# CONNECTIONISM AND THE INTENTIONALITY OF THE PROGRAMMER

A Thesis

Presented to the

Faculty of

San Diego State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Arts

in

Philosophy

by

Mark R. Ressler

Spring 2003

THE UNDERSIGNED FACULTY COMMITTEE APPROVES

THE THESIS OF MARK R. RESSLER:

| | |
|---|---|
| Robert M. Francescotti, Chair<br>Department of Philosophy | Date |
| Thomas S. Weston<br>Department of Philosophy | |
| Charles F. Dicken<br>Department of Psychology | |

SAN DIEGO STATE UNIVERSITY

Spring 2003

**TABLE OF CONTENTS**

CHAPTER PAGE

## LIST OF FIGURES

# CHAPTER I

# INTRODUCTION

I started programming computers when I was 13. As soon as I had mastered the subtleties of programming BASIC on a Radio Shack TRS-80, I started working on the serious problems of artificial intelligence, which for a 13-year old meant programming games. I didn't care about mental representations or the frame problem at the time; I just wanted to write a really good game that would not bore me too quickly. After I created my first virtual world, which consisted of a few obstacles placed in a torus-shaped environment resulting from wrapping the ends of the computer screen, I started programming some basic pursuit and evasion routines. I soon found that I could always outsmart my routines, and I was quickly becoming bored.

I was also lazy. Yet Larry Wall, the inventor of the Perl programming language, claims that laziness is one of three programmer’s virtues along with impatience and hubris (Wall, Christiansen, & Schwartz, 1996, p. 609). The idea behind laziness as a virtue is that a good programmer will go to great lengths to write a reusable routine so that he will not have to code the same basic routine again and again, thus resulting in an overall time savings. My laziness also extended to expending large amounts of time thinking about the possibility of the kind of artificial intelligence I was hoping to achieve rather than spending the time doing actual coding. After all, it would not be worth the effort to develop artificial intelligence routines if they were not possible. (I had not yet acquired the programmer’s virtue of hubris.) As I traced through the requirements in my mind and planned how it would have to be done, I began to realize that there was nothing inherently impossible about such an artificial intelligence existing, but it seemed impossible that anyone could actually do it, given the complexity of the task. It would require a great many nested subroutines at such a depth that no one could keep in mind what each of the levels was doing. For me it was a question of human limitations, not machine or algorithmic limitations. So I concluded that no one would succeed in creating artificial intelligence algorithmically, and I proceeded to other projects. Of course, it was very likely that my laziness itself influenced my conclusions. If artificial intelligence were possible in principle, it would certainly require a lot of work to achieve, so perhaps the human limitations that I predicted were a projection of my own laziness on the rest of humanity.

## Weak and Strong Artificial Intelligence

Other programmers were not so lazy. Their work has yielded many practical results, such as Optical Character Recognition or OCR for converting scanned images of text pages into text. Handwriting recognition, speech recognition, and text-to-speech routines are also products of artificial intelligence research. In the realm of games, the current range of real-time strategy games and first person adventure games wildly exceed my own youthful hopes.

Yet do any of these achievements really represent artificial intelligence? John Searle’s distinction between weak and strong artificial intelligence is helpful in evaluating these results:

> According to weak AI, the principal value of the computer in the study of the mind is that it gives us a very powerful tool. . . . But according to strong AI, the computer is not merely a tool in the study of the mind; rather the appropriately programmed computer really *is* a mind, in the sense that computers given the right programs can be literally said to *understand* and have other cognitive states. (1990, p. 67)

For strong artificial intelligence, intelligence lies within the computer program. “The programs are themselves the explanations” (p. 67) of cognitive states. What is significant about the program is not the language in which it was written or the kind of machine on which it is run, but rather the syntactical relations between the elements within the program, which correspond directly to elements within naturally occurring cognitive states.

Surely no one thinks that humans obey the same pursuit and evasion routines as computer opponents in computer games, which would be the claim of strong artificial intelligence in this case. It might be claimed, though, that these routines do demonstrate weak artificial intelligence, since the way in which these routines succeed provides some insight into the way humans themselves behave. Different behavior routines can be implemented and tested in the games, and the ones that do not succeed show that those routines represent poor models of human intelligence. If the routines that do not succeed are poor models, then by contrast, the ones that do succeed must be good models.

This argument seems to be a version of an argument from coincidence: If an artificial intelligence program produces intelligent behavior, it would be coincidence if the logic in the program were not at least part of human intelligence. Since such a coincidence is unlikely, the routines must provide a tool for understanding human intelligence itself. Of course, the weakness of this form of argument from coincidence is readily apparent, since it could just be a coincidence that the logical routine simulates human behavior, and this sort of coincidence may in fact be very likely.

I will put aside such indirect arguments, whether there is any merit in them or not, since I find it a more interesting question whether artificial intelligence programmers can code human intelligence directly. This question relates primarily to strong artificial intelligence in which the claim would be that a particular program does in fact demonstrate intelligence, because it was coded based directly on how human intelligence works. I will also refrain from a review of the various candidates for artificial intelligence, since I do not think that I can add anything pertinent to what has already been said (e.g., Dreyfus, 1997). Before I outline the main problem I will address in this thesis, I will mention two of the more prominent criticisms of strong artificial intelligence, namely those offered by Hubert Dreyfus and John Searle. My own argumentative strategy will emerge in the context of these two criticisms.

## Two Objections to Strong Artificial Intelligence

Forming his criticism against the theoretical background of Husserl, Heidegger, and Wittgenstein, Dreyfus notes that "[artificial intelligence] researchers have implicitly tried to treat the broadest context or background [of intelligence] as an object with its own set of preselected descriptive features" (1997, p. 179). But this strategy treats the world as an object within intelligence whereas intelligence properly appears within the world, which determines what it means to be an intelligent creature. Dreyfus's central criticism of artificial intelligence is that any explicit representation of the world sufficient for intelligence will never be complete. He cites Husserl's later concession that such an analysis forms an "infinite task." "Husserl thinks of intelligence as a context-determined, goal-directed activity" (p. 160). In order to create artificial intelligence, the programmer must uncover each context in order to code that context into a program. Yet each context itself arises within a further context, so the analysis will never seem to end. Eventually the analysis must fall back on the brute fact of the being of the intelligent creature, in which all contexts and goals are already present, which is something that cannot be formulated in a series of logical instructions (p. 180). Any attempt to formulate such instructions will always be incomplete in some critical way. It does not help the situation that artificial intelligence researchers have attempted to reduce the world in which artificial intelligence appears to “micro-worlds” in which the relevant features of the world are restricted to a more manageable set (p. 146). The completeness of the task must ultimately encompass the being of the intelligent creature, which cannot be represented algorithmically within the creature’s own intelligence. Rather, the being of the intelligent creature forms the basis of its intelligence, so any putative representation of that being within its intelligence would seem to be incomplete. At the very least, if the being of a creature could form the basis for the intelligent representation of its being, that representation does not seem to include the being that includes that same representation. This line of argument suggests Gödel’s Incompleteness Theorem, but Dreyfus’s argument is founded on the phenomenology of Husserl and Heidegger, rather than on the interplay of completeness and consistency within human intelligence. The key point of Dreyfus’s objection is that all of the contexts that make intelligence possible can never be identified such that they can be formulated into algorithmic steps.

Searle’s criticism stems from his notorious “Chinese Room” thought experiment (1990), in which he imagines that a person is placed in a sealed room with minimal communication with the outside world. People outside the room send messages written in Chinese to the person inside, but that person does not understand any Chinese. Searle then supposes that the process of understanding Chinese could be formulated into a set of manipulative rules, which are available to the person in the room. When messages are sent into the room, the person uses the rules to formulate a response, which is sent to the people outside. The person understands no Chinese, so these instructions are written in the person’s native language, for example, English. The responses are perfectly intelligent, such that the people outside are convinced that whatever is in the room understands Chinese. Searle then asks whether there is really any understanding of Chinese in the responses. The person in the room does not understand Chinese, and Searle argues that there is nothing else involved in the response that does. The resulting criticism of artificial intelligence arising from this thought experiment is that a set of manipulative rules cannot represent true understanding of anything. Even if the artificial intelligence program does convincingly replicate aspects of human intelligence, it cannot properly be said to understand what it is doing solely on the basis of the formal symbolic manipulation it performs in the execution of the program. If the program participates in a Turing test, it doesn't necessarily mean anything it says, since meaning cannot be coded into rules. Rather, meaning requires a set of capacities that operate in the background by virtue of which any mental activity means anything (1983, pp. 141-159; 1992, pp. 175-196). The relations of mental representations are what Searle calls the 'Network', which seems to have a structure capable of being coded into algorithms. Such a Network is what artificial intelligence workers use in their attempts to code intelligence into a program. However, the Network itself has meaning only with reference to a set of non-representational basic capacities and practices, such as seeing or eating, which Searle calls the 'Background'. Searle claims that this Background itself is something that cannot be coded into rules (1992, p. 193).

I will not examine these criticisms in great detail, but will only briefly compare them. First, both criticisms claim that strong artificial intelligence will never exist, whereas my lazy youthful analysis supposed that it could exist, but could never be created due to human limitations. Dreyfus claims that artificial intelligence can never capture all of the relevant aspects of the world in which it needs to act intelligently, whereas Searle argues that even if it does, such putative intelligence lacks meaning and therefore is not really intelligent. Second, both arguments link intelligence to a broader background, namely the world that intelligence must represent in Dreyfus's argument, and the capacities that give intelligence meaning in Searle's argument. The issue is whether the background can be taken as an object such that it can be coded into artificial intelligence. In Dreyfus's argument, the attempt to do so will inevitably fall back on the basic question of being, which links all contexts and goals, and which cannot be formulated into rules. Comparably, in Searle's argument, the attempt to formulate the Background into rules will fail because those capacities and practices that form the Background are themselves not necessarily representational.

## Argumentative Strategy

The question of the ability to take the background as an object relates to the philosophical principle of Intentionality, which is the feature of mental states that they are about something. If I want a piece of bacon, my desire exhibits Intentionality in that it is about something, namely a piece of bacon. Like Dreyfus and Searle, I think that the notion of Intentionality poses a problem for artificial intelligence. Rather than focusing on the background that intelligence requires, I will focus on the foreground, not of any putative instance of artificial intelligence, but on the act of creating an artificial intelligence program. In this thesis, I adopt a more modest goal than Dreyfus, Searle, or even myself in my earlier youthful evaluation of artificial intelligence. I will not argue whether artificial intelligence can ever be achieved, but will simply outline the specific problems that Intentionality raises for artificial intelligence in the very act of attempting to create an artificial intelligence system, specifically from the programmer’s point of view. Intentionality is a factor in artificial intelligence in two ways: (1) if the goal of artificial intelligence is achieved, then the resulting system should exhibit Intentionality, and (2) this result will be achieved by a programmer whose own Intentionality plays some role. This thesis will examine the relationship of these two aspects of Intentionality in artificial intelligence. The programmer’s point of view is the key factor, in my opinion, since the programmer is the agent whereby artificial intelligence is achieved or not. The tone of this thesis is primarily skeptical, in that certain claims have been made concerning both strong artificial intelligence and connectionism, but the involvement of the Intentionality of the programmer seems to offer reasons for doubting these claims. Perhaps these claims can ultimately be vindicated in an unmodified form, but first these doubts will need to be removed by eliminating the Intentionality of the programmer as a factor in the attribution of intrinsic Intentionality to the resulting artificial intelligence system.

There are a number of issues within the philosophy of mind that can be raised with regard to artificial intelligence, such as consciousness or qualia, namely the subjective nature of conscious experience. I will ignore the questions of whether artificial intelligence systems can be conscious, whether they can instantiate qualia, whether consciousness and qualia can be represented in artificial intelligence, or whether consciousness and qualia can emerge from artificial intelligence systems. Rather, I will focus on the cognitive aspects of artificial intelligence. While it is true that I am thereby ignoring the more interesting questions in the philosophy of mind as related to artificial intelligence, I think there is merit to pursuing my more modest goals as a prerequisite for an examination of these questions. I think it is pointless to argue whether a tower should have a flag on top when it is questionable whether the tower can even be built on the designated foundations. Even if it is argued that consciousness is essential to Intentionality, that the tower is not a tower without the flag, then my preliminary investigations on Intentionality in artificial intelligence should provide groundwork for evaluating that claim.

The form of artificial intelligence that I have been discussing so far assumes that intelligence can be coded into a set of rules or algorithms and is therefore strong artificial intelligence. I will follow the traditional designation and refer to this form as *classical artificial intelligence*. A second form of artificial intelligence seeks not to model the acts of intelligence themselves, but rather seeks to model an artificial mind in which intelligence will arise. This form is known as *connectionism*, since it posits that mental representations arise in the connections between nodes in a cognitive system, such as neurons in the brain. My primary concern is with connectionism, not only since it is now more actively pursued than classical artificial intelligence, but also since my own research interests lie in connectionism, and therefore I am using this thesis as a prolegomena to my future research. Yet I will still discuss the problems of Intentionality for classical artificial intelligence, primarily to put the problems for connectionism into stronger contrast.

In Chapter II, I discuss the nature of Intentionality and the traditional philosophical problems that have been raised with regard to Intentionality. In Chapter III, I outline four requirements for classical artificial intelligence related to Intentionality, and discuss the degree to which classical artificial intelligence can meet these requirements. I offer an overview of connectionism in Chapter IV, its relation to neural networks, and the forms of neural networks that could comprise a connectionist system. Chapter V contains the heart of my argument, identifying the aspects of Intentionality that cause problems for the connectionist programmer comparable to the problems for the classical artificial intelligence programmer discussed in Chapter III.  Finally, I briefly outline what is essentially a research project in Chapter VI that attempts to overcome the problems of Intentionality.

# CHAPTER II

# INTENTIONALITY

*Intentionality* is a technical philosophical concept meaning the feature of mental states that they are directed at something. If I desire a piece of bacon, my desire is directed at a piece of bacon. Intentionality can also be understood as mental states being about something, which is known as an Intentional object. Thus my desire for a piece of bacon is about the piece of bacon. It is difficult to imagine any sort of desire that is not about something. To say that I just desire, but don't desire something, is really to say that I am feeling a certain way, perhaps empty inside. It does not properly express a desire, which is always a desire for something. A number of mental states besides desires likewise exhibit Intentionality, for instance, beliefs, hopes, and wishes, all of which can be understood as Intentional states.

Right away, I need to distinguish the concept of Intentionality from the common notion of intentions and from another technical philosophical concept, *intensionality*. I will use John Searle's useful convention of capitalizing the philosophical term, 'Intentionality' to distinguish it from the ordinary term 'intentions'. This convention will also help somewhat to distinguish Intentionality from intensionality, which otherwise would have been differentiated from each other only by a 't' and an 's' in their spellings.

Intentions relate to volition. If I intend to fry a pound of bacon, I am commonly understood to be making a conscious decision to fry that pound of bacon as opposed to tossing it raw into a blender or doing something completely different. Thus intentions relate to the will, whatever the will may be and whether that will is free or not. Besides these interesting philosophical questions, intentions remain a recognizable sort of mental state, separate from desiring, hoping, or wishing. I can desire a pound of fried bacon. I can hope that a pound of bacon will be fried. I can wish that a pound of fried bacon be in front of me. However, none of these mental states is the same as intending to fry a pound of bacon, which, while not entailing action on my part, at least implies my future action of frying bacon.

The relationship between intentions and Intentionality, according to Searle, is that "intending to do something is just one kind of Intentionality among others" (1983, p. 3). Intentions therefore are directed at or are about something, namely the action that is intended. In the case of my intention to fry a pound of bacon, the Intentional object is the act of frying a pound of bacon. Consequently, the common notion of intentions fits within the broader scope of Intentionality, at least with regard to the relationship to an Intentional object. Though it is a potential issue within artificial intelligence, the relationship of the intention to the eventual action itself that may or may not occur will not concern me here, since my concern is with the Intentionality of the programmer in creating artificial intelligence.

The philosophical concept of intensionality is related to the Fregean distinction between sense and reference (Frege, 1997a), which corresponds to the notions of intension and extension, respectively. The distinction likewise corresponds roughly to the ideas of connotation and denotation. In analyzing the meaning of a term, such as 'bacon', the extension of the term is that to which the term refers, namely the bacon itself. The intension of the term is the meaning of the term, which in the case of 'bacon' does not seem to offer much distinction between its extension or reference. The contrast becomes clearer in analyzing a phrase such as ‘slippery slices of porcine flesh’, which has the same extension as ‘bacon’, namely the bacon itself.  However, the senses of the two are different, since ‘bacon’ is a fairly simple designator term, whereas ‘slippery slices of porcine flesh’ expounds on the meaning of bacon by explicitly referencing the source of the bacon, and adds a somewhat poetic description of the end result.

