Notes for the Development of a Philosophy of Computational Modeling

Terrence C. Stewart [1] <terry@ccmlab.ca>

[1] Carleton University Cognitive Science Technical Report 2005-04. http://www.carleton.ca/ics/TechReports

Introduction

In cognitive science, we study the most complex system that science has ever examined. Cognizing creatures are unimaginably more complex than anything seen in the physical sciences. To come to terms with this, cognitive scientists have been forced to carefully examine traditional scientific techniques, modify them, and develop new ones more suitable to this daunting domain.

This process has not been without controversy. The question of how cognitive science should be done is one that will continue to be discussed for many years to come. However, some approaches are fairly broadly accepted as ones that will be key to the discipline. One of these is cognitive modeling.

Cognitive modeling is simply the idea that in order to understand the mind, we need to develop models of the mind. These models should be structurally and causally similar to real minds, and so can provide an explanation of the observed cognitive behavior. By developing more complex and more accurate models, we can explain and predict more aspects of the intelligent behavior of living organisms.

Indeed, it can be said that it is this process which distinguishes cognitive science from earlier approaches to the study of the mind:

> “In a reaction against the subjectivity of armchair introspectionism, behaviorism had declared that it was just as illicit to theorize about what went on in the head of the organism to generate its behavior as to theorize about what went on in its mind. Only observables were to be the subject matter of psychology; and, apparently, these were expected to explain themselves” (Harnad, 1990).

Eventually, the methodology of simply relating stimuli to responses was found to be inadequate. In the classic water maze experiments, for example, rats who learn to navigate a dry maze by running can clearly still make use of this knowledge when the maze is flooded and they must swim, even though the stimuli and the responses are wildly different. Developing an explanation of this behavior must involve some discussion of what is “going on” in its head. Whether or not we must also discuss what goes on in its *mind* is a separate, and much more controversial, issue, and is not relevant for this paper.

In any case, we face an interesting problem if we are to develop scientific theories of what goes on inside the head of a cognizing organism. This problem is that *the only data we have is the stimuli and the responses.* These are the only “observables”. So how can we develop accurate models of a cognitive system when all we can see are the inputs and outputs?

This is the basic challenge for cognitive modeling. Much has been written about this topic, how it should be performed, and even whether or not it is valid science (Simon & Wallach, 1999). For the purposes of this paper, we shall assume that it is a useful and possible approach to the understanding of the mind.

Our main focus in this paper is to examine in detail one particular (and arguably dominant) type of cognitive model: the computational model. The argument is that this sort of modeling addresses a number of the general modeling concerns in a unique way, and also raises a few new concerns of its own. The plan, then, is to describe how computational modeling of cognitive phenomena *should be done,* in a manner that is both practical and scientifically sound.

It is the view of the author that computational modeling, if done right, can form a solid foundation for cognitive science as a whole (and, indeed, for other sciences which deal in complex and opaque phenomena). Computational modeling will be a tool as fundamental as mathematics is to the physical sciences, but only if we are careful about our methodologies and interpretations. The rest of this paper argues that successfully making use of computational models involves the use of non-standard statistical techniques, new computational tools, and a willingness to cross modeling boundaries.

Explaining versus Predicting For some researchers, the goal in creating cognitive models is to *explain* cognitive phenomena. For others, the goal is to build models that will *predict* cognitive phenomena. These two goals can also blur together somewhat, making it difficult to determine exactly what we want out of our models.

In this paper, these two goals are considered to be convergent. Developing an explanation for a cognitive phenomenon (or, indeed, for any account of the world) involves the creation of a set of tools (ie a 'framework' or a 'paradigm') which directs us in forming models which give scientifically accurate predictions of the phenomena in question. This idea is found explicitly in the work of Giere (1999, 1988), and is strikingly similar to Dennett's (1991) approach to epistemology.

Defining what is actually meant by a 'scientific explanation' is an ongoing project in the philosophy of science. However, all views seem to concur that a vital role is played by accurate causal account (ie models) of the situation at hand. Furthermore, much importance is placed on determining the explanatory scope (or “level of explanation) of a model: what range of behaviour, what temporal and/or spatial grain size, or what level of explanation it is suitable for. For an introduction to these issues, see (Woodward, 2003).

Given the fact that, whatever view on explanations one takes, one still needs predictive models to form explanations, this paper will focus on the problem of how one develops such predictive models. Some researchers (including the author) may believe that this is sufficient for an explanation (as will be argued later), but at the very least it is necessary. We will also examine the role of the level of explanation of these models as we discuss the role of using data from various different domains to test our models.

It may also be helpful to remember that even defining what exactly an 'explanation' is can be difficult, and our intuitions can be misleading. Lombrozo (2005) showed that people are quite willing to accept explanations based on an adaptationist account, even

though they themselves have little understanding of the basic components of how an adaptationist explanation should work. Ofttimes merely labeling something as an explanation is sufficient for us to believe we have an explanation, rather than basing such a decision on the rigorous predictive abilities of the model.

Laws versus Principles In *Science Without Laws,* Giere (1999) argues that science must not be trapped into only looking for universal 'True' Laws of Nature.

> Rather than thinking of science as producing sets of statements that are true or false in the standard objectivist fashion, we should think of it as a practice that produces models of the world that may fit the world more or less well in something like the way maps fit the world more or less well. (Giere, 1999, 241)

The basis of this approach is to interpret the key discoveries of science as *principles* rather than laws.

> Principles, I suggest, should be understood as rules devised by humans to be used in building models to represent specific aspects of the natural world. Thus Newton's principles of mechanics are to be thought of as rules for the construction of models to represent mechanical systems, from comets to pendulums. They provide a *perspective* within which to understand mechanical motions. … What one learns about the world is not general truths about the relationship between mass, force, and acceleration, but that the motions of a vast array of real-world systems can be successfully represented by models constructed according to Newton's principles of motion. (ibid., 94-95)

It is this view that we will be taking within this paper. The goal in our endeavor is to describe a methodology for using computational models such that we can discover *principles* which will allow us to create predictive and explanatory models of real-world cognition.

Computational Models in Science Before going further, it is useful to more precisely establish what is meant by the term *computational model.* We first examine some important theoretical distinctions to be made, and then give a more rigorous definition. This is then followed by a number of points of scientific methodology in their use.

Three Distinctions

In the following sections, we explore three distinctions which serve to distinguish this sort of model from the more generic use of the word.

