zalta (Ed.), *The Stanford Encyclopedia of Philosophy* (Fall 2016). Metaphysics Research Lab, Stanford University.

Davies, M. (1989). Connectionism, modularity, and tacit knowledge. *The British Journal for the Philosophy of Science, 40*(4), 541–555. https://doi.org/10.1093/bjps/40.4.541

De, T., Giri, P., Mevawala, A., Nemani, R., & Deo, A. (2020). Explainable AI: A Hybrid Approach to Generate Human-Interpretable Explanation for Deep Learning Prediction. *Procedia Computer Science, 168,* 40–48. https://doi.org/10.1016/j.procs.2020.02.255

Dennett, D. C. (1987). *The Intentional Stance*. MIT Press Ltd.

Doran, D., Schulz, S., & Besold, T. R. (2017). What Does Explainable AI Really Mean? A New Conceptualization of Perspectives, arXiv http://arxiv.org/abs/1710.00794v1.

Ehsan, U., Harrison, B., Chan, L., & Riedl, M. O. (2017). Rationalization: A Neural Machine Translation Approach to Generating Natural Language Explanations, arXiv http://arxiv.org/abs/1702.07826v2.

Fehling, M. R. (1993). Unified theories of cognition: modeling cognitive competence. *Artificial Intelligence, 59*(1-2), 295–328. https://doi.org/10.1016/0004-3702(93)90197-j

Goodfellow, I. (2016). NIPS 2016 Tutorial: Generative Adversarial Networks, arXiv 1701.00160.

Grill, T., & Schluter, J. (2017, August). Two convolutional neural networks for bird detection in audio signals, In *25th European Signal Processing Conference (EUSIPCO),* IEEE. https://doi.org/10.23919/eusipco.2017.8081512

Hancox-Li, L. (2020). Robustness in Machine Learning Explanations: Does It Matter? In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency,* Barcelona, Spain, Association for Computing Machinery. https://doi.org/10.1145/3351095.3372836

Hankinson, R. J. (2001, December 1). *Cause and Explanation in Ancient Greek Thought.* Oxford University.

Hélie, S., & Sun, R. (2010). Incubation, insight, and creative problem solving: A unified theory and a connectionist model. *Psychological Review, 117*(3), 994–1024. https://doi.org/10.1037/a0019532

Kotseruba, I., & Tsotsos, J. K. (2018). 40 years of cognitive architectures: Core cognitive abilities and practical applications. *Artif Intell Rev,* 1–78. https://doi.org/10.1007/s10462-018-9646-y

Kriegeskorte, N. (2008). Representational similarity analysis – connecting the branches of systems neuroscience. *Frontiers in Systems Neuroscience*. https://doi.org/10.3389/neuro.06.004.2008

Laird, J. E. (2012). *The Soar Cognitive Architecture*. Cambridge, Mass. London, England, MIT Press.

Marr, D. (1982/2010). *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. USA, W.H. Freeman/MIT Press.

Miller, G. A. (2003). The cognitive revolution: A historical perspective. *Trends in Cognitive Sciences, 7*(3), 141–144. https://doi.org/10.1016/s1364-6613(03)00029-9

Miller, T. (2019). Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence, 267,* 1–38. https://doi.org/10.1016/j.artint.2018.07.007

Mittelstadt, B., Russell, C., & Wachter, S. (2019). Explaining explanations in AI. *FAT* 2019 Proceedings, 1,* 279–288.

Montavon, G., Samek, W., & Müller, K.-R. (2018). Methods for interpreting and understanding deep neural networks. *Digital Signal Processing, 73,* 1–15. https://doi.org/10.1016/j.dsp.2017.10.011

Moravcsik, J. M. E. (1974). Competence, creativity and innateness. In J. M. E. Moravcsik (Ed.), *Logic and Philosophy for Linguists: A Book of Readings*. The Hague Atlantic Highlands, N.J, Mouton Publishers Humanities Press.

Newell, A. (1990). *Unified Theories of Cognition*. Cambridge, Mass, Harvard University Press.