Extensionality and intensionality are considered properties of statements.  A statement is extensional if its terms can be replaced by other terms with the same extension while maintaining the truth of the statement.  For example, “Mark fried a pound of bacon” is extensional, since its truth is preserved when replacing ‘bacon’ with ‘slippery slices of porcine flesh’.  However, the statement “Mark believes that bacon is edible” is not extensional, since the substitution does not necessarily preserve the truth of the statement, for example, if Mark did not know what ‘porcine flesh’ meant.  Such a statement exhibits intensionality.

According to Searle, the only relation between intensionality and Intentionality is that some statements concerning Intentional states exhibit intensionality (1983, p. 24).  The notions themselves are different, though Intentionality seems to help explain the intensionality of statements that ascribe intentional states.  For my purposes, intensionality has no immediate bearing on artificial intelligence, since it is not clear that an artificial intelligence programmer must specifically create a system that is capable of generating intensional statements.  Rather, the statements resulting from an artificially intelligent system will usually exhibit intensionality or extensionality without any specific intention to produce that result.

## Nature of Intentionality

It seems somewhat odd and even annoying that there should be three terms so similar yet with such different meanings, but this is a concern only in philosophical circles where the two technical terms of Intentionality and intensionality appear. The current usage of Intentionality derives from Franz Brentano, who considered Intentionality as a distinguishing feature of all mental phenomena from physical phenomena. “Every mental phenomenon includes something as object within itself, although they do not all do so in the same way” (1973, p. 88). On this view, if something is a mental state, then it must exhibit Intentionality.

Searle disagrees that every mental state is Intentional, and he offers as an example the fact that “there are forms of elation, depression, and anxiety where one is simply elated, depressed, or anxious without being elated, depressed, or anxious about anything” (1983, p. 2). However, mental states such as belief must be Intentional states, since I cannot simply be in a state of belief without believing something. For Searle, “every Intentional state consists of an *Intentional content* in a *psychological mode* [italics in original]” (p. 12). The psychological mode is believing, hoping, wishing, desiring, and so forth. The Intentional content can either be an Intentional object such as bacon, to which my desire is directed, or an action such as frying bacon, to which my intention is directed, or a proposition such as that bacon is good, to which my belief is directed. In the latter case, Searle insists on the distinction between the Intentional content and an Intentional object. If I believe that bacon is good, the Intentional object of my belief is not the proposition that forms its content, but rather the bacon that is the subject of the proposition. Desires or actions or propositions themselves may be taken as Intentional objects, as when I self-consciously think about what I am desiring, doing, or proposing. But for Searle, in the case of beliefs, the proposition is merely the content of the Intentional state, whereas the Intentional object is the subject of the proposition (pp. 18-19).

I will generally follow Searle in his analysis of Intentionality at least as far as these foundational principles, though I do not necessarily endorse his further conclusions about Intentionality, language, and the mind[1]. What is important for my discussion is simply that some mental states are directed to Intentional objects or content. I intend to show that this feature presents an important problem for artificial intelligence.

## Problems of Intentionality

Before I begin to outline this problem, by contrast I will briefly mention some of the traditional problems of Intentionality, without reviewing any of the proposed solutions. First, there is a question concerning the ontological status of Intentional objects. If I believe that I like bacon, the Intentional object of my belief, according to Searle, is myself, and so there does not seem to be any additional ontological problem of the status of that Intentional object above the ontological status of the self. However, if I desire bacon, the Intentional object of that desire is bacon, and the ontology of that bacon becomes a question. I don't desire the universal form of bacon, unless I am a Platonist or otherwise a realist about universals who explicitly desires that universal form. I desire some specific bacon that I will eat. Yet the specific piece of bacon that I will eat is not always known to me at the time of my desire. Furthermore, my desire may never be satisfied, since it may pass without my ever having

[1] For example, I am undecided about Searle's distinction between the content and the object in the Intentional state of belief that I just outlined. Perhaps in the Intentional representation of Intentional states, this distinction can be made, but I have a deep suspicion concerning the ontology of Intentionality. Perhaps the connectionist study of Intentionality could help decide such an issue. Furthermore, though I am sympathetic to Searle's view on the irreducibility of the first person ontology of consciousness, I take a more radical view on consciousness such that any conclusions about consciousness derived from his Chinese Room thought experiment can be undermined.

eaten any bacon. If my desire had been satisfied, the bacon that ultimately satisfied my desire could be understood as the Intentional object of my desire; yet if my desire remains unsatisfied, the Intentional object of my desire remains undetermined. In such a case, it becomes difficult to say exactly what the status of the bacon that I desire is, given that there is no particular bacon corresponding to my desire. The situation is worse with respect to Intentional objects that do not exist or are impossible. If I want my tongue to be composed of bacon such that I taste bacon continually, the Intentional object of my desire is apparently impossible, since bacon lacks the taste sensors required for a human tongue, and even if it were possible to create a tongue from bacon (unfried, I assume), there is no reason to think that such a tongue would taste itself. In such a case, the ontology of that Intentional object becomes a problem. This problem does not directly concern me here, since it will remain a problem in artificial intelligence as it is in human intelligence.

Second, there is question concerning the relationship of Intentional states to consciousness. Are all Intentional states conscious, and does consciousness require Intentional states? This is potentially an issue for artificial intelligence, if consciousness is necessary for intelligence. However, I am considering consciousness to be an additional question on top of artificial intelligence. If an artificial system is not considered intelligent only because it lacks consciousness, then what it does achieve without consciousness can be considered a form of intelligence that would be a significant accomplishment in its own right.

Third, there is a question concerning the relationship of Intentionality to representation. This is also a concern for artificial intelligence, particularly in evaluating the debate between classical artificial intelligence and connectionism. The issue of representations is first whether there are internal representations, and second whether these representations can themselves be formulated representationally.  In terms of Intentionality, the first issue seems to dredge up aspects of the ontological question of Intentional objects and asks whether internal representations can be dispensed with altogether in designing artificially intelligent systems.  The second issue concerns the implementation of Intentionality within an artificial system, whether it can be achieved using the algorithmic approach of classical artificial intelligence, or whether a connectionist system must be used. There is a related issue whether connectionist systems really are just representational systems in disguise.  These issues will be discussed again in Chapter IV on connectionism, though I will not analyze the issues in depth there.  These questions concern the posited end result of artificial intelligence, namely the artificially intelligent system and what it must be like in relation to human Intentionality.  As I stated in the introduction, my approach to artificial intelligence here is from the point of view of the programmer attempting to implement artificial intelligence, which presents some interesting problems that do not seem to have been explored before, given the focus on the end result.

An artificially intelligent system should exhibit Intentionality, if for example it is claimed that the system believes something.  How then will this be achieved?  The programmer could proceed on a trial-and-error basis, first creating a system and then checking whether it has Intentionality, assuming that there are criteria for Intentionality that can be applied to a mechanical system.  Both conceptually and in practice, this approach seems foolish.  I would not create a word processing program and check to see whether it has Intentionality.  If my goal is artificial intelligence, I will proceed by creating an artificial system that I think will result in intelligence.  In other words, I would intentionally create the Intentionality as a feature of the system, whether I think of it in these terms or not.  Since as

Searle notes, the intention of the programmer is itself an Intentional state, the Intentionality of the artificially intelligent system is bound to the Intentionality of the programmer. For the purposes of this thesis, I am assuming that the programmer successfully and intentionally creates an artificially intelligent system according to the model of strong artificial intelligence, namely classical artificial intelligence. It is possible that the programmer may succeed accidentally in achieving artificial intelligence, if for example, a “bug” in the program turns out to provide a key factor that the programmer missed in his intentional design. Yet since strong artificial intelligence requires that the intelligence of the system arise precisely by means of the syntactical relations between representations, such a bug could have been included within the intentional design. If, however, the accidental intelligence arises by virtue of some accidental connection between the hardware and the software, then the intelligence is not according to strong artificial intelligence, since it arises not solely by virtue of the syntactical relations. The link between the Intentionality of the system to the Intentionality of the programmer is necessitated in strong artificial intelligence by the consideration that it is the programmer who designs and implements the syntactical relations within the program that exhibits Intentionality. The next chapter examines in detail how this interplay of Intentionality works in the case of classical artificial intelligence, and what is required for it to succeed from the programmer’s point of view.

# CHAPTER III

# INTENTIONAL REQUIREMENTS FOR CLASSICAL ARTIFICIAL INTELLIGENCE

Assume that I as a programmer intend to write an artificial intelligence program. What is required to accomplish this? Certainly there are a number of background practices and abilities with which I must have a level of competence, primarily competence in at least one programming language. This presupposes that I have some means of implementing that programming language, for instance, that I know how to use a computer keyboard and a computer operating system and program compilers. These background abilities are not of primary interest to me, so I will grant them all. More importantly for my purposes here, my intention to write a program demonstrates that I have Intentionality, so if Intentionality is a requirement, it is already satisfied. The specific question I will examine here is what is required of that Intentionality in order for me to succeed in writing an artificial intelligence program.

I assume of course the possibility of writing such a program. Even if no putative classical artificial intelligence program can possibly achieve genuine intelligence according to the model of strong artificial intelligence, if intelligence cannot possibly be reduced to algorithmic steps, I can still examine the requirements for the creation of such a program. Even if I as a programmer will never succeed, my intention to create such a program and my attempt to do so can still be analyzed. If such strong artificial intelligence is truly impossible, then my examination could reveal some critical step in the creation process that fails. I do not think that I have identified such a critical failure, but what gets revealed in this examination becomes significant for my later examination of the claims of connectionism with regard to Intentionality. Therefore I think that the effort spent in the examination of classical artificial intelligence is valuable, even if I do not uncover a new argument against classical artificial intelligence, which is not my goal in any case.

Consequently, I will outline four requirements of the programmer with regard to Intentionality. Briefly, they are: (1) Introspective access to Intentional states, (2) Detailed knowledge of Intentional states, (3) Ability to understand Intentionality as an Intentional object, and (4) Ability to project that Intentionality.

## Requirement 1: Introspective Access to Intentional States

Suppose that I write a computer program that results in the following conversation between the program and a human:

```
Human: What do you think about current affairs?

Program: I think that President George W. Bush is
making critical errors in his aggressive foreign
policy.

Human: Why?

Program: Because the increase in aggression will
merely foster further terrorism.
```

This program may appear to demonstrate considerable intelligence, since it apparently understands the notion of current affairs and can explain its prior assertions. However, suppose that this program consisted entirely of the following pseudo-code[2]:

[2] Pseudo-code is a description of algorithmic steps to be implemented in a computer program, but not necessarily in any particular computer language. My pseudo-code typically looks like code in the BASIC computer language, since that is the first computer language I learned, corrupted by subsequent work in the Java and Perl computer languages.

```
If Question = “What do you think about current
affairs?” Then

  Print “I think that President George W. Bush is
  making critical errors in his aggressive foreign
  policy.”

End If

If Question = “Why?” Then

  Print “Because the increase in aggression will
  merely foster further terrorism.”

End If

If Question = “Do you like bacon?” Then

  Print “Yes, very much so.”

End If
```

This program would also result in the following conversation:

```
Human: Do you like bacon?

Program: Yes, very much so.

Human: Why?

Program: Because the increase in aggression will
merely foster further terrorism.
```

The program participating within this conversation is not very intelligent at all[3]. The program outlined in pseudo-code is an example of what is known as a “canned program,” namely a program that merely offers pre-defined responses to pre-defined questions. If the question posed does not exactly match any of the pre-defined questions, if I posed the

[3] There seems to be a common ontological imprecision concerning the bearer of intelligence in discussing classical artificial intelligence. As a philosopher, I would say in a strict ontological sense that the bearer of intelligence is the system running the program. Since classical artificial intelligence posits intelligence by virtue of the algorithmic steps within the program, and since that same program could run on different systems and exhibit the “same” intelligence (ignoring the philosophical distinction of type and token), it seems common usage to attribute intelligence to the program. In discussing classical artificial intelligence at this point, I will employ the imprecise usage, since it seems more natural.

question “Do you enjoy bacon?” for example, then the program will not offer any response at all, or may offer a default response such as “I don’t understand what you mean,” if it has been suitably programmed. The canned program outlined above does not even have sufficient intelligence to recognize that “Do you like bacon?” and “Do you enjoy bacon?” are really expressing the same question.

How do I know that this program is not intelligent? I suggest two possible criteria that lead to two possible approaches to the creation of artificial intelligence programs: a third person and a first person criterion. According to the third person criterion, I observe the program in action, without knowing any of the details of its inner programming logic, and judge the program to be intelligent or not. This is essentially the Turing test, which was demonstrated in the two sample conversations above. I didn’t need to know what programming steps were contained within the program to make my evaluation, since the request to clarify the program’s claim that it liked bacon produced an explanation that did not follow from its context, which is not very intelligent.

The first person criterion does require knowledge of the inner program logic. According to this criterion, I compare the program logic to my own introspective analysis of my intelligence and judge whether the program is doing what I do or not. Consequently, I see that the canned program is merely offering pre-defined responses, which is not what I do when I am asked a question, and I observe this in the instructions contained within the program logic, not merely the linguistic behavior as in the third person criterion. In my case, I recognize the meaning of a question, not merely the exact words, and generate a response according to that meaning. The canned program does not do that, so it is not intelligent.

Each of these criteria has conceptual problems associated with it. The first person criterion takes a particular token of intelligence, namely my own, as a standard for comparison, such that differences between individual intelligences may not be properly acknowledged. For example, I may judge a particular program to be unintelligent when compared to my intelligent behavior as revealed through my introspective analysis of what I am doing when I behave intelligently, whereas someone else may judge it to be intelligent according to his introspective analysis. In such a case, there is a difference in comparison to two token intelligences. This difference seems to require inter-subjective agreement to resolve the question of the intelligence of the program. In comparing my introspective analysis with the other person's, I may find that my analysis was incorrect, and that the other person's analysis more correctly represents what I myself do internally in producing intelligent behavior. Consequently, I could then agree that the program was intelligent. This example argues against the infallibility of introspection, which is a significant topic in its own right (see Lyons, 1986). Since inter-subjective comparison of introspective analysis demonstrated the problem, perhaps the third person criterion is preferable to the first person.

Yet the third person criterion has its own problems, for instance, the possibility that a critical observation may not be made prior to the judgment. If everything I observe about the program seems intelligent, that does not guarantee that in the next observation of the program it will also seem intelligent. Perhaps the programmer has created a program that seems intelligent in most circumstances, but there are other circumstances in which it does not. To make the evaluation of intelligence properly, it seems that I would need to see the program in all possible circumstances, which is impractical if not impossible. This problem is partly the problem of induction in which the inference from particulars to universals is questioned, but it is also a problem of finding a critical third person test to distinguish between genuine and counterfeit intelligence.  It seems that any putative critical test formulated precisely enough will entail results that may be counterfeited by a programmer who designs a program specifically to give such affirmative results, given knowledge of the critical test.  On the other hand, any putative critical test formulated sufficiently flexible to detect such counterfeits would rely upon human judgment to make the arbitration, and humans are not infallible in their judgment, even in inter-subjective judgments.

Why is this not also a problem in judging the intelligence of humans?  It does not require observation in all possible circumstances in order to make such a judgment.  I think that it is not a problem in this case since the general intelligence of the type has been accepted.  The capacity for rationality is typically considered a defining attribute of humanity.  As such, any instance of humanity is easily granted intelligence given fairly minimal evidence, for instance, speech in a recognizable language.  In the case of a computer program, the intelligence of the type is what is precisely at issue, and therefore, the evidence required is much stronger.  If a human is caught in unintelligent behavior, that exception does not invalidate the intelligence of the type; however, a single counterexample in the case of putative computer intelligence does seem to question whether the intelligence of the type has been established.   There doesn't seem to be an uncontroversial critical mass of putatively intelligent behavior identified for artificial intelligence in order to establish the intelligence of the type, such that these counterexamples can be dismissed as mere exceptions.  Perhaps this circumstance seems unfair or unjustified, but perhaps that should lead me to question my judgment of human intelligence, rather than to accept a looser standard for accepting artificial intelligence.