Computational Modeling versus the Computational Theory of Mind We start with a distinction that acts as a negative definition. The Computational Theory of Mind (Putnam, 1961) makes the strong (and widely believed) claim that the mind

simply is a computer. We can think of cognitive processes as being computation: data is read, rules are followed, and actions are taken. This 'Computer Metaphor' has become a basic assumption of all of cognitive science, and forms the basis of a common language of communication among cognitive scientists.

However, this is *not* what is meant by cognitive modeling. One can make use of the Computational Theory of Mind without ever actually using a computer; for example, Fodor's research program involves exploring the theoretical limits of potential behavior from a computational system, all without actually writing any actual computer programs. Indeed, the idea here is that we use the idea of a computer to allow us to visualize and understand how the inner workings of something as complex as the mind could be organized. It gives us a theoretical framework from which to tackle such a multifaceted phenomenon.

However, as we shall see, the process of computational modeling is intimately tied with the creation of implemented computer programs that are meant to *model* (ie to perform in a manner which is in some sense equivalent to) the actual mind. Of course, the key question is to determine what is meant by 'perform' and 'equivalent to'. These issues are the core of the rest of this paper.

Before continuing, however, it is worth noting that it is possible to argue that computational modeling can be done *without assuming the computational theory of mind.* If computational modeling is simply the idea that a particular computer program can act similarly to some aspect of the mind, then this is independent of the claim that the mind *is* a computer. This sort of claim is similar to that made by Searle (1980), and forms the basis of the argument that simply making computer programs which act the same as real minds might not tell us anything interesting about real minds. One of the goals of this paper is to convince the reader that, far from being merely an interesting technical feat, the development of computational models does, in fact, give us insight into and a deeper understanding of real minds, and that this understanding would not be possible in any other way.

Fully Specified Models versus Black-Box Models We also need to distinguish computational modeling from a common partial approach to modeling. In cognitive psychology (and in much of the rest of psychology), it is common to find 'boxes-and-arrows' diagrams depicting individual theories as to the processes underlying a particular cognitive phenomenon. These diagrams show separate components of the mind and how they communicate, enabling us to see how the system as a whole would exhibit the behavior in question. We can make use of such a diagram to predict what would happen if, for example, a particular component was damaged, or if two related tasks were attempted at once.

An important feature of these diagrammatic models is that they *leave the internals of each component unspecified.* That is, they describe how the system as a whole works, but not how the individual components work. For this reason, we call them *black-box* models (meaning that the internals are hidden from view). Another feature of these

black-box models is that they allow for *qualitative* predictions, but not *quantitative* predictions. Without knowing how the individual parts work, we cannot determine how quickly the system as a whole will work, or how accurate it will be, or exactly what responses it will give.

The alternative to the black-box approach is to give an explanation as to how each of the components in one's model functions. Each of these components can be broken down into smaller components, which can be further broken down. To avoid infinite regress, we need a set of primitive components that all others will be built from. Primarily, computational modeling uses computation to ground its components. Thus, *every component in a computational model must be a fully specified computer program.*

One can, of course, object to this idea. What if some part of the mind is not computable? What if Fodor (2005) is right in that "we need a new notion of computation … that perhaps differs significantly from Turing's"? This is, indeed, an empirical question. It may be that the human mind makes use of some quantum mechanical trickery, or indeed some other heretofore-unnotified system. As unlikely as we may believe this to be, it is possible. However, for the purposes of current computational modeling, this possibility is irrelevant.

The reason it is irrelevant is that the point of using a fully specified computer program is to *generate quantitative predictions.* Right now, any system capable of giving a quantitative result is itself Turing-computable. If, in the future, new tools are discovered, then certainly computational modellers will make use of them (after all, they are 'computational modellers', not 'Turing-computational modellers'). But, in the absence of such tools, we are not limiting ourselves by only considering the sort of computation that we can currently perform.

Indeed, we can see the use of computer programs as having the same role in cognitive science that the use of mathematics has had in the development of physics. Much of early physics had exactly the sort of qualitative predictions that is found with black-box modeling. The creation of the mathematical formulation, however, immediately led to powerful exact quantitative predictions. It is this transition that we hope to achieve with computational modeling.

Furthermore, computational models offer two vital advantages to the cognitive scientist. First, they force you to completely specify a theory and all of the assumptions it requires, "including aspects which might otherwise be overlooked" (Cooper et al, 1996). There are no hidden assumptions in the chain of logic which leads to the predictions: all of the assumptions are built into the code of the computer model itself. The necessity of this has long been known within cognitive psychology:

> One of the chief benefits of the model-based research approaches is the requirement they impose on the researcher to be explicit about assumptions concerning the factors that are involved in performance and the ways in which these factors combine to produce the observed responses. Once such assumptions

> are out in the open, they can be evaluated for psychological plausibility or put to empirical test. When the assumptions concerning the (possibly multiple) causes of underlying test performance are not explicit, Group x Task interactions may mislead the researcher about the foundations for her process-related inferences (Cole & Means, 1981, 135).

Secondly, computational models have an advantage over other sorts of models (be they mathematical or rule-based or otherwise), in that they can *produce the same sort of data as the real subjects.* That is, the same statistical analysis can be performed on both the raw observed data and on the raw data from the model. This greatly simplifies the task of ensuring that we are measuring the same thing about both the model and the real situation.

Iterative models versus Mathematical models

If a computational model is supposed to perform the same sort of quantitative role that mathematics introduced to physics, then what extra is to be gained by using a computational model? Why not use mathematical models within cognitive science?

Indeed, there are a number of mathematically specified cognitive models, such as the Rescorla-Wagner model of classical conditioning (Rescorla and Wagner, 1972). This expresses the change in learned associations between stimuli as a set of differential equations, much as would be expected of a theory in physics. Furthermore, it does give the sort of quantitative predictions that we desire.