Olah, C., Cammarata, N., Schubert, L., Goh, G., Petrov, M., & Carter, S. (2020). Zoom In: An Introduction to Circuits. *Distill, 5*(3). https://doi.org/10.23915/distill.00024.001

Olah, C., Mordvintsev, A., & Schubert, L. (2017). Feature Visualization. *Distill, 2*(11). https://doi.org/10.23915/distill.00007

Olah, C., Satyanarayan, A., Johnson, I., Carter, S., Schubert, L., Ye, K., & Mordvintsev, A. (2018). The Building Blocks of Interpretability. *Distill, 3*(3). https://doi.org/10.23915/distill.00010

Páez, A. (2019). The Pragmatic Turn in Explainable Artificial Intelligence (XAI). *Minds & Machines, 29*(3), 441–459. https://doi.org/10.1007/s11023-019-09502-w

Peacocke, C. (1986). Explanation in Computational Psychology: Language, Perception and Level 1.5. *Mind & Language, 1*(2), 101–123. https://doi.org/10.1111/j.1468-0017.1986.tb00321.x

Peters, P., & Ritchie, R. (1973). On the generative power of transformational grammars. *Information Sciences*, *6*, 49–83. https://doi.org/https://doi.org/10.1016/0020-0255(73)90027-3

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016, August). "Why Should I Trust You?": Explaining the Predictions of Any Classifier, In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM. https://doi.org/10.1145/2939672.2939778

Rudin, C. (2018). Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead. *Nature Machine Intelligence, Vol 1, May 2019, 206-215*, arXiv http://arxiv.org/abs/1811.10154v3.

Salvucci, D. D. (2014). ACT-R and Beyond. In S. E. F. Chipman (Ed.), *The Oxford Handbook of Cognitive Science*. Oxford University Press. https://doi.org/10.1093/oxfordhb/9780199842193.013.7

Salvucci, D. D., & Taatgen, N. A. (2010, September 1). *The Multitasking Mind*. Oxford University Press.

Smolensky, P. (1988). On the proper treatment of connectionism. *Behavioral and Brain Sciences*, *11*(1), 1–23. https://doi.org/10.1017/s0140525x00052432

Stowell, D., Stylianou, Y., Wood, M., Pamuła, H., & Glotin, H. (2018). Automatic acoustic detection of birds through deep learning: the first Bird Audio Detection challenge, arXiv http://arxiv.org/abs/1807.05812v1.

Summers, J. S. (2017). Post hoc ergo propter hoc: some benefits of rationalization. *Philosophical Explorations*, *20*(sup1), 21–36. https://doi.org/10.1080/13869795.2017.1287292

Sun, R. (1995). Robust reasoning: integrating rule-based and similarity-based reasoning. *Artificial Intelligence*, *75*(2), 241–295. https://doi.org/10.1016/0004-3702(94)00028-y

Sun, R. (2015, May). The CLARION Cognitive Architecture. In S. E. F. Chipman (Ed.), *The Oxford Handbook of Cognitive Science* (pp. 117–136). Oxford University Press. https://doi.org/10.1093/oxfordhb/9780199842193.013.11

Van Fraassen, B. C. (1988). The Pragmatic Theory of Explanation. In J. C. Pitt (Ed.), *Theories of Explanation* (pp. 135–155). Oxford University Press.

Varma, S. (2011). Criteria for the Design and Evaluation of Cognitive Architectures. *Cognitive Science*, *35*(7), 1329–1351. https://doi.org/10.1111/j.1551-6709.2011.01190.x

Varma, S. (2014). The subjective meaning of cognitive architecture: A Marrian analysis. *Front. Psychol.*, *5*. https://doi.org/10.3389/fpsyg.2014.00440

Vellinga, W.-P. (2020). Xeno-canto - Bird sounds from around the world. Xeno-canto Foundation for Nature Sounds. https://doi.org/10.15468/QV0KSN

Wang, D., Yang, Q., Abdul, A., & Lim, B. Y. (2019). Designing Theory-Driven User-Centric Explainable AI, In *Proceedings of the 2019 CHI conference on human factors in computing systems - CHI '19*, ACM Press. https://doi.org/10.1145/3290605.3300831

Weitz, K., Schiller, D., Schlagowski, R., Huber, T., & André, E. (2019). "Do you trust me?": Increasing User-Trust by Integrating Virtual Agents in Explainable AI Interaction Design, In *Proceedings of the 19th ACM international conference on intelligent virtual agents*, ACM. https://doi.org/10.1145/3308532.3329441

Woodward, J. (2019). Scientific Explanation. In E. N. Zalta (Ed.), *The Stanford Encyclopedia of Philosophy* (Winter 2019). Metaphysics Research Lab, Stanford University.

Zednik, C. (2019). Solving the Black Box Problem: A Normative Framework for Explainable Artificial Intelligence. *Philosophy & Technology*. https://doi.org/10.1007/s13347-019-00382-7

Zerilli, J., Knott, A., Maclaurin, J., & Gavaghan, C. (2018). Transparency in Algorithmic and Human Decision-Making: Is There a Double Standard? *Philosophy & Technology*, *32*(4), 661–683. https://doi.org/10.1007/s13347-018-0330-6