I think that the reason for this difference in standards is that the third person criterion itself seems to be dependent upon the first person to some extent. In a similar way to Hume’s attack on induction, I could argue that observation of behavior alone cannot lead to any judgment of intelligence. Rather it would lead to a mere collection of perceived actions presented in a certain order. This collection may be recognized as intelligent by inter-subjective agreement, for instance if I observed those actions and was told that they are intelligent. However, the problem of particulars and universals arises again, since the ostensive attribution of intelligence to one token series of actions does not necessarily extend to another token series of actions. Rather it requires the recognition of the type of action such that other tokens can be subsumed under the type. Such recognition, at least, takes place from the first person point of view. Consequently, it seems that any criteria of intelligence must entail both third person and first person aspects to succeed, since a first person criterion must be validated against third person inter-subjective agreement, and a third person criterion relies on first person recognition. This combination makes the judgment of human intelligence much more natural, since the inter-subjective agreement that forms part of the criterion itself involves the presumption of intelligence in humans such that inter-subjective agreement can be possible. Of course, this does not strictly justify the difference in standards in judging intelligence between humans and machines, but it does suggest a reason why the difference occurs. Perhaps this line of inquiry should be pursued further, but it would take me too far from my topic of concern in this thesis.

I therefore turn from considerations of judgment of intelligence to considerations of creation. Taken separately, the first person and third person criteria suggest two different approaches to creating an artificial intelligence program. According to the first person approach, I could perform an analysis of my Intentional states and attempt to code those states algorithmically into a program. According to the third person approach, I could analyze instances of intelligent behavior and replicate the behavior within my program.

The third person approach seems compatible with writing the canned program outlined earlier, since the canned program merely aims to produce some behavior that is intelligent. In such a case, the structure of a canned program seems appropriate. If certain stimulus is given, the program should output some canned response. The intelligence of the response is sufficient for the intelligence of the system as a whole. However, the task is complicated for the same reasons that the canned program was not considered intelligent, namely that the same stimulus may produce different behavior in different contexts. Therefore, the programmer must take context into account within the program, and therefore the program size and complexity will need to increase to accommodate the required contexts. To some extent, this context can be determined according to third person observation. Perhaps a particular stimulus always elicits the same response when issued within a restaurant as opposed to in a park. Therefore, if in a restaurant, give response A to stimulus S; if in a park, give response B. However, not all contexts reduce to something like a place. Consider the context of knowing that someone is lying. If the liar were to elicit a response from the program based on the lie, in order to behave intelligently, the program needs to account not only for the discrepancy between the liar's statement and the observable facts, but also the fact of the lie. The program could merely respond "You are mistaken about the facts" to each of the liar's statements, but it should also recognize not only the pattern of factual discrepancies, but also the difference between intentional lies and inadvertent mistakes. Of course, this is also a problem for humans determining whether other humans are lying or are merely mistaken. My point is not that this distinction cannot be determined infallibly based on some third person evidence, but rather that whether the distinction is made infallibly or not, it must be made on some inner logic that cannot easily be devised directly from third party observation.

This logic might be modeled on observations of a particular person, and careful observations may identify certain signs the person looks for in another to determine whether that other person is lying; however, the recognition that the person is identifying signs of lying goes beyond direct observation. There is an inference from the observation that the person looked at the other person's palms to the judgment that the person was checking to see whether the other person's palms were sweating. It seems to me that the reason this particular inference was made, as opposed to an inference that the person was checking whether the other person had "I am a liar" written on his palms, is that my observations are correlated with my introspective awareness of what it means to be a liar and the effects of lying on a human body. This introspective awareness is mediated by memory, since I do not need to be lying myself in order to judge that someone else is lying. This memory has Intentional content, most likely in the form of a proposition, such as "When I lie, my palms sweat." This proposition in turn relies on the recognition of the state of lying, which seems to me to depend on first person awareness of that state. The proposition remembered could be in a universal form, such as "A lying human typically has sweaty palms," but I would argue that the notion of lying in this case still seems to involve first person awareness of that state, which is subsequently applied to the third person observation concerning sweaty palms, since the meaning of the term 'lying' is dependent both upon first person awareness and inter-subjective agreement. So it seems that the conclusions of my observations themselves are dependent on some first person awareness. If I had no such awareness of lying, then I would never make the inference that someone checking another person's palms is checking for a sign of lying. I might simply be confused by his behavior.

Therefore it seems that introspective awareness is a requirement for creating an artificial intelligence program, whether it appears as a factor in third party observations or whether it is used directly in a first person analysis of my Intentional states. Of course, such a first person analysis is not free from third person influence. For instance, even if I did not know about lying until I myself lied, I would not know that liars' palms sometimes sweat unless I had noticed my own palms sweating, which is a third person observation of my own body. Certainly the designation of such behavior of mine as "lying" is made through inter-subjective agreement, since the correlation of a term to a particular type of behavior within a language requires the agreement of all speakers of that language. Consequently, both first person introspection and third person observation are required.

I singled out introspection over observation as a requirement for implementing artificial intelligence, since in the act of writing a program introspection is immediately used. Observation is still present in the form of memory, but introspection provides the critical bridge between remembered observations of intelligent behavior and the intended intelligent program. I do not keep some example of intelligent behavior in front of me as I write the program; I do not have a liar continually lying in front of me as I program. Rather, I remember instances of lying, whether my own lying or someone else lying, and introspectively analyze the structure of that lying with regard to its corresponding behavior and take that lying as an Intentional object to be implemented within my program.

The problem of incorrigibility of introspection is not important in this requirement, since it does not matter whether I correctly recognize whether I am lying at any particular moment. It is sufficient for this requirement only that I recognize what it means to lie, even if I personally have never lied in my life. This recognition is subject to questions of incorrigibility, though, since I may not correctly recognize that I am recognizing what it means to lie. However, this recognition itself is not the point of the introspection in this case. The point is that the introspection is a precondition for the further step of programming the results of the introspection. If that recognition were faulty in some way, the fault would be transferred forward into the artificial intelligence program. Therefore, if incorrigibility of introspection is not an issue as I have claimed, infallibility most certainly is.

The notion of introspection has fallen on hard times as of late (see Lyons, 1986). I do not mean to support or rely on any particular notion of introspection in my argument, but I do think that there is a distinction underlying statements of the form “I just had a strange feeling” and “I just saw aliens eating raw bacon”. Whatever underlies a statement of the first form is what I understand as introspection. Perhaps mere consciousness suffices for this, but my views on consciousness would require expanding the scope of this thesis to address this point. Therefore, I leave the notion of introspection largely unanalyzed. My point in this requirement is that a different mode of access than third person observation seems necessary for the programmer in order to achieve Intentionality within an artificial intelligence program, whatever that introspective mode of access turns out to be upon further analysis.

## Requirement 2: Detailed Knowledge of Intentional States

In order for me to understand what it means to have a particular Intentional state, I must have correctly recognized that I am in that Intentional state at least at one point in time.

Suppose that someone else were to explain to me what a particular Intentional state was like, such as believing. What I would get from that person would be a set of sufficient conditions for believing, but I claim that such sufficient conditions are not necessarily sufficient for a full understanding of that Intentional state, though they will enable me to recognize that what I am experiencing at a particular time is believing, and that recognition will enable full understanding. If the sufficient conditions that are given do in fact contain all the necessary conditions, introspection will make this clear in a way that the sufficient conditions themselves will not. Note that this third person verification in receiving sufficient conditions from others is needed to overcome the fallibility of introspection. If I think I believe that bacon is delicious, someone else can verify that I can distinguish bacon from a rotting fish wrapped in newspaper, and that whenever I eat a piece of bacon, I exhibit certain signs of pleasure, and therefore that I have correctly recognized the belief. Consequently, this aspect of fallibility of introspection with regard to recognition of Intentional states can be overcome.

Yet there is a more critical problem with the fallibility of introspection that faces the programmer. William James distinguishes two general kinds of knowledge: “knowledge of acquaintance and knowledge-about” (1950, vol. I, p. 221). ‘Knowledge-about’ is more important to the programmer than ‘knowledge of acquaintance’, since acquaintance requires the apprehension only of sufficient conditions for the Intentional state, whereas the formulation of the algorithm that instantiates that Intentional state in a computer program requires all the necessary conditions for that state, which only ‘knowledge-about’ can provide. The fallibility of introspection with regard to the recognition that I am in a particular Intentional state was fallibility of ‘knowledge of acquaintance’. Introspection seems also fallible with regard to ‘knowledge-about’, as I suggested earlier with regard to the evaluation of the intelligence of the program using the first person criterion. A program may seem to me to be doing what I do when I think about something, but it may not seem that way to someone else.

James notes that ‘knowledge of acquaintance’ and ‘knowledge-about’ are relative terms and do not refer to distinct knowledge types. “But in general, the less we analyze a thing, and the fewer of its relations we perceive, the less we know about it and the more our familiarity with it is of the acquaintance-type” (p. 221). There does not seems to be a definite point at which ‘knowledge-about’ ceases and ‘knowledge of acquaintance’ begins. Rather the two types are relative to each other in the sense that ‘knowledge of acquaintance’ involves the recognition of fewer analytical relations than ‘knowledge-about’. Given this relative nature of the two knowledge types, if inter-subjective agreement can mitigate the fallibility of introspection with regard to ‘knowledge of acquaintance’, it should likewise address the fallibility of introspection with regard to ‘knowledge-about’. What is required is simply greater analysis of the relations such that the sufficient conditions that make recognition possible are gradually built up and augmented such that they become more like the necessary conditions involved in complete ‘knowledge-about’ something. James himself seems to hold this opinion. “The only safeguard is in the final *consensus* of our farther knowledge about the thing in question, later views correcting earlier ones, until at last the harmony of a consistent system is reached [italics in the original]” (p. 192). Here James relies not merely on an immediate inter-subjective agreement, but a historical, dialectical process resulting in the final consensus. Assuming that such a consensus could be reached, this poses a practical problem to the programmer, in that he does not necessarily want to wait years, decades, or even centuries until such a consensus is reached about the nature of particular Intentional states; he wants to program right away.

Of course such a consensus need not have already occurred prior to the programmer beginning an artificial intelligence program. Rather, the programming attempt itself seems to be part of the dialectical process as an “empirical inquiry” (Newell & Simon, 1997). My first attempt at representing a particular instance of intelligence in a computer program will invariably be critiqued and criticized by others, who find that the algorithm coded into the program does not match the introspective analysis of their own Intentional states. This criticism leads me to make corrections to my program, resulting not only in a better program, but also another round of criticisms requiring me to make further corrections. The hope is that this dialectical process ultimately would reach a point of consensus. Hubert Dreyfus’s argument mentioned earlier was that this process was an “infinite task,” given the nature of the interconnected relationships within the intelligible world. Consequently, no such consensus could be reached.

Whether Dreyfus’s criticism holds or whether a final consensus could validate this empirical inquiry is not my primary concern here. Therefore, I will leave it as an open problem associated with the requirement that the programmer have detailed knowledge of his Intentional states. More importantly for my purposes, the level of detailed knowledge required appears to extend beyond the relationships among Intentional states to the Intentionality itself that underlies these Intentional states. If Intentionality is a feature of intelligence in the Intentional states that are exhibited, it seems that knowledge of the necessary conditions for Intentionality in general are required to implement those Intentional states, as I will address next.

## Requirement 3: Ability to Understand Intentionality as an Intentional Object

Introspection compounds the Intentionality of Intentional states. In thinking about a mental state involving an Intentional object, I take the mental state itself as an Intentional object. This act is not particularly problematic for the programmer, who routinely performs such introspection. However, if an artificial intelligence program is to be truly intelligent, it will need to exhibit Intentionality with regard to its artificial mental states. If I code my complete knowledge of particular Intentional states into a computer program, I do not yet have any warrant to claim that the program exhibits Intentionality, any more than a book that contained a complete description of those Intentional states would exhibit Intentionality in itself. As noted earlier, the programmer typically does not create a program and then check to see whether it exhibits certain features; rather he intentionally designs the program to exhibit those features. If one of those features is Intentionality, then it seems that the programmer must take Intentionality itself as an Intentional object.

Thus far this requirement seems trivial, since in writing the last sentence, I myself took Intentionality as an Intentional object. However, as with other Intentional states, mere acquaintance is not sufficient for the programmer. Detailed knowledge is required, so it seems that the programmer must become an expert on Intentionality. Of course, Searle wrote an entire book on Intentionality (1983), so it seems possible for the programmer to use such a book rather than performing an analysis of Intentionality on his own. Yet Searle avoids an issue in his analysis that the programmer cannot, namely the implementation (p. 15). Searle is concerned to describe the logical properties of Intentionality, however they are implemented. This knowledge is useful to the programmer, but not sufficient. The programmer must go further and figure out how Intentionality can arise within a system, in a similar way as a neuroscientist would figure out how Intentionality arises within a biological brain.

This requirement and its problems are related to the last requirement that I will present, and so I will discuss them at length together. The neuroscientist seems to have a certain advantage over the programmer in studying the implementation of Intentionality in that the neuroscientist's understanding of Intentionality will be mapped directly to the study of the brain. Whatever possible explanatory gap this leaves, it is different from the problems that the programmer faces, since the programmer's understanding of Intentionality must be abstracted from the form in which it appears within his own brain, such that it can be projected outward, not explanatorily onto another brain, but creatively onto a non-biological computer system.

## Requirement 4: Ability to Project Intentionality

The programmer's understanding of Intentional states must be projectable, by which I mean that the programmer must be able to take his understanding out of the context of his naturally occurring mental states in such a form that it can be placed in another context, most importantly in the context of procedural steps written in some language. If his understanding were not projectable, he could never intentionally code those Intentional states into a computer program, but could only contemplate those Intentional states and their relationships in his own mind. As noted in the previous requirement, not only must those Intentional states be projectable, but so must the Intentionality underlying the states, since the necessary conditions for Intentionality in general are part of the necessary conditions for specific Intentional states.

The ability to abstract this Intentionality in order to make it projectable entails abstracting from the peculiarly human context in which it appears. To some extent, this is already done in the contemplation of Intentionality within other animal species. Searle, for example, is willing to grant mental states to his dog (1992, p. 73) and presumably thereby Intentional states as well. If Searle's dog wants a piece of bacon, just as I do, then Searle's dog likewise has Intentional states. It seems that we share the same Intentional object, namely the piece of bacon, but I don't know whether this entails that we both have the same Intentional states. Perhaps dog Intentionality is significantly different than human Intentionality. Still, there may be some warrant for abstracting the same Intentionality from both cases, based on shared features of those instances of Intentionality. This basis seems to be precisely the same as the reason that Searle is willing to grant mental states to his dog, namely "because I can see that the causal basis of the behavior in the dog's physiology is relevantly like my own" (p. 73).

Therefore, it seems possible that the programmer could capitalize on the neuroscientist's efforts by taking the neuroscientist's results on how Intentionality arises in different biological brains, assuming that such results can be achieved, abstracting the pattern of Intentionality from its biological implementations, and projecting that abstraction onto a computer program. Yet here the programmer runs into the possible problem that certain cognitive functions cannot be encoded algorithmically, as I will discuss briefly with regard to connectionism later. Whether those cognitive functions operate according to representational relations, the programmer must take them as representations in order for them to be projectable, but it is not clear whether the subsequent algorithms resulting from this projection truly capture the cognitive function so represented. This problem is not directly important to me, since my concern in this thesis is with the Intentionality of the programmer in writing an artificial intelligence program, whether using a classical artificial intelligence or a connectionist model.

Personally, I do not know how to project Intentionality algorithmically within a program, perhaps because I do not have detailed knowledge of Intentionality as an Intentional object, in accordance with the previous requirement. If I had such knowledge, I would write a very different sort of work than this one. However, I do not have an argument against the possibility that Intentionality could ever be projected programmatically. Searle believes that he has several arguments (see 1992, pp. 197-226), but I will not review and critique them here. From the point of view of the programmer, however, there is one way that Intentionality may seem to be projected. I will take some time to critique this approach, since it bears directly on the practice of connectionism that I will discuss later.

## An Object-Oriented Approach to Programming Intentionality

The approach I have in mind divides the process into two stages: (1) projecting the form of the Intentional states, and (2) projecting the Intentionality of those states. The first stage defines the Intentional object as an object and defines the mode in which the object appears in the state, such as desiring or hoping, and the second stage makes that object and mode into an Intentional state. Immediately, there is a potential problem in that it may be the case that the two processes are not distinct at all, and that Intentionality arises as an integrated rather than a staged process. More importantly for my purposes here is the relationship of this alleged Intentionality within the program to the programmer's own Intentionality. Consequently, I will argue that claims of Intentionality deriving from this approach are made doubtful precisely because of the role of the programmer's Intentionality.