To see why we would want to go beyond mathematical modeling, consider this passage on the topic of using Newton's mathematical physics laws to predict the motion of the planets:

> Supposed that at a given time the position and velocity of a planet can be determined, and that the force is known. Then, according to Newton's laws we know the change in velocity during a short time interval. Knowing the initial velocity and its change, we can find the velocity and position of the planet at the end of the time interval. By a continued repetition of this process the whole path of motion may be traced without recourse to observational data. This is, in principle, the way mechanics predicts the course of a body in motion, but the method used here is hardly practical. *In practice such a step-by-step procedure would be extremely tedious as well as inaccurate.* Fortunately, it is quite unnecessary; mathematics furnishes a short cut, and makes possible precise description of the motion in much less ink than we use for a single sentence. (Einstein and Infeld, 1938, page 31, emphasis added)

This highlights the two possible ways to use a mathematical model to predict the outcome of events. The mathematics of Newton's laws specifies how the positions and forces on the planets (or any other body) will change over time. Thus, if we know where the planets are now, we can determine the forces exerted on each planet at this moment. We can then take these forces and determine where the planets will be a (very) short time

from now (say, one second). Given this new position, we can calculate the new forces, and then the new positions two seconds from now. Importantly, Newton's laws are only accurate if we allow this 'time step' to become extremely small. As is pointed out in the quote above, this means that if we want to predict what will happen many years from now, we would need to *iterate* this procedure thousands or millions of times, making it "tedious" and "inaccurate".

It is this problem which leads to the alternate approach to using mathematical models. The invention of the Calculus (independently by Newton and Leibnitz) was done specifically to allow all of these millions of intermediate steps all at once. This is the "short cut" alluded to in the quote above. In certain situations, it is possible to implicitly perform the millions of steps all at once, producing a *solution* which specifies exactly where the planets will be at every point in time. Thus, if we know the current position of two planets, we can use this mathematical trick to very quickly indicate exactly where the model predicts the planets will be at every point in time in the future. This is an astounding ability, and gives strikingly accurate predictions.

Unfortunately, this process is dependent on the mathematical model of being *tractable.* It turns out that if you want to do the same trick in a situation where there are three planets, in general *it is impossible to solve the equations* (This is the infamous three-body problem). The mathematical short cut does not work, and we are left with having to do the predictions the long way. We have to iterate through millions of steps in order to make our long-term predictions. This is such an impossible process that, later in the same book, Einstein and Infeld indicate that in Newton's theory "only big steps connecting distant events are possible" (page 153). That is, they are implicitly ignoring the idea that Newton's laws could be used in this small-step manner, as it is simply too complicated.

However, that was written in 1938. In modern times, we have a new tool which is *specifically designed to accurately perform exactly this sort of repetitive and tedious task.* Indeed, that is what the term 'computer' means. , this means that if Newton had had a computer, *he would not have needed to invent the calculus.* We can use a computer to perform exactly those repetitive tasks for which he needed a mathematical short cut. Furthermore, we can use a computer to give numerical predictions for situations where the mathematics are intractable. We do not have to limit our studies to the domain of problems which happen to be tractable.

This line of reasoning clearly also applies to cognitive models. Indeed, given the complexity of cognitive models, it is extremely unlikely that any of them will be mathematically tractable. But by making use of an iterative computational model, we can gain all of the quantitative benefits of the mathematical approach, without limiting ourselves to a particular sub-domain of solvable problems.

It should be noted that mathematical techniques can certainly still be a useful tool within computational modeling. For example, in (Stewart, 2000), I use basic calculus to show that the Distributed Adaptive Control learning rule (Verschure et al, 1992) is incapable of

extinguishing associations. Such mathematical short cuts are extremely useful when found, but they will not always exist (or, perhaps equivalently, they will not always be findable by the researcher). We should thus not be dependent on this approach, but should make use of it when we can.

What is a Computational Model?

Given the distinctions described in the previous sections, we can now define what we mean by a computational model. Quite simply, it is *any* implemented computer program used to generate data that will be compared to some real-world behavior. “In this way the implications of the total rule set can be unambiguously derived and compared to the [subject]'s actual behavior” (Klahr, 1976).

It should be noted that it is debatable whether *all* uses of computer programs within cognitive science fall under this definition. For example, what should we make of the work investigating the evolution of cooperation strategy in the Prisoner's Dilemma (Axelrod, 1987)? Or of the studies in how simple, local rules can lead to the emergence of complex group behavior (Reynolds, 1987)? There, we are using computer programs, but the comparison to real behavior is qualitative rather than quantitative.

Whether one wants to call this qualitative comparison 'computational modeling' is a matter of definition. Some may prefer to call it 'theoretical biology' or 'artificial life'. In any case, qualitative comparisons *are not* the topic of this paper. Here, we will only be dealing with computational modeling for the purpose of quantitative comparison.

In any case, the key question is still what makes for a *good* computational model. This question is intimately tied to the issue of how we, as scientists, are to make use of such computational models, and so will be addressed via the rest of this paper.

It may seem that some general statements about how we expect a useful computational model to look can be made. For example, it is likely that all computational models will make extensive use of iteration, as argued in the previous section. They will also need to sense (have inputs) and act (have outputs) on some environment (which can also be a computer program). However, these general statements also apply to *all* computer programs, and so do not appear to be useful.

On Parameters and Model Fitting It

is also worth noting that this definition has a subtle distinction from many typical uses of the term *model.* In particular, a computational model is a *fully-implemented* computer program. This means that *there are no free parameters.* This choice of definition has some useful consequences.

In many discussions about modeling, the general term *model* is given to systems that do not specify specific internal parameters (such as how quickly a particular value changes, or the exact strength of association between to components). These are left as parameters which are to be adjusted so as to give the 'best fit' of the model to real-world results.

Continuing with our example of predicting planetary motion, these parameters would be values such as the universal gravitational constant (G). In physics, numerous experiments consistently show that the best fit between the mathematical model's predictions and the real world occurs when the value of G is set to be 6.673ÿ10-11 m3 /kgÿs2 . The hope for most modellers is to find similar strong, consistent results for their parameters.

This then leads to a common problem with computational modeling: parameter fitting. In particular, it is well known that, given enough parameters, one can fit certain sorts of models to *any* given data. Furthermore, this parameter twiddling seems to be something undesirable in a scientific inquiry. We want to predict outcomes, not use outcomes to determine what the parameters should have been. This is particularly problematic since many computational models in cognitive science seem to *require different parameters for different situations.* In other words, we can determine the parameters which fit a given situation, but cannot determine a priori what the parameters are for a new situation. This is clearly not what we want in a useful model.

This situation is remarkably similar to that of Ptolemy's Epicycles. This system for predicting the future path of the planets (circa 150 AD) consists of layers on layers of circles guiding the planets' motions around Earth. The precise configuration of these circles was determined by parameter fitting: finding the one set of parameters which best fit the observed motions. This system actually worked quite well for predicting future astronomical events (indeed, slightly better than the Copernican model which replaced it). However, if new astronomical bodies were discovered, there was no way to use the model in the new situation. It required a different set of parameters for every planet, and there was no a priori method for obtaining those parameters.