Note that this is not the only possible approach to programming Intentionality within classical artificial intelligence, but the results of the investigation into this approach will be useful in understanding the problems that I will later raise for connectionism, which is my primary concern in this thesis.

Assuming that I have complete knowledge of a particular Intentional state and its Intentional object, in accordance with the second requirement outlined in this chapter, it is certainly possible for me to represent the internal and external structures of the state and its object. I could do so simply by writing out those structures in English or some other language. Having done so in a natural language, it also seems possible for me to represent that knowledge within an artificial language, such as a computer language. Different languages support different syntaxes for accomplishing such a task. There is an approach within computer languages that derives from the work of Frege (1997b), which is particularly useful in this regard, namely object-oriented programming, such as in the C++ or Java languages. This approach allows the programmer to encode data relationships within structures called "classes" that can be instantiated into data entities called "objects". Such classes can be instantiated into multiple objects, which thereby share certain defining properties according to their class. Not only can static properties be defined within classes, but also functional relationships between properties as determined by actions known as "methods" performed on the objects. Consider, for example, the following pseudo-code that defines a class called "Bacon" with several properties and their relationship:

```
class Bacon {
  property Cooked default value = “false”;
  property Texture default value = “slippery”;
  method Cook_It {
        Let Cooked = “true”;
        Let Texture = “rough”;
    }
}
```

There are two properties, called “Texture” and “Cooked”, which initially have the default values “slippery” and “false”, respectively. The idea here is that when I buy bacon at the store or rip it directly out of a pig, bacon naturally comes uncooked with a slippery texture. The method called “Cook_It” defines an action that changes the properties of the bacon. After I cook bacon, it is obviously then cooked, and its texture becomes rough. These properties may already be set when I get the bacon, as for example, if I am served bacon in a fine restaurant. Typically in such a case, I do not experience the bacon first as raw and then cook it. I receive it already cooked. Programmatically, this feature is conveyed by passing parameters to the class when it is instantiated and then creating an initialization method to adjust the properties accordingly. For instance, I could modify the previous pseudo-code as follows:

```
class Bacon (parameter Is_Cooked default value =
“false”) {
    property Cooked default value = “false”;
    property Texture default value = “slippery”;
    method initialization {
        If Is_Cooked = true Then
            Let Cooked = “true”;
            Let Texture = “rough”;
        End If
    }
```

```
        method Cook_It {
            Let Cooked = “true”;
            Let Texture = “rough”;
        }
    }
```

When I am given the bacon, I typically know whether it is cooked or not, which state is represented in this example by the parameter “Is_Cooked”, set by default to the value “false”. Initialization methods, such as the one explicitly called “initialization” in this example, are activated whenever the class is instantiated into an object. If the parameter “Is_Cooked” is true, the method will set the “Cooked” and “Texture” properties accordingly; otherwise, the properties will retain their default values.

Accordingly, it seems as though the form of Intentional objects could be encoded very easily using such an object-oriented approach. Given this form, projecting of Intentionality itself would seem to be accomplished by merely instantiating the class into an object, as for example in the following pseudo-code:

```
    Let My_Bacon = instance of Bacon;
```

If a class is instantiated in an object, then the fact that an object exists seems to entail Intentionality. Furthermore, Intentional states related to those objects could be encoded as methods associated with a class representing a creature capable of having those Intentional states. In other words, Intentional states are states of someone whose activation is directed to an object instantiated from a class. For example, the following pseudo-code represents someone desiring bacon:

```
    class Person {
        property Desiring;
        method Desire (parameter About_What) {
            If About_What = “bacon” Then
                Let Desiring = instance of Bacon;
            End If
        }
    }
```

When instantiated, this “Person” has a property “Desiring” that is empty if the person does not desire anything.  If the person does desire something, the method “Desire” is called with a parameter “About_What” that indicates what the desire is about.  If this parameter is set to “bacon”, this method in turn sets the “Desiring” property to be an instance of the “Bacon” class, and consequently the person would have bacon as an Intentional object in the “Desiring” mode.  Of course, the method would need to be much more flexible to handle a wide variety of objects of desire besides bacon.  Still, this seems to satisfy Searle’s description of Intentional states mentioned earlier, in which “every Intentional state consists of an *Intentional content* in a *psychological mode* [italics in original]” (1983, p. 12).  The mode is represented by the property “Desiring”, and the content is the value of this property, such as an instance of the “Bacon” class.

I have two main criticisms of the claim that this program exhibits Intentionality: (1) the putative Intentionality of the system is parasitic on the Intentionality of the programmer, and (2) the Intentionality of a particular object is undermined by the underdetermination of the formal structures of that object that are instantiated by assignment.

The focus of my first criticism is the instantiation of the class into an object, such as the pseudo-code statement, "Let My_Bacon = instance of Bacon."[4] Such assignment statements occur not only in object-oriented languages, but also in procedural languages such as BASIC or C, such as "Let X = 2". Not only is the very writing of this statement an intentional act of the programmer, but the form of the statement is itself of an intentional act. The programmer's intention is that the program intend to take one thing as another. I think that it is incorrect to say that the program truly intends to take one thing as another, but rather the interpretation of what it does is an artifact of the programmer's own Intentionality in writing the program in this way. What it does in itself is thus subject to interpretation.

The programmer sees the program doing what it was intended to do, and may therefore be inclined to see Intentionality within the system, since the program takes an object to be an instantiation of a particular class. Even a non-programmer observing the results of the program might be willing to grant Intentionality to the program. This seems to be a clear instance of what Searle calls "as-if Intentionality" (1992, p. 78), based precisely on the intention of the programmer to create Intentionality. This ascription need not be restricted to the programmer who created the program, but may be recognized by someone else examining the code. In such a case, I think it is the nature of the assignment statement that misleads the interpreter. The assignment statement leaves any putative Intentionality within the system merely as a projection of the programmer's intentions, rather than as native Intentionality arising within the system. Human Intentionality does not appear to consist

[4] There is an aspect of Platonism in this kind of assignment, in that the class or Form must exist in its perfection prior to its instantiation in an object. Whereas according to Plato the Forms exist in a different realm, in this case all the Forms must exist within the program itself. It seems problematic that I would already have explicitly defined within me all possible classes that I could ever use, and therefore strange that an artificial intelligence program needs to have all classes pre-defined. Of course, there are programming languages such as LISP that allow the program itself to create new structures or classes dynamically without having previously been defined, which may address this concern.

merely in the assignment of an empty placeholder entity to be an object of a particular type, and therefore it does not seem proper to grant a machine Intentionality in the analogous case. Such Intentionality remains dependent upon the recognition of the program's action as Intentionality, and therefore is parasitic on human Intentionality. If Intentionality happens to arise natively within such a system, it might be by virtue of some accidental interaction between hardware and software, but if it does not arise solely by virtue of the algorithmic steps within the program, then it is not classical artificial intelligence. If the algorithmic steps include a bare assignment statement that is intended to suffice for Intentionality, then such putative Intentionality is a mere relic of the programmer's intention, and therefore the perceived Intentionality is parasitic on the programmer's own Intentionality. Requirements 3 and 4 concerning the ability to take Intentionality as a projectable Intentional object do not seem to have been met.

It might be objected that this seems to be a problem only when considering the writing of the program. Once the program runs on a computer, the functioning of the program on the system corresponding to the bare assignment statement will no longer take the form of a bare assignment as I have presented it here. For example, in the instantiation of class into an object, a certain kind of function is called, known as a "constructor" function, which performs various tasks as part of the creation of the object. It may be claimed that it is what happens during the constructor function that grants Intentionality. Yet I would counter that, on the one hand, if it is claimed that what grants Intentionality is what the hardware does in conjunction with the constructor function, such as allocating physical memory on the computer, then it is not a case of classical artificial intelligence, since it is not by virtue of the algorithmic steps that Intentionality arises, but rather by virtue of the physical implementation. On the other hand, if it is claimed that the constructor function contains algorithmic steps that grant Intentionality, then those algorithmic steps must not themselves contain any assignment statements else the problem is merely pushed down a level. For the constructor function not to issue any new assignment statements means that the algorithmic steps within the function must act on existing data structures, which must have been allocated by some assignment statement elsewhere. Therefore, I do not think that the problem of the deficiency of assignment statements in creating Intentionality can be avoided within classical artificial intelligence by appealing to the functioning of the system running the program.

My second criticism addresses the formal aspect of representing Intentional states and objects in classes. The problem that I see here is that ultimately the structure of the class must be represented in some fundamental data type on the computer, such as an integer or a string, that itself is ultimately reduced to data bits within the computer system. Reduced in this way, I think there is a question as to what makes a particular object an Intentional object of one kind as opposed to another. The pseudo-code of the bacon class that I outlined earlier had two properties, “Cooked” and “Texture”, but many other objects have these representational properties, such as any other kind of meat or even vegetables. Even if a set of properties is defined with regard to bacon that identifies it distinctly from vegetables or other meats, those properties are represented in the computer in low-level data types. The property may be labeled “Cooked”, but it is essentially just a binary variable, which could have been labeled “Hairy” or “Transcendental”, which in turn would have defined something quite different. The meaning of the putative Intentional object within the program is underdetermined by its underlying data structures. For the system, these data structures are merely collections of related values, whether numbers or words or references to other data structures, and on a binary computer, these values are ultimately reduced to binary bits, namely ones or zeroes. The labeling of properties in this case is potentially deceiving, since in practice the property indicated by the label "Cooked" really does stand for such Intentional content, but only for the programmer. Those labels exist only as tokens within the programming language that the programmer uses to develop the program, whether BASIC or C++ or Java or some other language. The programmer's code must be compiled into machine code[5] in order for the computer to run the program, at which time the labels themselves are removed by the compiler software and replaced by bare references to data locations. Consequently, the system has no awareness of the labels at all, let alone awareness of the meaning of the labels. For the system, the putative Intentional object is merely a collection of data relations. If such a collection of data relations could correlate to several different Intentional objects, then the putative Intentional object is underdetermined with regard to its data relations within the system. It cannot be a particular Intentional object, since there is no difference within the system between that Intentional object and another with the same data relations. Yet if the size and complexity of such a collection uniquely correlates to a particular Intentional object, the question remains whether this unique correlation within the underlying data structures is sufficient to qualify as Intentionality or whether it is merely correlation.

Here my criticism intersects with both Dreyfus's and Searle's criticisms of classical artificial intelligence that I mentioned earlier. Perhaps a particular representation of an Intentional object is underdetermined in its own underlying data structures, but that

[5] Machine code is a series of binary data fed as instructions or data directly to the computer's central processing unit or CPU, which has been designed to interpret particular numerical values in the binary data as particular instruction. In the case of an interpreted language platform such as Java or many forms of BASIC,

representation within the context of other representations may uniquely determine a particular Intentional object. This representational situation is comparable to Searle's notion of a Network of Intentional beliefs (1983, 65-71). Both Dreyfus and Searle claim that this Network itself is inadequate: for Dreyfus, because it must encompass the being of the creature itself; for Searle, because the Network has meaning only against a background of abilities and practices. Since neither the being of a creature nor the background of abilities can fully be represented algorithmically, the task will always remain incomplete, and therefore Intentionality will never be achieved in such a case. For my part, I am not prepared to agree with Dreyfus or Searle that the task is impossible, but given the challenge of the underdetermination of underlying data structures, I think the task is prohibitively difficult.

## Evaluation

I have outlined four requirements in this chapter related to the Intentionality of the programmer that must be met if the programmer is to succeed in intentionally creating a genuine artificial intelligence program. The problems associated with introspective access to cognitive functions and complete knowledge of such functions may be solved through a combination of first person and third person methods. However, the need to take Intentionality itself as an Intentional object such that it is both completely defined and projectable seems to be a more difficult problem. If the putative Intentionality within the system remains parasitic on human Intentionality, then it seems that the programmer's knowledge of Intentionality as a projectable Intentional object is faulty or incomplete. The

however, the programmer's code is compiled into an intermediary byte code format which the interpreter implements into machine code when the program runs.

perceived Intentionality in such a case leaves the Intentional objects underdetermined in their underlying data structures, further weakening any claim for Intentionality within the system.

Whether these problems prove fatal for classical artificial intelligence, I will leave as an open question.  I have outlined the requirements of Intentionality for classical artificial intelligence primarily to set the stage for a comparable discussion of Intentionality within connectionism, which I will discuss next.  However, the two major problems identified here, namely parasitic Intentionality and underdetermination of underlying data structures, will reappear within the practice of connectionism.

# CHAPTER IV

# CONNECTIONISM

It might be supposed that connectionism was developed in response to the problems of classical artificial intelligence, specifically to address those problems. In truth, both approaches to artificial intelligence arose approximately at the same time, which makes a better story in my opinion, in which both approaches struggle against the other for ascendancy. For a brief history, see Dreyfus and Dreyfus (1990).

Connectionism is an interpretation of neural networks in which cognitive functions arise as a result of the connections between nodes in such a network. Unfortunately, there is not a single interpretation of the meaning of these connections, as I will discuss later. Before I review the various interpretations, I will outline the general principle of neural networks and distinguish the two primary forms of neural networks.

## General Principles of Neural Networks

Neural networks are a processing paradigm in computer science in which the processing of a system does not occur at a single, central point in a serial fashion, as derived from the Turing machine model and its later implementation in a von Neumann machine. A Turing machine is a logical conception involving a single executive control function that reads and writes data from a tape and processes that data as instructions in accordance with a set of rules in a pre-defined table in reference to a number of internal states that can be read and modified by the system itself according to those rules. Von Neumann developed this logical conception into an architecture for a digital computer on which modern computers are based.  In both cases, there is a single point at which the instructions are processed, in the Turing machine as an abstract role that is performed, and in the von Neumann machine as a definite piece of hardware, namely the computer's central processing unit or CPU.  In a neural network, however, processing is distributed across multiple processing points, or nodes, and proceeds in a parallel fashion with each node working at the same time as other nodes.  Any biological brain is a neural network in which neurons comprise the processing nodes as physical entities.  Brains are natural neural networks, but what are most often studied in the discipline of neural network research are artificial neural networks.  Such networks are typically simulated on serial computers, such as any personal computer, but they may also be created using physical processing nodes, such as an array of personal computers or even a collection of pot-bellied pigs suitably connected together.  Computer simulations of neural networks are most useful, since the structure and processing rules within the software may be changed more easily than an array of hardware units, whether of electronic components or of pigs.

Connectionism depends upon neural networks, but research into neural networks need not depend upon connectionist interpretations.  Neural network research has become a discipline in its own right, whose goals include the analysis of data relations within neural networks and the creation of new types of neural networks that perform better than other types of networks.  Consequently, a researcher into neural networks may not care about mental representations or the relationship of artificial neural networks to the cognitive functions of the human brain at all.

Figure 1 shows a fairly representative structure for a neural network.  Each circle represents a processing node, and the arrows indicate the flow of data from one node to

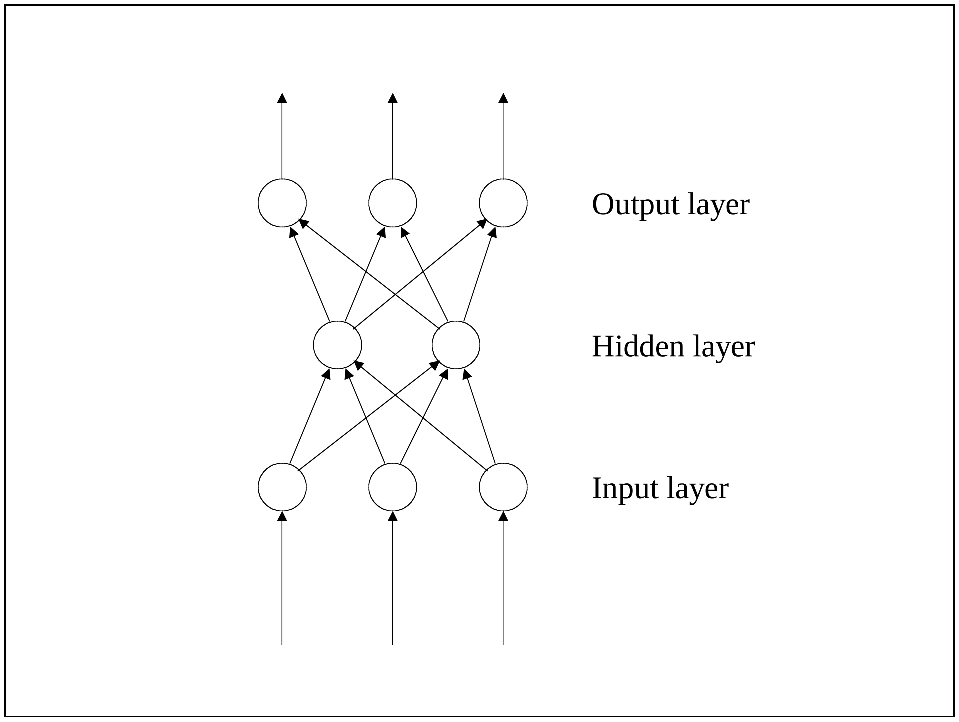

*Figure 1*. Typical neural network structure.

another. There are usually three layers of nodes within a network, an Input layer, an Output layer, and a middle processing layer known as a Hidden layer, because these nodes have no direct connection outside the network except through the nodes in the Input and Output layers. A network need not explicitly have these three layers, but it will almost always display these same three functions, namely receiving data from outside the network, processing the data internally, and passing the results of the processing outside the network.