It is this sort of problem that leads to our recommendation of not calling something a computational model *unless every parameter is specified.* This would mean that there could be a (rather large) *family* of models, all of which have the same basic structure, but all of which have different parameter settings. Indeed, this family of models can be infinitely dense; we can imagine two models whose parameters differ by only an infinitesimal amount. However, for the purposes of this paper, it is important to think of each of these as *different models.*

Sometimes, it is useful to refer to the general family (or, equivalently, the *set* or *type)* that a model is a part of. However, care should be taken in assuming that the members of a particular model family will behave similarly. Sibley and Kello (2004) demonstrate clearly that gradual adjustments to a single parameter in a relatively simple computational model of learning can lead to distinct qualitative shifts in the overall behavior of that model. It is due to this potential that we wish to maintain our terminology that any two implemented models whose computer code differs by any amount should be seen as different computational models.

This definition changes the way we view model fitting. In particular, it means that model fitting does not mean *adjusting a model until it fits a particular situation* (and then

adjusting it again to fit another situation). Instead, it means the search for a particular 'good' model (or 'good enough' model) among the space of potential models within a particular family. Importantly, it is worth remembering that not all model families will even have a model which matches the desired data – the search may, indeed, be hopeless. This definition also means that if we require different parameter settings for two different situations, then what we are proposing are *two different models.* If we want a general model which will cover both situations, then we need a model which handles both situations *without changing any parameter settings.*

On Comparative Studies

Given that useful computational modeling is going to involve comparing model performance to real-world performance, it is worthwhile to take a moment to examine how this sort of comparative process is done in other sciences.

In most experimental situations, we identify some aspect of the situation that we want to adjust (the independent variable), then divide our subjects randomly into a control group and an experimental group, and then perform the experiment and collect a dependent measure. Much of this process hinges vitally on the randomness of the division into the two groups: this is what ensures that there is no consistent bias in the groups that may cause us to erroneously conclude that the dependent variable caused any observed performance difference.

However, in our situation with computational modeling, we are comparing the performances of two *fundamentally different groups:* the models and the real cognitive organisms. We thus need to pay considerable more attention to the process of comparison that we use. We cannot simply say that the measurements we make in our model can be directly compared to the measurements that we make in the real world. Importantly, *the experimental environment for the two are completely different.* We do not bring our computational models into a room, get them to sign an informed consent form, and then get them to fill in questionnaires and read instructions and push buttons in response to stimuli. So how can we really say that the data is comparable?

This is not a new problem. Indeed, this is exactly the standard problem faced in cognitive psychology when comparing the performances of people from different groups. Cole and Means (1981) offer a book full of a variety of methods for dealing with exactly this problem in cognitive psychology. Admittedly, it is focused on interpreting performance *differences,* rather than on showing equivalence, but many of the points and methodologies are still relevant for computational modeling. In particular, they identify two types of problems that occur when “all other things are not equal:”

> (1) Our comparison groups will generally differ on not one but a multitude of dimensions. Hence, we will have difficulty in proving that any one particular difference between our groups is the source of the observed differences in performance. (2) We will not be able to ensure that our treatment is equivalent for different groups at all points in the experimental task. If our groups are, in a

> psychological sense, receiving different treatments, the interpretation of observed group differences is further clouded. (Cole & Means, 1981, 35)

The first of these types of problems does not concern us greatly. Our “comparison groups” are the model and the real organism, and we are modeling all aspects of that organism. Admittedly, it does mean that we tend not to be able to model *just the one aspect of the cognitive behavior we are interested in.* That one aspect may not be separable from the rest of the behavior of the organism. However, this is an empirical question, and the process of creating and testing models is what will lead us to identifying what aspects of behavior are due to a single underlying mechanism. The main way of dealing with this problem is simply to generally be aware that other factors are likely to come into play when modeling one particular cognitive behaviour.

The second problem of comparison is directly applicable to our computational modeling situation. When comparing two different human populations, it is certainly likely that one may have to have *different* stimuli. For example, when using a questionnaire to compare English and Chinese populations, *the questions must be translated.* That is, the 'appropriate' stimulus is different for the two groups. The same situation is going to happen when comparing models to real organisms (or, at least, this will be the case until we get computational models complete enough to read the questionnaires for themselves). As Cole and Means point out, “there is reason to suspect the most standard experimental procedures are neither equally motivating for, nor equally well understood by, college students, preschoolers, retarded adolescents, schizophrenics, and Austrailian tribsmen” (Cole and Means, 1981 , 44).

In one sense, this is an unsolvable problem. How can we modify experimental stimuli so that our computer model receives it in a manner equivalent to the real organism? Does this mean that any comparison is hopeless until we have a complete cognitive model of everything?

Such a conclusion would doom any initial attempts at modeling. Instead, a more productive approach is to be more explicit about this comparison. To a certain degree, choosing how this comparison is made is as important as the model itself, and it tends to get significantly less attention. Cole and Means specifically point out that this comparison can go awry in any and all aspects of an experiment, and specifically break it down into looking at the task materials, the provided instructions, and the actions requested. We can adopt this list to help us be more explicit about the hidden assumptions in our comparison.

The task materials are the clearest of the points of comparison. The particulars of the experimental stimuli (words on flashcards, colored lights, visual patterns, and so on) are going to be very different for humans and for computational models. Common analogues include using a particular input in the computer model to correspond to a particular letter in the human stimuli. Even in models of mammalian vision it is common to give the visual input as a grid of inputs representing a static picture, rather than the quickly changing array of data generated by the human eye jumping in sudden saccades all

around a scene. This is not, per se, a problem, as long as *we are explicit about the fact we are doing it.* When we avoid the object recognition problem by giving our models simple, elemental stimuli, we are trying to focus on other aspects of the behaviour. Of course, this may not be a successful idea, it is quite possible that by simplifying the problem in this way, it may actually be *harder* to create predictive models. But, some simplification is always necessary – we just need to be clear about what such assumptions are being implicitly made by the choice of stimuli.

Turning to the task instructions, we find a more interesting problem. With adult human subjects, it is relatively easy (although not trivial) to create verbal or written instructions in their native language about what they are to do. With infants or animals, we use other, well-established 'motivations' to set up situations where they will (hopefully) choose to do what we want them to (although this is also a potential source of controversy). With computational models, the situation is different. A model is often set up explicitly to do one particular task. This, on its own, is reasonably uncontroversial, as it can be argued that what we are modeling is the activity of the agent once it has understood the instructions and decided to do what we have asked. However, when a model gets used for more than one situation (as successful models will have to do), we often need some way of 'telling' the model that its situation has changed. Sometimes, this is done by having certain parts of the model be inactive at some times, and then the experimenter activates them back on when required. For example, it is common with neural network models to have the learning mechanism 'turned on' at certain times and 'turned off' at other times. In such situations, it needs to be pointed out that this adjustment *is not part of what is being modeled.*

Finally, we have the procedure itself: the thing the model is being asked to do. In comparisons between human groups, it is argued that this is important because different subject pools may be differently familiar with the task in question. For example, Western school-children are substantially better at tracing a pattern with pencil and paper than Zambian school-children. But, since this performance difference disappears in a task such as copying a pattern using modeling clay, we should conclude that Western children are more familiar with the tracing task than Zambian children, and we should not conclude that there is some internal cognitive ability which makes the Westerners better at tracing. It is difficult to know how to see this difference in the situation of computational modeling. One way is to note that our computational models tend to be extremely specific to particular sorts of situations; none is as flexible and general as the human mind. From this perspective, our models are *much more familiar* with the task at hand than the real subjects are. After all, *all the models ever 'experience' is the performance of the task.* On the other hand, the living subjects can apply a vast array of previous knowledge of other, possibly relation situations to the task at hand. In this sense, they are the more experienced.

Overall, then, the conclusion is that these sorts of comparison issues need to be included in the presentation and discussion of any model. They help us clarify exactly what is and is not being modeled, and they help point out key differences that may be responsible for

some modeling discrepancies. By making them explicit, we can question them, and improve and expand our models.

Identifying 'Good' Models Now that we have established that one of the major tasks to be performed in computational modeling is the identification of sets of 'good' models that 'fit' some particular real-world behaviour, it is time to take a closer look at what we mean by 'good' and 'fit'. What does it mean to say that a given computational model gives results that are 'good'? Simon and Wallach (1999) comment that “the question as to what qualifies as a good approximation has only pragmatic answers”, so what pragmatic measures can we follow?

In the physical sciences, this connection is established by comparing the numerical predictions of the mathematical model with the observed data from the real experiment. These two values will clearly never be *exactly* the same, but if they are deemed to be 'close enough', then we can accept the model. We now, of course, need to determine what we mean by 'close enough'.

Consider what would happen if we had a computational model which was a *perfect* model of the real system. Even if the model was an exact representation of the 'real' world, there would still be sources of variability which would cause the model data and the real data to differ. For a more in-depth discussion of what it means to be a perfectly correct model of a 'real' system, and its implications for philosophical realism, see the section on Truth and Reality, as well as (Giere, 1988, 106-110). For now, we will discuss the implications of two sorts of variability: measurement error and internal variation

First, there is measurement error: no observational technique is perfect. In physics, we may measure that a rock took 12.4 seconds to fall to the ground, while the actual time might be closer to 12.3895843. In cognitive science, we may measure that people choose option A over option B 68 percent of the time, while the 'actual' number may be closer to 54,382 (due to sampling and sample bias). Fortunately, we are used to dealing with this: most statistical techniques include a 'confidence interval'. This is a mathematically provable statement that some desired value is within a certain range a certain percentage of the time (if we assume that the data has a certain distribution). Without getting into the details of such a process, it is in any case common to treat measured values as ranges, and we shall keep this in mind.

Second, there is an internal variation: the behaviors of interest to cognitive scientists *tend not to be exactly repeated.* To a certain degree this is true in the physical sciences as well: when one measures how long it takes a rock to fall from a certain height, there is always a small variation in the initial height that the rock is placed at, and so there is always a slight difference in the actual time taken. Fortunately for physicists, it turns out that for most of the situations they are interested in, this variability in *initial state* turns out to be very small, and furthermore has very small impact on the final outcome.
There are physical systems where this is not the case. For example, in the three-body problem (discussed previously as a situation where the mathematical models

become intractable), infinitely small changes in the exact initial configuration lead to drastic changes in the eventual paths of the planets. This sort of system falls into the general category of a *chaotic system:* a system governed by rigid, deterministic rules, but where small initial changes generally lead to large changes in the final behavior.

In the cognitive sciences, most of the situations we are interested in are likely to have this problem. No matter how tightly we control the exact initial situation, people are going to behave differently. Even if we could have two identical twins, raised since conception in separate artificial wombs kept in exactly the same conditions, exposed to exactly the same environments to be raised in, and then placed into the exact same experimental situation, *they could still give different results .* At first glance, this seems to be a crushing blow to the idea of replicability within cognitive science (or, indeed, within any social science as well). How can our models reliably give the same results in a given situation if the situation itself doesn't give the same results?

The solution to this problem is to notice that if the real situation has a certain degree of variability due to factors beyond our control, *then the model should have the same degree of variability.* That is, we are not interested in modeling the *average* performance – we are interested in modeling *individual* performance. Indeed, given any task with a bimodal performance distribution, there may be *no individuals* who perform at or near the average. Clearly, any model which gives the same average but without a similar distribution is not a good model of individual performance. In general, this is a very difficult problem, since characterizing the distribution of the real-world performance can be a highly ill defined problem. At the bare minimum, we should make use of the mean and standard deviation, and be on the lookout for highly non-normal distributions.

Given this, we need to measure the variability of or cognitive situation of interest. Certainly, as we learn to control different relevant factors, we may change how variables the real situation is, but this simply means that our models of the new, controlled, situation must take that into account. However, this now means that we will generally be working with models with a large degree of internal variability, meaning that in order to evaluate them, *we have to run each model many times.* Only in this way can we get an accurate measure of how variable the model is, so that can be compared to the real world data.

Given these two sources of variability, we need a more flexible definition for what it means for a model to 'fit' a given situation than a simple numerical comparison between the output of the model and the real data. Fortunately, there is a relatively obscure mathematical technique from statistics which nicely fits our purposes: Equivalence Testing.

The Equivalence Test

Equivalence Testing is, in some senses, the exact opposite of the common t-test. The t test is used to determine if two sets of varying data are *different.* It does this by assuming that the two sets of data are being generated by the same varying system, and then proceeding to determine how likely it is for the observed outcome (or any other, even

more unlikely outcome) to have happened. This probability is the *p-value* given in many research papers; if p is 0.05, then we know that only 5% of the time are we wrong in saying that the two sets of numbers are different.