The Input layer will receive some value, but that value may represent data in a high level form, such as the price of pork belly futures on the commodities market, or in a lower, more raw form, such as a reading from a voltmeter. Likewise, the value that is output by the Output layer might be in one of these forms. In either case, the network is simply processing values. There can be quite a large number of nodes in each layer, and there may in certain cases be several hidden layers.

Note that in the typical example in Figure 1, each node in the Input layer is connected to each node in the Hidden layer, and likewise each node in the Hidden layer is connected to each node in the Output layer. This is a common structure, but is not necessary. For example, some nodes in the Input layer might not connect to certain nodes in the Hidden layer. Nor must the hierarchy of layers be strictly observed. A node in the Input layer might by-pass the Hidden layer entirely and connect directly to the Output layer. Furthermore, data flow need not always be in the direction from Input layer to Output layer. Nodes in the output layer might be connected back to nodes in the Hidden layer, such that the results of the network are themselves taken to be inputs to the same network for further processing.

The processing within each node needs to take the data passed to it and decide what data to pass on to the next layer, if any at all. This processing may take different forms and implement different rules, but the typical approach is inspired by the processing of actual neurons within a biological brain. Associated with each input path is a weight value that is applied to the incoming value. All weighted inputs to a node are combined and compared to some threshold value. If the sum of the weighted inputs exceeds or is equal to the threshold value, then the node will output some value, otherwise it outputs nothing or the value 0. A node using this sort of processing is known as a Threshold Logic Unit, and is illustrated in Figure 2. More sophisticated processing algorithms have been devised, but most if not all of them share the same technique of using weighted sums to determine the output according to some formula. I will not review other processing types, since the specific algorithm used within a node is not significant for my purposes here (for details, see Gurney, 1997; and Picton, 2000). What is important to appreciate here is that the kind of processing within a neural network is distinctly different from the kind exhibited in classical artificial

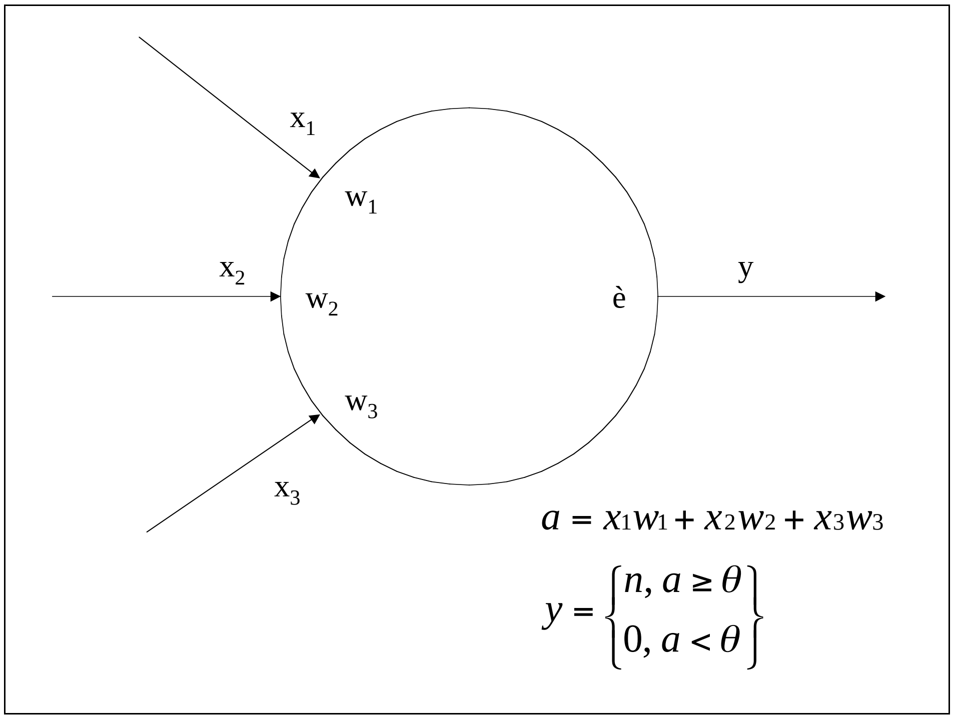
$x_1$
$w_1$
$x_2$
$w_2$
è
y
$w_3$
$x_3$
$a = x_1w_1 + x_2w_2 + x_3w_3$
$y = \begin{Bmatrix} n, a \geq \theta \\ 0, a < \theta \end{Bmatrix}$

*Figure 2*. Typical processing within a single node.

intelligence. There the processing steps mirrored the programmer's reflective understanding of the structure of intelligent operations, whereas here the processing does not seem to correspond to anything recognizable in intelligence. Rather, each node within the system takes values from other nodes and combines them in some way to provide a single output value, which it passes on to still other nodes.

Taken abstractly, such a neural network would merely be an intellectual curiosity in which the interrelationship of values could be traced throughout the network. However, values within the network, at least the input and output values, typically have some meaning, and therefore the network is understood to be doing something, such as making a decision. If my input values are the price of pork belly futures over the past few days, my output values could represent a decision either to buy, to sell, or to hold my contracts. A common use of neural networks is pattern recognition, in which the raw data of some phenomenon, such as pixel patterns from an image, are input to the network, and the network recognizes whether the input is a letter of the alphabet or a face, for example.

How can a neural network make such decisions? This decision or recognition ability is a result of the patterns of weights within the system. Given a suitably configured system of weights, the proper output will be given to the appropriate inputs. How are such suitable weights determined? Such weights are a result of the training of the network. There are two broad categories for training in a neural network: supervised and unsupervised learning.

## Supervised Learning Networks

If a network is to be trained using a supervised learning method, a set of training data is required. Training data consists of a set of inputs and the correct set of outputs. This correlation of inputs to outputs is not generated by the neural network, but is created externally. I may, for example, watch the price of pork belly futures over several days and record what I understand to be the correct response to the data, whether to buy or sell or whatever. Or I may collect a series of images and record whether that image is a particular letter of the alphabet. Such training data provides a set of paradigm cases in which the network should give the indicated output when given the corresponding input.

Yet given such input, a neural network will typically not provide the correct output on the first try, since the weights are not properly set. The weights must be adjusted such that the correct response is given by the Output layer. It is a difficult task to know exactly how to adjust particular weights, since the contribution of each node to the end result is not known beforehand. Adjusting the weights for a node too far in a particular direction may seem to give the correct results for a particular set of training data, but may give wildly erroneous results in another set.

There are a number of training methods, which I will not review here (see Gurney, 1997; and Picton, 2000), but which typically make numerous minor adjustments to the weights in the system until the functioning of the system settles into a stable pattern. Each minor adjustment in a weight typically makes some contribution to the resulting output of the system, whether correctly or as an error, though very small changes may make no appreciable difference. This contribution is used as a basis for re-adjusting that weight in the appropriate direction, whether increasing or decreasing. The weight is typically not changed much at all, but is only nudged in the correct direction. Since there is an interdependence of weight factors within the system, it will often take very many iterations of such minor adjustments until the total arrangement of weights produces the correct results for a particular set of training data. Yet that would only provide correct results for that instance. Consequently, the process is repeated for different sets of training data until the network provides correct results for all the training data. Such a training method is known as "back-propagation of error" since the net error of the results in a particular attempt is pushed back into the network as a guide for adjusting the weights.

Thus far the network is trained for specific training data, but if the training data cover an appropriate range of possible cases, the network will be able to apply its system of weights correctly to novel cases of input data. Therefore, rather than merely being able to reproduce my decisions concerning whether to buy or to sell my pork belly futures, the network will be able to monitor future prices and make such good decisions that I could entrust my entire portfolio of pork belly futures to its judgment. Such is the hope for this sort of system, at least.

Again, the particular algorithm used within the training process is not important for my purposes.  What is important is that within supervised learning, a set of training data is created by the programmer in which the desired outputs are matched to a set of inputs.  Then the network is trained such that it can reproduce those outputs given the same inputs.

## Unsupervised Learning Networks

There is another general type of neural network that does not require such forced learning, whose learning is therefore unsupervised as opposed to the supervision required in creating a training set and adjusting the weights properly.  Unsupervised learning commonly occurs within network architectures that are called “self-organizing nets,” usually employing a technique called “competitive learning.”

Figure 3 shows an example of such a self-organizing net featuring competitive learning.  The important difference between this network and the network in Figure 1 is that nodes in the same layer are connected to each other.  Node B is connected to nodes A, C and D.  Though not shown for reasons of simplicity, these other nodes in this layer are likewise interconnected as node B is.  The interconnections within this layer serve as inhibitory inputs to the other nodes in the layer.  Thus the output of node B has a negative weight in node A that tends to inhibit node A’s output.  The effect of this inhibitory influence is to make the nodes within this layer compete for their output.  The node that inhibits its competitors most is the one whose output will prevail over the others for a given set of inputs.  Consequently, given an initial random set of weights within such a network, and given some set of inputs, one of the nodes A, B, C or D will prevail over the others, and will therefore be considered the winner.

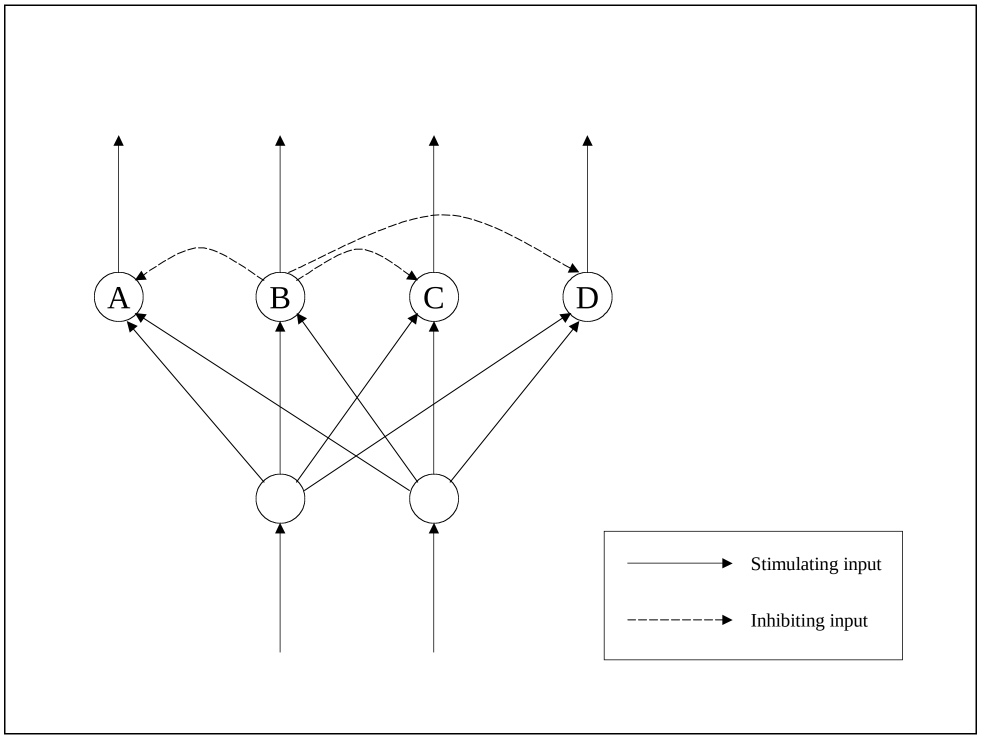

*Figure 3*. Competitive learning in a self-organizing network.

There is still some training that occurs in such a network, since given an instance of competitive success by node B, the stimulating weights associated with the input to node B will be strengthened to reinforce the positive results from this node, and the inhibitory weights on nodes A, C and D will be strengthened to reinforce the negative results from these nodes.  Consequently, the system trains itself as it operates, though it may still require external judgment concerning the criterion to determine at what point the network is adequately self-organized.

The benefit of this sort of network over a network that requires supervised training is not only that the set of training data need not be correlated to specific desired outputs beforehand, but also that the possible outputs need not be determined beforehand.  The programmer need not stipulate whether the system is supposed to distinguish between bacon or pork chops, since the self-organizing principles of the network would classify the incoming data into different groups that might represent bacon or pork chops.  Thus the network could be understood as discovering certain features of the input data, such that the output from the system determines a topography of the data.

There are many variants of networks employing unsupervised learning, which I will not review here (see Gurney, 1997; and Picton, 2000).  What is important for my purposes is that such networks have the capability of classifying input data into categories without those categories being determined by the programmer beforehand.  After it has processed several sets of data, the weights within the system are set to reinforce the categories that it has organized within itself, such that after a certain point in time, further training is not necessary, since it will correctly classify novel sets of data in accordance with the weights it has already established.

## Connectionist Interpretations of Neural Networks

The study of neural networks as such is a field in itself.  The interpretation of neural networks for cognitive science and philosophy is connectionism.  These interpretations typically are formed based upon the positions taken on the issues of mental representation and computability of thought and often arise in contrast to the Language of Thought hypothesis.  This hypothesis holds that thought occurs in the form of mental representations that have semantic and syntactic relations to each other in the same way that a language does (Fodor, 1990).  These language-like representations have genuine causal powers and are not merely useful devices for discussing the experience of cognition while not necessarily referring to the underlying mechanism of thought.  The Language of Thought hypothesis holds that these syntactical structures of representations depend upon the underlying physical structure of the brain, which explains their causal powers, and that the semantic features of mental representations are due precisely to their syntactical features. Under the Language of Thought hypothesis, my belief that bacon is delicious is a product of my mental representations of bacon and of the property of being delicious, each of which bears a syntactic relation to the other within my thought that is similar or the same as in the statement of the propositional content of my belief. These representations have sufficient causal powers in relation to other mental representations that I hold such that they result in my particular belief state.

If the Language of Thought hypothesis is correct, then classical artificial intelligence may be a possibility, since the programmer need only uncover the exact nature of the mental representations as well as their semantic relationships and replicate them within the syntax of a computer language. If existing computer languages are insufficient to model such representations and semantic relationships, then it would still seem possible to create a new computer language with adequate capabilities, since the representational nature of thought under this hypothesis makes it projectable in some form of external representation. Of course the attempt to formulate natural language comprehension within classical artificial intelligence has not completely succeeded, but this is not definitive proof that the Language of Thought hypothesis is incorrect. Rather it may simply show that just as the strict formulation of natural language is more complex than was previously imagined, so is the Language of Thought.

Thus the Language of Thought hypothesis is compatible with the notion of the computability of thought, depending upon the computability of language. If thought does have a language-like structure, then that structure can be formulated into a series of rules, and those rules can in turn be coded in such a way that a digital computer or some other calculating device can implement those rules by means of more basic calculative functions. The relations of each of these levels ultimately means that thought can be formulated into a form such that it can be computed. The notion of syntax in a computer language itself derives from language. Any formulation of computational steps in the creation of a computer program requires syntax within a formal system of symbols, whether the language used is BASIC, C++, Assembly language, or even low-level machine code. Logically these last two statements do not entail that language is computable, but parallels between language and programming may seem compelling.

Various connectionist theorists take differing positions on the issue of mental representations and computability. The Language of Thought hypothesis holds one view of mental representations, namely that such representations have a language-like structure and not merely semantic structure. Another view is that representations are structured like pictures. Within a connectionist system, it is not obvious whether there are language-like representations, pictorial representations, or any representations at all. Whereas it is obvious that there is some computation occurring within the connectionist system, explicitly within the processing of a single node, it is not clear that computation occurs between representations within the system, if in fact there are representations at all.