The t-test, at a first glance, seems to be exactly what we want. We have the data from the real-world, which usually takes the form of a list of numbers representing the behavior of each person (or animal, or group, or any cognizing organism of interest) on whatever experimental situation is being considered. We can also generate a similar list for the computational model by simply running the model multiple times. Now that we have these two sets of numbers, we can apply the t-test, and get a result. If we find that $p<0.05$, then we can say that the model and the real data are different (and we'll be wrong at most 5% of the time).

Unfortunately, this is not quite what we want. We want to know that the model and the real system are the *same.* And, unfortunately, we cannot simply say that if we do not find a difference (ie if $p>0.05$), then they are the same. *Absence of evidence is not evidence of absence.* Not finding a statistically significant difference does not mean that they are the same; it just means that we did not find a difference. , it turns out that if we just increase the number of items in each set (ie if we examine more people and run the simulation more times), then eventually *we will always find a difference.* Since the data from the real system and the data from the model are, in fact, from different sources, there will always be a slight difference. If we examine enough data, we can *always get a p-value below 0.05.*

Fortunately, there is a variant of the t-test called an *equivalence test* which is more suited to our situation. This test, often used in medical studies to show that a new treatment is as effective as some other (usually more expensive) treatment, is the reverse of a t-test. Instead of having $p=0.05$ signify a 5% chance of what we observe being the result of two identical groups (and thus allowing us to conclude that the groups are probably different), ap value of 0.05 means that there is a 5% chance of generating the observed data from two data sources *that differ by more than a given amount* (thus allowing us to conclude that the groups are probably similar to within some particular range).

This last statement is worth reviewing. There are two changes from the standard t-test. First, the direction of analysis has changed. A small p-value means that the groups are the same (ie that the model matches the real world situation), rather than the other way around. Second, we have introduced the idea of being 'similar enough'. We are not looking for a model which *exactly* matches the observed data. We are looking for a model which matches *close enough for our current purposes.* This, of course, leads us to a new question: how do we decide what 'close enough' means?

Choosing this 'close enough' range is not trivial, and is highly dependent on the situation. There are currently no useful guidelines for doing so. Certainly, the range should be at least as large as the measurement error inherent in determining the real-world data, and certainly it should not be so large as to indicate that *any* model is 'fits' the situation. In any case, using this technique means that computational modellers will simply have to

report the ranges they have chosen, and an eventual consensus will hopefully be reached (much like the eventual consensus of ap value of less than 0.05 being good enough to conclude that there is a statistically significant difference).

Measuring Multiple Aspects

To complicate matters even further, we also have to remember that we generally care about *more than one value.* For example, if we are modeling single word reading, we might measure the accuracy rate for known regular worlds, known irregular words, and unknown regular words. Indeed, there are numerous possible points of comparison between model and reality. The following aspects have been identified in (Simon & Wallach, 1999) and originally in (Wallach, 1998) as being particularly useful sorts of correspondences:

- *Product:* the final outcome (reading a word, solving a puzzle)
- *Intermediate Steps:* observable aspects of the cognitive process (eye movements, thinking aloud reports)
- *Time:* how long responses take (in terms of a number predicted by the model, not necessarily exactly how long it takes a computer to run the model)
- *Error:* both humans and the model should differ in the same ways from the imaginary 'ideal' behavior (both number and type of error)
- *Context Dependency:* the effects of changing the environment or other interfering factors
- *Learning:* the effects of practice (and forgetting)
- *Lesioning:* effects of damage to the systems

The ideal, of course, is to develop models which match in all of these areas (and more). In the meanwhile, we can collect a ground of measures appropriate to the particular *grain size* we are investigating. For a model to be 'good', it should match the human data on all of these measurements. The equivalence testing we have just discussed only handles a single measurement at a time. Certainly, we can get a p-value for each one, but we want a measurement that will encompass each of these key factors.

Combining p-values is a well-studied and unsolved problem in statistics. There are numerous techniques for doing so – Fisher's method (Fischer, 1958), Stouffer's method, and recent innovations like the Truncated Product Method (Zaykin et al, 2002) – but none are suitable for our situation. These techniques are suitable for situations where we want a combined p-value assuming *all of the null hypotheses are true.* Thus, a combined p value of 0.05 after our multiple equivalence tests would mean something like “if our model differed from the real world on *every one* of the measurements, then there'd be a 5% chance of observing something as extreme (or more so) as we did". This is not what we want: we want to know if *any* of the measurements differ. That is, if a model matches on two measurements, but is wildly off on a third, then we do not want to classify it as a 'good' model.

Unfortunately, there seems to be no clear objective technique for doing this. A simple method like taking the largest of the measured p-values seems to be a safe option. This is

equivalent to saying that a model is 'good' if and only if *all* of its measurements pass the equivalence test. This can lead to over-rejection (after all, if a model has 100 measurements, random chance will cause some of those will fail the equivalence test even if they are equivalent). But, for reasonably small numbers of measurements, it is the recommended option.

When Your Best Isn't Good Enough

Given the complexity of model making, it is certainly possible that none of the initial models under consideration are 'equivalent' (in this sense) to the real situation. This is certainly the case when there are multiple measurements under consideration at the same time. After all, we often want our models to predict more than one aspect of the real situation. Indeed, even just predicting the mean and variability of a single measurement can lead to difficulties. In this sort of situation, are we to simply say that all the models are equally bad? Or is there a technique which can guide us towards better models?

There is no perfect answer to this situation. Due to the afore-mentioned unpredictability of the behavioral effects of changing models, we cannot assume that identifying some models as 'somewhat good but not equivalent' will help us develop better models in the future. However, this sort of gradual improvement from initially poor models has been successful in other domains, so it would be useful to be able to quantify how far away a model is from giving 'good' results. How can this be done?

Again, statisticians come to our rescue. The technique is known as Relative Likelihood Ratios, and involves comparing models in terms of how likely it is that a particular model could have generated the observed real-world data. Remembering that the models are, generally, non-deterministic, we can analyze their output in terms of the distribution of their results. Given this (and certain assumptions about the distributions of those outputs), we can calculate for each model what its probability is of giving us the sort of data we are looking for. Now, for models that are not 'equivalent', this number will be very very small. However, we can *compare* these values among the models that we are looking at. For more information on this technique and it modeling applications, see (Glover & Dixon, 2005).