Terence Horgan identifies three notable interpretations among connectionist thinkers (1996, pp. 95-96). The most common interpretation is that there are mental representations within a connectionist system, but that these do not have a language-like or pictorial structure at all. Rather the structure of mental representation that appears within neural networks takes the form of activation vectors distributed throughout the hidden nodes of the system. These vectors appear as the weighted values exchanged between nodes. Therefore, there is computation among these vectors, but the distributed and interconnected nature of these vectors means that the computation cannot be understood as taking place among representational entities such as the Language of Thought and the pictorial theory of mental representations hold. The syntactic requirement of the Language of Thought hypothesis seems to demand that representations be separable, in order for there to be proper syntactical structures between the representations. Within a connectionist system, however, it is not clear or even possible to determine the demarcation between one putative representation and another.

A second interpretation denies that there are representations at all within connectionist systems, primarily because any putative representation posited by the first interpretation among the hidden nodes of the system plays no explanatory role in the understanding of cognition. If I think I recognize something corresponding to the representation "salty" in the hidden nodes of a connectionist system that recognizes bacon, that does not mean that what I take to be that representation has causal powers as a representation, since the system would function just as well without my recognition of any representations within its hidden nodes, and in fact would function just as well whether there are any representations at all present within the system. Connectionists may posit such representations, but they are merely mapping folk psychological notions onto the connectionist system for convenience. The representations seen in this way are not fundamental, but are simply devices used in understanding the connection between the underlying cognition and folk psychology.

A third interpretation, held by Horgan himself, agrees with the Language of Thought hypothesis that there are mental representations and that they typically do take the form of language-like structures; however, this interpretation denies that the relationships between these representations are computable. Rather, the relations between mental concepts take the form of dynamical systems that are not reducible to linear algorithmic computation, but rather employ differential equations to describe the behavior of complex systems. There is therefore some computation involved in the functioning and description of the system, but the computation that underlies the mental concepts within the system is not a computation between those concepts as such. There is no computation between the concepts of “salty” and “fatty” that is involved in the mental representation of “bacon”, though there is computation underlying the complex system in which these concepts and representations arise.

I will not attempt to evaluate these interpretations directly in this thesis, since more fundamentally, I doubt that there is sufficient data available on which to base any such interpretation properly. In the next chapter, I will challenge the notion that there can be any grounds for holding any of these interpretations yet, based on the current practice of connectionism. The problem, as I will argue, is the role of the Intentionality of the programmer.

# CHAPTER V

# PROBLEM OF INTENTIONALITY FOR CONNECTIONISM

In contrast with classical artificial intelligence and its problems, connectionism appears better suited for achieving artificial intelligence. After all, connectionism and neural networks were based upon the processing structures in biological brains, so the connectionist strategy for achieving artificial intelligence is to replicate artificially the underlying structures of naturally occurring intelligence.

The connectionist goal of creating an artificial brain seems less ambitious than the coding of cognitive functions algorithmically as in classical artificial intelligence; yet the issue of mental representations that is so central to classical artificial intelligence continues to be an issue for connectionism. If, for example, Horgan's interpretation of connectionism is correct, and connectionism can show that mental representations have structures similar to language but that the relations between them cannot be computable, this interpretation would show why classical artificial intelligence thought that it could encode cognition algorithmically, but could not succeed. The language-like structures of mental representation seem amenable to algorithmic representation, in the way that one language can be translated into another; however, since the relations between those representations are not computable, no algorithm can provide a successful translation of those relations.

Yet Horgan's is only one competing interpretation of connectionism. How can one adjudicate between these competing interpretations? It depends on what is taken as a representation in a connectionist system. I will argue here that these interpretations cannot yet be adjudicated, since the current practice of connectionism has not yet yielded sufficient data on the nature of representations on which a rigorous interpretation can be based. The problem here again is Intentionality, and once again the culprit is the programmer. The objections that I raise are directed specifically at aspects of the current practice of connectionism, not at connectionism in general. In the following chapter, I will offer a proposal whereby connectionism might amend its practice to overcome these objections.

## Problems with Supervised Trained Networks

Consider the following remark by Paul Churchland in explaining the way a particular connectionist system recognizes faces: "Not to put too fine an edge on it, what the network has developed during training is a family of rudimentary *concepts*, concepts that get variously activated by sensory inputs of the appropriate kind [italics in the original]" (1995, p. 50). In this case, Churchland is discussing a network that features supervised training, namely that a training set of data was created in which the correct outputs were matched to the appropriate inputs, and the weights in the network were adjusted to give the correct results for the training data using the technique of back-propagation of error.

At the very least, I think what Churchland says here is imprecise. Consider the sample neural network in Figure 4. Suppose that this network were fed raw sensory data in the form of values from various sensors that register certain qualities of the fragrance of an air sample, in other words, that it were fed raw smells. The task of this network is to distinguish between various pork products. Consequently, the output nodes A, B, and C represent whether the input smell is from bacon, ham, or pork chops, respectively. Of course, a good network of this kind would need either to include a comprehensive set of output possibilities, or a default output node that would register when the input smell was

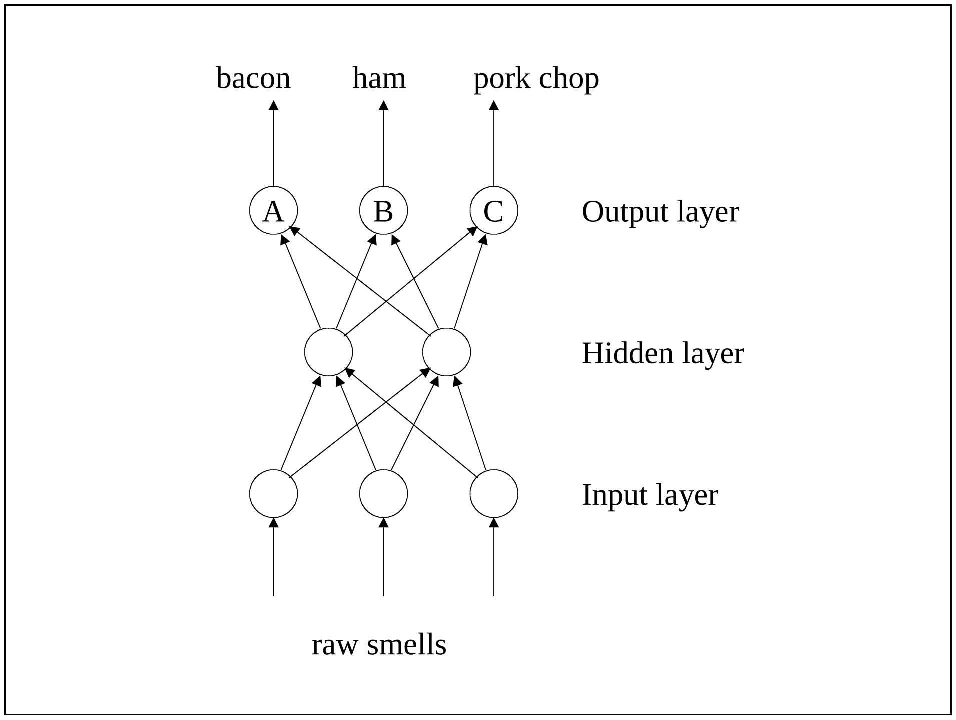

*Figure 4*. Sample supervised trained network classifying pork products.

none of the other output nodes, and this will become significant later.  In order to train the set, I would create a training set of values taken from sensors that had been fed samples of bacon, ham, and pork chops of different varieties and correlate those values to the correct output values in each case.  I would then perform training runs using the sample data, and adjust the weights in the system until the network correctly classified all of my training data. Thereafter, when fed new raw smell data, say from a nice Virginia ham, I would expect the network to classify that novel data correctly as ham rather than as bacon.

According to Churchland, this network has developed a set of rudimentary concepts corresponding to bacon, ham, and pork chops.  If Churchland means by “rudimentary” something corresponding to Searle’s as-if Intentionality, I would agree.  The network certainly behaves as if it had concepts.  Furthermore, if the statement is intended to indicate that concepts seem to result in some way from the pattern of weights in a network, then I would likewise agree.  A biological brain capable of forming concepts seems to have available only the mechanism of such a system of weights in order to form concepts. However, if Churchland's statement is intended to make a claim that the network in question reveals anything significant about the nature of the particular concepts in question, then I think the claim has overstepped its justification in the data.

The problem, as I see it, is in the assignment of meaning to the output layer.  The programmer assigns output node A to represent bacon, and node B to represent ham, and so forth.  It certainly does not matter which node is assigned to which concept, since the pattern of weights in the middle layer would still train to essentially the same pattern.  But the very act of assignment in this case is a close parallel to the act of assignment in classical artificial intelligence, in that the programmer is projecting his own fully formed Intentionality onto the system by an intentional act in the form of "Let node A represent bacon."  That projected Intentionality is then reinforced in the system by the process of training, in which the raw sensory data is pre-associated with the programmer's own representation of that data as an Intentional object.  Note that this situation is distinctly different from what happens in a self-organizing network, in which the assignment of nodes is performed by the system itself, though as I will discuss shortly, a self-organizing network is not immune from problems. Within a trained network, however, the assignment of nodes is an intentional act of the programmer, projecting his Intentionality onto the system, just as the assignment of a variable to represent an object of a specific class in classical artificial intelligence merely projected Intentionality onto the system rather than creating conditions whereby Intentionality could arise intrinsically within the system.

This intentional projection in the case of a neural network, however, seems different than in the case of classical artificial intelligence, in which the projected Intentionality seems to remain in the system as an artifact of that projection.  It could be argued that in the case of a neural network, the act of training serves to reverse the effect of the intentional projection of Intentionality.  By adjusting the weights in the system, the assignment of output nodes is given a “native” foundation in the network, since the intended Intentionality implied in the output nodes is not left merely as an artifact of the programmer’s act of intentional assignment, but rather it is given a foundation and justification in itself such that the Intentionality of the output nodes could arise in its own right.

To a certain extent I agree with this argument.  There seems to be some grounding of Intentionality within a connectionist system that is missing in a classical artificial intelligence program.  However, I claim that traces of the programmer’s intentional act of assigning concepts to the output nodes still remain within the system, such that any claims of genuine Intentionality must be questioned.  Furthermore, any claims that the resulting network reveals something significant about the concepts at issue must likewise be questioned, given the role of the programmer’s Intentionality in these cases.

The first issue is the nature of the output itself.  A single node within a single output layer represents an entire concept.  Churchland himself notes that this configuration is unrealistic.  “That small population of cells is there only to provide the researchers with a convenient means of monitoring the network’s performance.  It is not intended to correspond to anything in the brain” (p. 51).  There is no reason to think that the concept of bacon within my brain is represented by a single neuron, or that the distinguishing of bacon from ham and pork chops in my brain occurs between three neurons, as in Figure 4.  In reality, the structures are much more complex, and the simplicity of the structure in the connectionist system is designed to simplify the task of the researcher in evaluating the effects of the hidden layer. Yet if the realistic structure of the output is more complex than as a single node within a single layer of nodes, then it seems to me that the structures of weights within the hidden layer that determine that output must be correspondingly more complex. Consequently, that complexity of the output would seem to be part of the concept as well, not merely in the pattern of weights in the hidden node that contribute to that output considered as a single compact entity. The complex structure of the output is not a simple content in the form of a positive value output through node A in my example in Figure 4, representing bacon by virtue of that positive value. Rather, the content and the complex form of the output would seem to be bound up in each other. This complexity cannot be seen in a connectionist system such as the example in Figure 4, precisely for the reason that the system is set up to mirror the programmer's Intentionality in the output, namely one node representing each Intentional object. Therefore, what can be claimed about concepts from such a system is merely a hypothetical statement, namely, "If concepts could be represented at a single nodal point, then the system of weights in the middle layer of the network represents their relations." To this extent, Churchland is correct in attributing a rudimentary system of concepts to the network. However, the system is too rudimentary to say anything significant about the concepts themselves, for instance, how those concepts relate to other concepts. Some other connectionist system may not be so rudimentary, but it would need first to determine how concepts in general could be implemented within a connectionist system, and it seems problematic to determine this in a system that simply posits a concept at a single nodal point.

The second issue I have is with the relations of the output nodes to each other, given the selection of the particular output nodes to be trained into the system. This selection seems to be either arbitrary on the part of the programmer, or represents the programmer's own Intentional analysis of the conceptual relations that the network is supposed to implement. Within the system, of course, the output nodes do not have any direct relation to each other, at least in my example, since the nodes are not directly connected to each other. Rather, data flows into them from the hidden nodes and not from other output nodes. There could be a network in which the output nodes are connected to each other, and I will consider such cases below. The relations I consider here are the indirect relations resulting from the training. The distributed nature of a connectionist system means that the weights throughout the system do not correspond exclusively to one concept or another. Each weight contributes to the output in a greater or lesser extent, and the particular extent is determined during the training process. This means that if there were just one additional output node in the output layer, the weights would need to be adjusted differently to accommodate the extent to which each node affects the additional output node. The direct consequence of this situation is that whatever rudimentary concepts are developed in the hidden layer are determined against the entire set of concepts selected for representation within the system. The programmer's choice of which concepts to assign in the output layer shapes the conceptual scheme in the hidden layer (see Dreyfus & Dreyfus, 1990, p. 331). Nor is it clear whether the relation of the conceptual scheme given one set of outputs has any determinate relation to the conceptual scheme given the same set of outputs with the addition of a single output node. A new output mode does not merely add a new part onto the system, like adding a new module, since the network functions by means of interconnected nodes. It may be that the pattern of weights in the hidden nodes is determined so holistically relative to the output nodes that the conceptual schemes between the two systems are incommensurable. The difference between the two systems is not merely that certain weights are different; rather all of the weights are different. In this sense, the system of concepts that the network develops is not rudimentary in the sense that it is an early form on which the system builds its mature system of concepts. Rather the system of concepts is rudimentary in that it is deficient in comparison to the system of concepts it requires relative to a larger set of potential outputs. It is deficient because the true range of the conceptual scheme is much broader than the restricted group of outputs that the programmer has selected.

It might be argued that a complete set of outputs within a given topic would negate this issue, since the programmer's choice of outputs would then match the complete possibilities within the topic. For example, if a connectionist system were created to identify letters of the alphabet, the programmer would assign each letter to an output node, and that would exhaust the possibilities. However, such an argument assumes that the recognition is always made among letters of the alphabet, which would still result in a hypothetical statement of the concepts in the system: "If the system only considers letters of the alphabet, then the system of concepts of such letters would be in such and such a way." Humans do not merely distinguish between letters of the alphabet, but they also distinguish letters from punctuation, for example. If the system of concepts within the hidden nodes is determined holistically, then the concepts of each letter will be incommensurable in systems either including or excluding punctuation. The situation is even worse, since letters are not only distinguished relative to other letters and punctuation, but also between non-grammatical marks or shapes, such as triangles or smiley-faces. Unfortunately, the full range of possible marks cannot be given within such a system, so if systems of concepts are incommensurable in the way I have suggested, then the concept of a particular letter within such a connectionist system is not even rudimentary in the sense of deficiency, since its putative deficiency cannot possibly be made up. The holistic nature of concepts demands that all concepts be represented in the output, which seems to be a practical impossibility.

Perhaps this holism is not ultimately a problem. In the end, it may be discovered that a particular concept can be instantiated in a small portion of the network without having all other concepts present; however, the nature of that concept must first be determined. Given that the individual weights within the network contribute not to a single concept, but potentially to all putative concepts given in the output nodes, if claims about the nature of the concept are made based on a subset of concepts within the network, there is some reason for wondering whether that claim is dependent upon the programmer's particular selection of concepts. If it were possible to represent all concepts in the output, then this doubt would be eliminated, and it might be possible subsequently to show that the same nature of the concept is exhibited in networks in which only a subset of concepts were included in the output. Alternately, if it were possible to show that the same nature of the concept is exhibited in a sufficient series of networks each implementing a subset of concepts, this doubt would be reduced, though there would still be questions concerning what would count as a sufficient series of conceptual subsets.

The third issue I have is with the potential underdetermination of the system as a whole. In the network in Figure 4, for example, I assigned output node A to represent bacon. What makes this node bacon as opposed to something else? As a result of its training, the system of weights seem to make it bacon, but as noted in the last issue, this determination as bacon is only in contrast to ham or pork chop or whatever happens to be selected within the output layer. Beyond the issue of relativity to the output, there is the issue of the meaning of the network as a whole. In itself, the network is merely data in a dynamical relationship. However, what gives meaning to that data does not seem necessarily within the system itself, but is imposed from outside, from the programmer. The bacon-detecting system is fed raw smells, but these smells take the form of values presented to the system. The system itself does not necessarily take these values as smells, but merely processes them as values. Likewise, the system does not necessarily take values in node A in the output layer as bacon, but merely processes the output as a value. The system seems indifferent to the nature of the data except as mere values. Where this becomes a problem for the comprehension of concepts in connectionism is that an entire system may have the exact same weights and may process the exact same values as another system representing different inputs and different outputs. Suppose that I create and train a network to classify moral acts based on some set of inputs. If it so happens that this network is identical to my bacon detecting network except for the labels that I assign to the input and output nodes, would I then conclude that moral acts have fundamentally the same conceptual structure as pork products? Such a coincidence seems unlikely, but given the wide possibility for inputs and outputs among concepts, it does seem at least possible that some coincidences may occur. In such a case, it does not seem even proper to say that the resulting system of concepts in its hidden layer is even rudimentary, since it is in fact indeterminate, since the meaning of the system is underdetermined by its structure.

These three issues are intertwined with regard to their possible resolution. A particular network may be underdetermined in its conceptual meaning if its output nodes are taken to be simple singular points of output, but if their complexity is taken into account, the chance for coincidence of structure given different outputs is greatly reduced, if not eliminated, thereby increasing the chance that the activity of the system itself results in genuine Intentionality.  Likewise, the underdetermination of the system may be resolved if the outputs are not intentionally selected and assigned by the programmer.  Furthermore, the complexity required in the output nodes may address the way in which concepts arise in ways that are not relative to a complete set of alternative outputs.  These three issues relate directly to the parasitic Intentionality and underdetermination noted in the consideration of classical artificial intelligence.  The projection of Intentionality in the output nodes and the selection of a particular set of outputs as intentional acts makes any apparent Intentionality in the connectionist system parasitic on the Intentionality of the programmer or the interpreter, and the possibilities of underdetermination noted in both approaches are roughly parallel. The central issue here is again the Intentionality of the programmer, both in selecting output nodes that singly represent the programmer’s own Intentional objects and in training the system to replicate the programmer’s recognition of those Intentional objects given the same sensory input.

## Problems with Self-Organizing Networks

Not all networks are trained in a supervised manner.  Consequently, the possibility of unsupervised training of a network may provide a means whereby the Intentionality of the programmer may not pose a problem for the connectionist system.  In such a network, the programmer does not assign the meaning to a particular output node and does not train the network intentionally to produce the desired Intentional output from a training set of data.

Rather, the training within the system is designed to reinforce the selections that the network itself makes.

Consider the revised bacon-detecting network in Figure 5, designed to be a self-organizing network. Initially, the outputs of the system are not assigned any meaning at all. Rather, the raw smell data input to the system activates the initially random settings of the weights, such that a particular smell pattern activates one output node more than the others. The inhibitory pathways across the output layer cause the strongly activated node to reduce the output of the other nodes, explicitly drawn for node B in the example, though it should be understood that each node has similar inhibitory pathways. The unsupervised training of the network reinforces the success of the strongly activated node and the failure of the other nodes by adjusting its weights accordingly. Eventually, this self-training of the network will organize the smells across a number of its output nodes, possibly leaving some nodes as undetermined, as with node D in the example.

The programmer has not intentionally assigned any of the nodes to represent anything, but the system seems to have assigned some meaning to it on its own. Here it does seem that the system has developed its own system of concepts, however rudimentary it may be, and the Intentionality of the programmer did not interfere with it. Yet the simple classification in a single output node does not seem to have any meaning in itself, but rather exists as an abstract distinction in relation to the competing output nodes. “Obviously, it is up to the user to interpret the output” (Picton, 2000, p. 115). In itself, the output is merely a value in a node, and as such has no intrinsic meaning other than that of a mere value. If there is any meaning understood in the output of the system, the meaning derives from the interpretive act of the programmer. With such interpretation, there arises again the

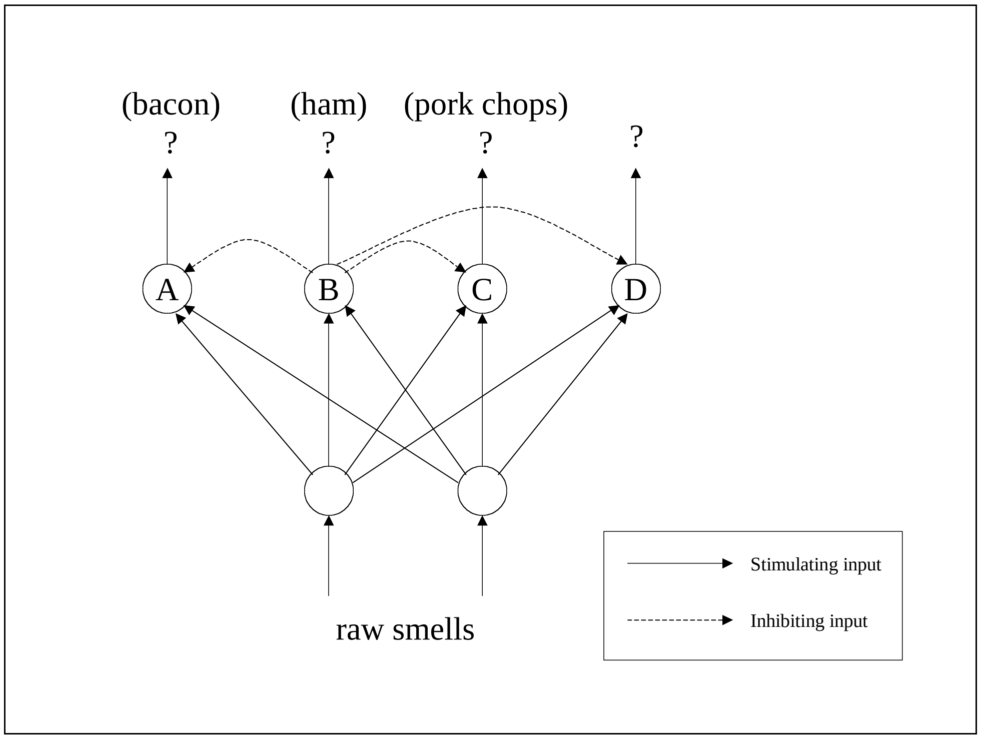

*Figure 5*. Sample self-organizing network classifying smells.

possibility that the Intentionality of the interpreter may compromise the system in some way. Yet if such is the case, it does not compromise the system as egregiously as in classical artificial intelligence or in supervised learning networks, since the system is clearly doing quite a bit of work on its own.

First, I note that there is still potential for underdetermination of the system as a whole. It may happen that two different types of input data may be self-organized in exactly the same way. Unlike the case of underdetermination in supervised training networks, here I might want to say that such a coincidence indicates a similarity in structure between the two sets, because the similarity arose from the system itself, rather than as an accidental artifact of the programmer’s projection of Intentionality. Yet the fact of such a coincidence indicates that the system itself is indifferent to what it is doing, since the inputs and outputs are simply values. I might interpret the system as classifying smells, but that is because I know that the values that I am feeding it are smells and not sounds. Yet for the system itself, there is no immediate difference between values as smells or as sounds.

Second, this problem with underdetermination of the system as a whole leads naturally to an underdetermination of the particular outputs. For example, I understand the output from node A to be a classification of a smell as bacon, not because I intentionally designate the node that way, but because I empirically interpret it as such. If I place a piece of bacon in front of the sensor, I see that node A is registered, which happens every time I place a piece of bacon in front of the sensor and never when I do not. I then conclude that node A represents bacon. If I suddenly change the sensors to register sounds and not smells, without retraining the network, I might in the same way interpret the output from node A to represent a baritone voice. The system itself does not know the difference. In other words, the putative difference in Intentional content is not determined intrinsically by the system, but is interpreted extrinsically by the programmer.

Therefore the attribution of concepts to such a system can only be made in a very rudimentary way, in which the output remains as neutral nodes, as Quine put it in another context (1992, p. 34). The content of such nodes is not given by the system itself, but only the relations between such neutral nodes in the output layer. Yet if such relations were sufficient to determine an Intentional object, then given a case of systemic underdetermination, say between smells and sounds, there would be no conceptual difference between bacon and baritone voices, for example. Since I take it that there is a difference between such concepts, either the bare relations are insufficient to determine the Intentional object or underdetermination is not a possibility. I have been arguing for the possibility of underdetermination, so I think that there must be more to the concept recognized in the interpretation of the output than the system itself is presenting, particularly before any Intentionality is attributed to the system. Without a fuller determination of the concepts, the system would only display as-if Intentionality.

There is a further problem with self-organizing networks that bears directly on the Intentionality of the programmer rather than on the Intentionality of an interpreter. Given a set of sensory inputs, a self-organizing network determines certain features of the data that can be interpreted conceptually. I will ignore for the moment the possibility of underdetermination in the interpretation of the results. The connectionist researcher has no control over what features the self-organizing net organizes. Rather, he must stand back passively and see what develops from the system. On one hand, this necessity provides a measure of objectivity preventing the programmer from biasing the system with his own Intentionality. On the other hand, it restricts his access to a connectionist investigation of certain concepts, particularly higher-level concepts only indirectly depending on raw sensory data. In order to investigate such concepts, the programmer must either pass the network higher order data or must extend the structure of the network. If I choose to pass higher order data, such as stock prices, then the interpretation of such input as stock prices represents a projection of my own Intentionality onto the system in a way that passing it raw smell or sound data would not, since those stock prices exist solely in the context of human social practices, whereas raw sensory data has at least a component that can be considered independent of human practices. Any putative conceptual results would then fall into the same problems with parasitic Intentionality facing supervised training networks. If on the other hand I choose to extend the structure of the network, I risk designing the system with a structure mirroring the structure of my own Intentional objects, which then poses the same problems with parasitic Intentionality facing classical artificial intelligence. If I structure a self-organizing network with the goal of producing higher order representations from raw data, the intentional design of the network seems to represent my understanding of own Intentional object corresponding to that higher-level representation. Here again, my own Intentionality is projected onto the system, this time in the form of network structure. Consequently, I think it would be problematic to claim that the representation within the network corresponds to an intrinsic Intentional object; rather, the putative representation is simply a mirror of the programmer's Intentional object. The challenge in such an investigation is to create intentionally a higher-level connectionist system in which the only Intentionality projected on it is the intention that the system develops Intentional objects on its own.

## Conclusion

I framed this chapter in terms of the need to interpret connectionism with regard to mental representations. This in turn depends upon the recognition of such mental representations within a connectionist system. Given current connectionist practice, I think that it is problematic even to acknowledge such representations in a supervised training network. The Intentionality of the programmer, both in assigning output nodes to stand for Intentional objects as well as conducting the training process to reinforce that projected Intentionality, compromises the status of concepts recognized within the system. Any interpretation of connectionism on the basis of such concepts as representations is an interpretation of parasitic Intentionality inherited from the acts of the programmer and not of pure representations in the system itself.

Although self-organized networks insulate the system from the programmer’s Intentionality to a large degree, their underdetermination both as a whole and in their particular outputs indicate that whatever concepts are inherent in the system remain in such a rudimentary state that they are insufficient to serve as evidence in interpreting connectionism.  There is a chance, though, that the programmer can extend self-organizing networks to address these problems.  The next chapter outlines a program of study whereby this might be accomplished.

# CHAPTER VI

# CONNECTIONISM REFINED

If the Intentionality of the programmer does pose a problem both for classical artificial intelligence and for connectionism, the question remains whether it is an intractable problem or one that can be solved by changes within the respective practices. With regard to classical artificial intelligence, if a solution is possible, it is not immediately clear what form it might take. The difficulty with classical artificial intelligence is that any putative Intentionality within the system seems to be a direct product of the Intentionality of the programmer. For the system to use a concept, the structure of that concept must be mapped out by the programmer in advance, and the use of the concept ultimately reduces to a simple assignment statement representing the acquisition of an Intentional object.

The case with connectionism is appreciably different, since the intentional acts of the programmer affect the putative Intentionality of the system less directly than in classical artificial intelligence. The programmer intentionally creates a system, but does not intentionally create the Intentionality. Rather, the programmer intentionally creates a system in which Intentionality can arise on its own. My argument in the previous chapter was that this goal of self-arising Intentionality in an artificial system is still compromised by the Intentionality of the programmer by the particular practice of assigning Intentional meanings to certain nodes in the system or to the system as a whole, whether intentionally in advance of the network's activity as in a trained network or interpretively after a self-organizing network has performed its self-organization. Given the indirect relation of the Intentionality of the programmer to the putative Intentionality of the system, it does seem possible to overcome the problems that the Intentionality of the programmer poses to connectionism by modifying the questionable practice.

## Proposal for a Corrective Practice

The Intentionality of the programmer seems to enter into the connectionist system in the assignment of Intentional meaning to the input nodes, the output nodes, and to the system as a whole. Yet the interpretation of the system as a whole seems dependent upon the assignment of the input and output nodes. If I assign various pork products to the output nodes of the system, then the system as a whole is interpreted as a pork product classification system. Therefore, the focus for the correction of connectionist practice seems to be the input and output nodes.

For many connectionist systems, the input nodes seem to pose no more of a problem than human sensory input itself poses. While it is possible to input values with pre-existing Intentional content to a system, such as prices of pork belly futures or stock levels of bacon in a grocery store, many connectionist systems start with more basic input corresponding fairly closely to the sensory input that a biological creature might receive. One fanciful example of this was the raw smells input to a self-organizing network that classified the smells of various pork products in Figure 5. A more common example is the replication of visual input through the retina by breaking a picture into component elements or pixels and feeding the light or color values for each pixel into separate input nodes in a network. Another possibility is the replication of audio input by inputting the amplitude of sound waves at various frequencies to separate input nodes. The biology of sensation and perception provides connectionism with fairly uncontroversial ways to input data to a connectionist system without involving the Intentionality of the programmer. Even though that raw input can be understood as an Intentional object by the programmer, such Intentional content need not be recognized as such in the input to the system, but rather the data can be accepted merely as raw input from which the system derives Intentional content. If input can be understood as raw sensory data in a biological system, the same input should be understood the same way in an artificial system. Therefore, it seems that current practice already adequately isolates the input to a system from the Intentionality of the programmer. This much is already gained by connectionism over classical artificial intelligence, in which the input typically must already be grasped as an Intentional object in order for the algorithmic steps in the program to work on it.

Consequently, the assignment of Intentional meaning to the output remains the problematic factor. In a supervised training network, the Intentionality of the programmer is blatantly imposed on the system both by the arbitrary assignment of each node to represent some Intentional content, and by the reinforcement of the assignment by a training process based on training data in which the programmer's own Intentionality is already correlated with sample input data. In a self-organized network, this problem was less blatant, since the system itself made the assignments to particular nodes and reinforced the assignment itself through internal adjustments of its weight values. Yet the problem still arises since the self-organized output is still subject to the interpretation of the programmer, in which the programmer's own Intentionality gets projected down onto the system. A self-organized network structure seems close to achieving native Intentionality, but there is a remaining problem in how to eliminate the need for the programmer to interpret the self-organized output.

This problem seems almost intractable, since any presentation of results by the connectionist system is subject to interpretation by the programmer. What the programmer sees in the system is a network of interrelated values resulting in a set of values in certain nodes understood to be output nodes. Without interpretation, those values simply remain values for the programmer, and therefore challenge the recognition of cognition amid such values. The entire connectionist and neural network project seems reduced to an intellectual curiosity in which values get transformed into other values. Such a curiosity seems comparable to the games that used to be played in the early days of handheld calculators, in which numbers are entered into a calculator and certain rules are given, whether for multiplying or dividing by other numbers. In the end, the calculator is turned upside-down, and the inverted numbers seem to spell out words. There is no point analyzing such apparent words and the inverted numbers representing them or analyzing the numerical transformations that led to such results. The relations between the numbers and the words recognized in the end are mere formal coincidences. Yet in a connectionist system, the correlation between the numerical output of the system and the interpretation seems much more than coincidence. There does seem to be some genuine process within the connections of the network corresponding to cognition. The problem ultimately remains in the form of the output, since it does not seem obvious that any concept naturally results in a single value at a single point.

The solution, it seems to me, lies in the presentation of output from the system in a form that naturally corresponds to the inevitable interpretation that the programmer is going to give to the output. How do I recognize the functioning of any concept within another person? As discussed in Chapter II, I recognize concepts through third-person observation, correlated against first-person recognition, validated against third-person inter-subjective agreement. I observe someone identifying a piece of bacon and recognize that the identification correlates to my own identification, which is validated against the inter-subjective agreement of such identification confirmed in every successful use of the term ‘bacon’ in conversations dating back to the time I was first taught the word ‘bacon’ by my parents. There is interpretation in this process in the first-person correlation, yet this form of recognition of concepts in another person is not typically accused of projecting Intentionality onto that other person. The difference between this case and the interpretation of connectionist systems that I have been criticizing is the nature of the third-person observation and the inter-subjective agreement. What is observed is not merely a value emerging from a single node, and what is agreed upon is not merely that the value from the node will represent a particular Intentional content. To correct the practice of connectionism and to reduce the potentially questionable effects of the interpretation of the output according to the programmer’s own Intentionality, I suggest that the connectionist system must exhibit what another person exhibits, namely behavior.