Researchers should be warned in interpreting the results of this relative measurement. If model A is 1000 times more likely to give the observed result than model B, that does not mean that model A is 1000 times as likely to be the correct model. If they fail the test of equivalence, then neither of them is correct. However, knowing that, in some sense, model A is *closer* to being correct may help guide the development of new models.

Computational Models as Hypotheses Now that we have more clearly defined what a computational model is, and what techniques we can use to find 'good' ones, we need to know what we are going to use them for. Where do computational models fit within the scientific method? Do they introduce a new aspect of scientific methodology, or do they fit nicely into the standard practice of science?

The major claim of this paper is that computational models do fit cleanly into the standard scientific method. In particular, *computational models are hypotheses.* That is, a particular computational model can be seen as a hypothesis that this computer program is predictive of some particular real cognitive behavior. This fairly innocent statement turns out to have some surprising implications and ramifications that are explored in the rest of this paper.

To start this exploration, we need to take a closer look at the classic scientific method and how it is affected by having this sort of hypothesis. We examined previously the typical picture of having some data, generating a hypothesis that is consistent with that data, using that hypothesis to generate a prediction about the world, testing the prediction, which gives us more data, and the process repeats itself. What changes when we say that the hypothesis can be a computer program?

Model Complexity The

first major difference is that a computational model is significantly more complicated than any hypothesis traditionally found in science. This results in two major effects.

The first effect of complexity is that the development of models is highly unconstrained. We are used to simple hypotheses, such as “the rock will follow the trajectory given by this formula” or “Group A and Group B will perform differently on Task A but not on Task B.” Instead, we have a hypothesis saying that “the real-world behavior in this situation will be the same as the output from the computer program specified as follows <insert 10,000 lines of source code here>”. Furthermore, the generation of this hypothesis requires the creation of a full, working computer program, which is not an easy task.

The second effect is that computational models are *predictively opaque.* That is, determining what a given model predicts will occur in a particular situation is a non trivial process. In order to do this prediction, the computer program itself must be run. That is, *knowing what the model is does not tell you what it does.* This idea (sometimes called *emergence)* often comes as a surprise to those unused to complex iterative programs. These computational models generally produce behavior that surprises the person who wrote the program in the first place. The entire fields of Distributed Cognition and Artificial Life are rife with examples of this sort of occurrence, and the unexpected performance of large computer programs is certainly something that any software engineer can attest to. While a computer may do exactly what you tell it to at a low level of description (the code itself), this certainly does not mean it will do what you expect at the higher level (overall behaviour).

What this means for us is that it is significantly more difficult to verify that the behavior of a given model accurately matches the *known* data, even before we start worrying about predicting new data. Indeed, in many current journals, it is considered a publication worthy result to just get a computational model which matches already known data (the issue of what it means to 'match' known data is addressed later; for now we will simply assume that this is possible, and refer to such models as *good* models). While this state

of affairs may be acceptable for now, in order to make real progress, we need to be doing more than 'fitting' a model to a particular set of data. After all, with a given set of data, it is *always* possible to generate a computer program that will produce that result.

Model Proliferation

The second issue that arises when dealing with computational models is a sudden proliferation of potential models. This is clearly a direct result of their complexity: indeed, in any scientific domain where we must deal with complex hypotheses, there are going to be many possible hypotheses. Making any one of a practically infinite number of changes to a given computational model gives a new model which may be better or worse than the original model. Furthermore, there may be numerous completely different approaches to creating models, resulting in hundreds of models which are completely different on an implementation level, and yet may behave extremely similarly. How are we to deal with this situation?

Occam's Razor is, of course, a tool that can be used in this situation. Some of these models may have components which, if removed, result in models with equivalent predictions. Here, the principle of “not multiplying entities beyond necessity” clearly holds. But this tends to be a rare occurrence; more often we have millions of very similar models with slightly different predictions, or dozens very different models with slightly different predictions. We might use the approach of using the 'simplest' model, but defining 'simple' practically can lead to tremendous difficulties.

One way to envisage this embarrassment of riches in the number of models is by taking advantage of the previously mentioned organization of models into families. Within each family, the models differ only by their particular parameter settings. This means that, within a family, the models are each of *exactly the same complexity,* and so can only be distinguished in terms of their predictive abilities. However, each of these model families contains an infinite number of models. How can we possibly find the best one?

Quite simply, we cannot. If we accept that the only reliable method for evaluating a model is by running it, then we must also accept that we must evaluate an infinite number of models if we want to find the best one *even of just those within a family.* It might be thought that we can choose some small difference in parameter settings and say that we will only consider changes of this size, and we might choose some bounds on the ranges of parameter settings, thus leading to an astronomically large, but finite, number of models to evaluate. However, this is not only impractical (as evaluating an astronomical number of models is not going to happen within anyone's research time frame), but not guaranteed to work. In general, one cannot predict how large a change in behavior will result from a given (small) change in the parameter settings. In other words, it is theoretically possible for a highly accurate model to be hidden directly between two highly inaccurate models. It is, of course, possible for a mathematical solution to arise which will provably identify an optimal model, but *we cannot rely on this possibility.*

What, then, can we do to investigate these model families? We cannot try out all the models. We know that models which differ by a small amount in terms of parameters

may differ by a large amount in terms of behaviour. Are we lost, looking for a needle in an infinitely large haystack?

This problem is dealt with by redefining what we are looking for. We should not be looking for the one particular model which best fits the known real-world data. Instead, we look for *areas of the model space which tend to include 'good' models.* That is, instead of looking for one good model, we can find whole clusters of models, all of which match the observed data. Ideally, we may find that, for a particular model family, setting parameter A between 0.5 and 0.7 and parameter B between 1.8 and 400 consistently gives 'good' results. Of course, we cannot prove that *every* model in this range is 'good', for the same reason that we cannot examine every model in the first place.

Indeed, if we use the equivalence testing procedure discussed earlier, then we are guaranteed to *always be making mistakes about whether a model is 'good'.* Remember, the equivalence test gives ap value which indicates the chance of getting the observed result if the model and the real world are different by more than a certain amount. Thus, if we use ap value of less than 0.05 to mean that the model is 'good enough', then up to 5% of the time a bad model can be determined to be good. This, along with the possibility of good models not being identified as good, and the fact that we are evaluating thousands or millions of models at a time, means that we are not generally going to care about 100% accuracy in determining the space of generally 'good' models.