It should be obvious that this suggestion does not reduce the project of connectionism to any form of philosophical behaviorism, since the behavior in this case is not the focus of the research, but rather a means of isolating the interpretation of the programmer from the underlying cognitive mechanisms. The connections in the network are still the focus of research, since the connections are what underlie the behavioral response to the input. The Intentionality of the system arises because of the connections of the nodes in the network, not merely because of the behavioral response to the input. There is still room for a behaviorist to claim that it is because of the behavioral dispositions which these connections yield that mentality arises, but this is a competing claim to the connectionist interpretation and not a necessary consequence of it. For a behaviorist, the necessary behavior might just as well result from a non-connectionist system, whereas for the connectionist, behavior without the underlying cognitive structure of connections is insufficient for cognition. The methodological demand that the system display behavior enables the connectionist interpreter to analyze those underlying cognitive structures as they natively occur within the system, free from any contaminating influences of extrinsic Intentionality. Of course, the interpretation itself is still Intentional content for the interpreter and therefore may be a projection of the interpreter's Intentionality rather than a recognition of any native Intentionality; however, in such a case, the fault lies with the interpreter and not the programmer, since the system has been created free of the programmer's Intentionality. Such errors are also not distinct to an interpretation of connectionism, but could arise in the same way for an interpretation of human behavior.

The behavior that I have in mind is anything recognizable to the interpreter as a native manifestation of underlying concepts. The interpreter must observe the system doing something, not merely observe the internal functioning of the system. The behavior must be more than the output of values in an Output layer, which I have argued is underdetermined with regard to the Intentionality of the intended output. Consequently, the system would need to be given some physical or virtual presence, such that the interpreter could observe it doing something, so it may need to be given a body and appendages or a simulation of these. Thus the connectionist project of modeling a mind may need to be extended to the task of modeling a complete organism in order to fulfill its goals. Such an organism could well be an instance of as-if Intentionality: the system behaves merely as if it understood what bacon is. What would give this interpretational as-if Intentionality the status of genuine Intentionality is the cognitive structure in the form of the network of nodes and the system of weights that result in that behavior. Thus the conceptual output no longer terminates in a value in a single nodal point, but rather terminates in a complex pattern of actions that depend on certain conceptual structures.

The identification of the exact structures corresponding to the particular concepts thus becomes a matter of empirical research. Such research is certainly more difficult than in previous connectionist models, in which the intentional assignment of Intentional content to output nodes restricted the scope and therefore the size of the system, such that only a relatively small number of hidden nodes needed to be examined in order to identify the putative conceptual structures. Clearly, a system capable of producing recognizable intelligent behavior would need to be much larger than any system illustrated here. There would need to be many hundreds or even thousands of layers needed to produce even the most rudimentary of intelligent behavior, and perhaps even this is an underestimation. Within so large a network, not all of the connections and weights may contribute significantly to the behavior under study, and not all of those that do may represent a particular concept inherent in the behavior. Such empirical research would need to sift through a lot of data, which is inevitably subject to interpretation. Yet such interpretation would no longer represent the Intentionality of the programmer, but rather the same theoretical methodology present in any scientific research, whether into the neurological functions of the brain or into the structure of an atom. The special problems of Intentionality for connectionism disappear, leaving the general problems of science.

One way to achieve this sort of behavior would be to create a connectionist brain within a robotic body, as with MIT's Cog project (Dennett, 1995). However, such a connectionist brain would need to be initially untrained, even with regard to the operation of its robotic limbs, else the preliminary work to make the limbs move correctly risks introducing the Intentionality of the programmer and the engineer as to how such functioning should occur. Consequently, when activated, the robot would very likely flail about randomly, possibly breaking laboratory equipment and injuring the stray researcher. Eventually, it is hoped that the robot could be trained to control its actions, as the system of weights within its connectionist brain become developed, correlating its random movements with the input from its visual field and other sensors. Perhaps such a system could be created with all the behavioral dispositions sufficient for genuine Intentionality already present, but the intentional design of such a system would need to be evaluated to ensure that the Intentionality was not parasitic on the Intentionality of the designers. In any case, such a project would not contribute to the development of connectionism as a discipline.

Another approach is merely to simulate such embodiment within a computer system as a virtual reality world. The connectionist system in this case would be linked to a virtual body and appendages within a virtual world, and certain of its output nodes would be tied to virtual mechanisms enabling it to manipulate those appendages and to maneuver within its virtual world. The virtual world would be presented to the connectionist system by means of visual and possibly tactile input fed to it according to its position and orientation within the virtual world. As with the robotic approach, the connectionist system would need to be initially untrained, such that it could develop its own Intentionality within the virtual world.

Of course, this approach may not succeed. One problem is that researchers may never be able to induce intelligent behavior in such an artificial or simulated organism. A biological organism typically exhibits pleasure and pain behavior, which enables training by means of positive and negative reinforcement. It is not clear what would be required in order for positive and negative reinforcement to occur within a connectionist system. Failing this, the robot or virtual organism may flail about wildly forever without coordinating its actions and sensations in a recognized cognitive structure. Any proposal to incorporate pleasure and pain mechanisms risks reintroducing the Intentionality of the programmer or engineer, and introduces the problem of absent qualia and consciousness.

Another problem is that researchers may never be able to make any sense of the internal data gathered from such a system in order to reach conclusions about the nature of cognition. Data in a connectionist system is multi-dimensional, since at any point in time, there is data dispersed throughout the network of nodes in the form of weights and values transferred between nodes. Furthermore, this data is extended temporally, since these data values exist at every moment in time. The researcher must therefore examine not only the relationships of the weights and values within the structure of the network, but the changes in those relationships across time. Furthermore, this data will need to be correlated between separate instances of relevantly similar behavior in order to identify similarities of internal cognitive structure. The task of data analysis seems quite daunting.

## Benefits

The motivation for correcting certain practices in connectionism was to overcome the problems raised by the Intentionality of the programmer, but there appear to be some further benefits. For example, I think that this suggested correction addresses the criticisms levied against classical artificial intelligence by both Dreyfus and Searle. By situating the connectionist system in a world, whether the physical world or a virtual world, the system need not represent the entire world internally or algorithmically prior to its acting within that world. Both Dreyfus and Searle appear favorably disposed toward connectionism yet express reservations about the current state of connectionism (Dreyfus & Dreyfus, 1990, p. 331; Searle, 1992, pp. 246-247). Perhaps my suggestion for a correction of connectionist practice would advance the field to a point at which both critics could recognize the possibility of genuine intelligence within an artificial system.

Such a system would also provide a level of transparency to the inner workings of a cognitive system that would not be available to empirical research on biological brains, since such research always risks destroying or at least interfering with the normal function to be studied. For any given behavioral action, all the values within the system can be identified and traced. Long-term changes can be traced over time, charting the structure of learning within the system. Of course, current connectionist models exhibit the same sort of transparency, but the abstracted subjects of study within most current connectionist models are much more restricted than the possibilities that are opened in examining all the details of a simulated organism in contact with an environment.

Perhaps more significantly for philosophy, the revised practice of connectionism offers an opportunity to examine empirically the extent and limits of empiricism in general. If a connectionist system is presented as a blank slate to a world, whether virtual or otherwise, if concepts can be induced into the system solely by virtue of its experience with that world, then empiricism as a philosophical doctrine would seem to gain some support. The success of the connectionist system could guide neurological research to demonstrate that humans acquire concepts in the same way as the artificial connectionist system. Of course, such support is empirical support, so there may be some methodological problems remaining there for empiricism as a philosophical doctrine. Still, it may happen that it is discovered empirically that only within certain structures can intelligent behavior be induced, thereby identifying the prior physical grounds for subsequent empirical acquisition of concepts.

Out of such research, connectionism would be able to offer a well-founded interpretation of neural networks with regard to the nature and role of representations within cognition. Rather than defining representations within a theoretical context, the researcher would be able to demonstrate the nature of representations within a connectionist system based on an analysis of the data, possibly even demonstrating that there is nothing properly corresponding to representations at all. With such a demonstration, an interpretation of connectionism would seem much better grounded.

# CHAPTER VII

# CONCLUSION

There is a lot of neural network research that does not aim to simulate human concepts or to produce Intentionality within the system, but rather aims merely to explore new connectionist structures and new learning algorithms. The problems of Intentionality that I raised here do not affect such work, given their goals. The problems become significant when such neural network research is implemented in particular cases and when philosophical interpretations are imposed on such implementations that make conclusions about cognition in general.

My argument here is that any such conclusions cannot be justified so long as the models used in the explanation entail a putative Intentionality within the system that is parasitic on the Intentionality of the programmer. If an argument does not rely on particular models as examples, but merely uses the general principles of connectionism, thereby appearing to avoid the problems of the Intentionality of the programmer, then it seems that the argument has insufficient foundation for making strong claims about cognition and ultimately falls back on the very weak claim that cognition arises in the connections between nodes, as between the neurons in a biological brain. The interpretation of this generalization of connectionism that is needed to understand cognition fully requires specific explication and demonstration of concepts in action, thereby requiring particular models of cognition, and thereby encountering the potential problem that the Intentionality of the programmer poses.

My suggestion for overcoming this problem is to extend the practice of connectionism such that the connectionist system is embodied within a complete organism, whether virtual or robotic, which is studied just as a natural organism is studied, with the additional benefit that its internal structure is transparently available to the researcher without altering or destroying the subject of study. Not only would the connectionist interpretations of neural networks be grounded better using demonstrable data, but also the very approach enables an investigation into the philosophical doctrine of empiricism, thereby providing better grounding for empiricist assertions or criticisms.

Connectionism risks stagnation the same way that classical artificial intelligence seemed to stagnate. After some initial successes, connectionism may not be able to advance further and fulfill its promises, possibly lapsing solely into a commercial endeavor. My identification of the Intentionality of the programmer as a problem for connectionism is intended to isolate a factor that might be responsible for such stagnation. My proposed solution to that problem is intended to chart a path away from stagnation not only toward a fulfillment of the goals of cognitive science, but also toward a revitalization of the debate concerning empiricism in philosophy.

## REFERENCES

Brentano, F. (1973). *Psychology from an empirical standpoint* (A. Rancurello, D. Terrell, and L. McAlister, Trans.). London: Routledge & Kegan Paul. (Original work published 1874.)

Churchland, P. M. (1995). *The engine of reason, the seat of the soul*. Cambridge, Massachusetts: The MIT Press.

Dennett, D. (1995). Cog: Steps towards consciousness in robots. In Metzinger, T. (Ed.), *Conscious experience* (pp. 471-487). Exeter: Schoningh-Imprint Academic.

Dreyfus, H. L. (1997). From micro-worlds to knowledge representation: AI at an impasse. In Haugeland, J. (Ed.), *Mind design II* (pp. 143-182). Cambridge, Massachusetts: The MIT Press. (Reprinted and adapted from Dreyfus, H. (1979). *What computers can't do: a critique of artificial reason*. New York: Harper and Row.)

Dreyfus, H. L., & Dreyfus, S. E. (1990). Making a mind versus modelling the brain: Artificial intelligence back at a branch-point. In Boden, M. A. (Ed.), *The Philosophy of artificial intelligence* (pp.67-88). Oxford: Oxford University Press. (Reprinted from *Artificial Intelligence*, 1988.)

Fodor, J. (1990). Why there still has to be a language of thought. In Lycan, W. (Ed.), *Mind and cognition: A reader* (pp. 282-299). Oxford: Blackwell Publishers. (Reprinted from Fodor, J. (1987). *Psychosemantics* (pp. 135-67). Cambridge, Massachusetts: Bradford Books/The MIT Press.)

Frege, G. (1997a). On Sinn and Bedeutung. In Beaney, M. (Ed.), *The Frege reader* (pp. 151-171). Oxford: Blackwell Publishers. (Original work published 1892.)

Frege, G. (1997b). On concept and object. In Beaney, M. (Ed.), *The Frege reader* (pp. 181-193). Oxford: Blackwell Publishers. (Original work published 1892.)

Gurney, K. (1997). *An introduction to neural networks*. London: UCL Press.

Horgan, T. (1996). Connectionism. In Borchert, D. (Ed.), *The encyclopedia of philosophy: Supplement*. New York: Macmillan.

James, W. (1950). *The principles of psychology* (Vols. 1-2). New York: Dover Publications, Inc. (Original work published 1890.)

Lyons, W. (1986). *The disappearance of introspection*. Cambridge, Massachusetts: The MIT Press.

Newell, A., & Simon, H. (1997). Computer science as empirical inquiry: Symbols and search. In Haugeland, J. (Ed.), *Mind design II* (pp. 143-182). Cambridge, Massachusetts: The MIT Press. (Reprinted from *Communications of the Association for Computing Machinery*, 19.)

Picton, P. (2000). *Neural networks* (2nd ed.). Basingstoke, Hampshire: Palgrave.

Quine, W. V. O. (1992). *Pursuit of truth* (Revised ed.). Cambridge, Massachusetts: Harvard University Press.

Searle, J. (1983). *Intentionality*. Cambridge: Cambridge University Press.

Searle, J. (1990). Minds, brains, and programs. In Boden, M. A. (Ed.), *The philosophy of artificial intelligence* (pp. 67-88). Oxford: Oxford University Press. (Reprinted from *The Behavioral and Brain Sciences*, 1980, 417-24.)

Searle, J. (1992). *The rediscovery of the mind*. Cambridge, Massachusetts: The MIT Press.

Wall, L, Christiansen, T., & Schwartz. L. (1996). *Programming Perl* (2nd ed). Sebastopol, California: O'Reilly & Associates.

## ABSTRACT

## ABSTRACT

Connectionism seems to avoid many of the problems of classical artificial intelligence, but has it avoided all of them? In this thesis I examine the problem that Intentionality, the directedness of thought to an object, raises for connectionism. As a preliminary approach, I consider the role of Intentionality in classical artificial intelligence from the programmer's point of view. In this investigation, one problem I identify with classical artificial intelligence is that the Intentionality of the programmer seems to be projected onto the system, rather than the programmer creating a system whereby Intentionality arises intrinsically within the system.

In considering the current practice of connectionism, the same problem with Intentionality reappears. The assignment of Intentional content to input or output nodes in a neural network likewise projects the Intentionality of the programmer onto the system, and that projection is often reinforced by the training process of the neural network. However, connectionism seems to have an advantage over classical artificial intelligence in this respect, in that there is a form of neural network in which the network itself organizes the output nodes. The challenge is to utilize these self-organizing networks without having the programmer project Intentionality onto the system in an act of interpretation.

I suggest that the way to overcome these problems in connectionism is to embody the neural network within a world, whether physical or virtual, and to allow the system to develop concepts purely empirically. In this manner, connectionism can make scientific observations concerning the nature of cognition without the risk of contamination from the programmer's own Intentionality.

## ABSTRACT OF THE THESIS

Connectionism and the Intentionality
of the Programmer
by
Mark R. Ressler
Master of Arts in Philosophy
San Diego State University, 2003

Connectionism seems to avoid many of the problems of classical artificial intelligence, but has it avoided all of them? In this thesis I examine the problem that Intentionality, the directedness of thought to an object, raises for connectionism. As a preliminary approach, I consider the role of Intentionality in classical artificial intelligence from the programmer's point of view. In this investigation, one problem I identify with classical artificial intelligence is that the Intentionality of the programmer seems to be projected onto the system, rather than the programmer creating a system whereby Intentionality arises intrinsically within the system.

In considering the current practice of connectionism, the same problem with Intentionality reappears. The assignment of Intentional content to input or output nodes in a neural network likewise projects the Intentionality of the programmer onto the system, and that projection is often reinforced by the training process of the neural network. However, connectionism seems to have an advantage over classical artificial intelligence in this respect, in that there is a form of neural network in which the network itself organizes the output nodes. The challenge is to utilize these self-organizing networks without having the programmer project Intentionality onto the system in an act of interpretation.

I suggest that the way to overcome these problems in connectionism is to embody the neural network within a world, whether physical or virtual, and to allow the system to develop concepts purely empirically. In this manner, connectionism can make scientific observations concerning the nature of cognition without the risk of contamination from the programmer's own Intentionality.