Given this issue, using this approach is not as simple as it seems. Furthermore, there is certainly no guarantee that such an area will exist with the model space. Even if it does, finding may be a very difficult task. It is also likely that there will be multiple such clusters, of varying sizes, and they are certainly not going to be as nicely delineated as in this example. Parameters may have all sorts of complex cross-correlations and interaction effects. Fortunately, there are techniques for approaching this sort of situation, as will be discussed later.

For now, we can simply note that if we can find such areas where we have identified models with 'good' parameter settings (using something like the equivalence testing discussed earlier), then we can generally consider any of the models within that set as functionally equivalents. They are all 'good enough' for our purposes.

If we are willing to say this, then we can also make use of this generalization to affect our original *principles* (in Newton's sense, as discussed earlier). That is, if we find that a particular sort of model tends to be 'good' with parameter settings in a particular range, and if this happens across a number of different domains, then we can *include that parameter range as part of the principles for constructing such models.* This can currently be seen in ACT-R models, where numerous experiments show that the models which best fit the human data have the Latency parameter set to 0.5 and the noise parameter set near 0.3. However, by indicating a range of 'good enough' settings rather than one 'optimal' setting, we get a better sense of how *stable* the parameter settings are.

For a good example of this sort of range-based approach, one can look at the standard results of the study of artificial neural networks which learn through back-propagation of error (Rumelhart, Hinton, and Williams, 1986). Here, researchers have generally found two regions of the parameter space which tend to lead to 'good' models (in this case, 'good' is defined as models which learn the input-output mappings that they are taught in a reasonable amount of time). It is unimportant for our discussion what those regions are, but they tend to work well for a variety of situations. Given this result, we can certainly hope that *any* new system which uses such a neural network should be reasonably 'good' with these settings. This is not guaranteed, by any means, but if the neural network is to be a useful component in modeling, *then we must be able to find such a regularity.* If we cannot find such regularities in our models, then it we must go in search of completely new sorts of models.

Behaviorally Identical Models The previous section describes how we should interpret multiple models *within the same family* which give the same (or similar) predictions. But what do we do if we find two (or more) models *of completely different types* which give 'good' predictions?

This is clearly theoretically possible; given any computer program, it is always possible to re-write it into a new program which gives *exactly* the same outputs for the same inputs, and yet has entirely different source code (Moore, 1956). It is not even theoretically possible to consistently detect this sort of convergence; we cannot, in general, know that two programs will give identical inputs and outputs without trying all of the infinite number of inputs, which is certainly not a practical methodology. So what should we do when we find two models wildly different in implementation, and yet identical in behaviour?

This is, of course, the same problem as in all of science when there are two competing theories, and they both give the same predictions. Some may argue that the 'simple' model (in some, vaguely defined, sense) is the one that should be used, although this seems to be inconsistent with the actual practice of science. What actually tends to happen is that both models stay in use, with one usually significantly dominating in presence over the other. The deciding factor may then be *which model is more conducive to further exploration of the phenomena.*

There is an interesting analogue to this situation occurring within the physical sciences is from the field of quantum physics. Both the Copenhagen Interpretation and the Bohm Interpretation give *exactly* the same predictions (for all situations where the Copenhagen Interpretation is unambiguous). However, the Copenhagen Interpretation (that 'observing' quantum events 'collapses the waveforms') is the dominant one, both among quantum physicists and within popular media (for example, the Schrodinger's Cat paradox, that if you kill a cat based on the outcome of a quantum event, you end up having to treat the cat as both alive and dead at the same time until you actually observe it, is only true in the Copenhagen interpretation). In contrast, the Bohm Interpretation (that there are a large number of hidden, non-local variables that we can see the results of

be not actually measure) leads to no such situations, but has not received wide-spread support.

There seem to be a number of *sociological* reasons for this difference, including meme spreading or simply the attraction of strange ideas. However, there is also a very good *scientific* reason why one might pick one model over another, in the absence of any distinguishing data. One model may be better for *thinking about the problem.* This is, it may be easier to work with, or tend to more readily lead to interesting or useful predictions.

With this in mind, finding multiple highly different models which give the same predictions is not a problem for computational modeling. We keep them all, and can continue to use them at our convenience. If future information allows us to decide between them, then we can do that; in the meanwhile, there should be no pressure to determine which is the *real* model. Indeed, most quantum physicists regard worrying about the multiple interpretations to be a waste of time.

Truth and Reality So far, this paper has avoided one of the aspects of scientific knowledge that many consider fundamental: the search for Truth. The theories and information gained through the process of science is supposed to explain to us how the universe *really* works. We want the components of our scientific discoveries to be real entities, not just convenient fictions. Is this possible? What does it mean to do this?

Dennett's Real Patterns

In *Real Patterns* (Dennett, 1991), and in other works, Dennett argues that theoretical constructs such as *beliefs* are exactly as 'real' as such physical science constructs as *centers of gravity.* The same argument can be applied to our theoretical constructs: the components of the computational models.

Dennett's approach starts with noting that some philosophers (such as Dretske), believe that centers of gravity are 'real', while others do not. He then goes on to raise the question of the reality of strange abstract objects as “Dennett's lost sock center: the point defined as the center of the smallest sphere that can be inscribed around all the socks I have ever lost in my life.” He says:

> These abstract objects have the same metaphysical status as centers of gravity. Is Dretske a realist about them all? Should we be? I don't intend to pursue this question, for I suspect that Dretske is — and we should be — more interested in the scientific path to realism: centers of gravity are real because they are (somehow) *good* abstract objects. They deserve to be taken seriously, learned about, used. If we go so far as to distinguish them as *real* (contrasting them, perhaps, with those abstract objects that are *bogus),* that is because we think they serve in perspicuous representations of real forces, “natural” properties, and the like. (Dennett, 1991, 97)

He then goes on to argue that 'good' abstract objects (or *patterns)* are those that allow us to make useful predictions about future events. For him, this is motivated by the question of treating *beliefs* as real entities. His conclusion is that when we predict someone's behavior on the basis of a collection of beliefs and desires (as in the Intentional Stance), those beliefs and desires are real entities (or, at least, if we decide not to call them real entities, we should be consistent and refuse to call *anything* 'real').

The same argument can certainly be applied to computational models. Indeed, Dennett's focus on prediction exactly matches the requirements we have for 'good' models. He also highlights the fact that these 'patterns' *do not have to be perfectly predictive.*
Indeed, a vital component of his argument comes from the idea that a given situation can be described by *multiple different, mutually exclusive patterns.*

In other words, we should expect to find that a given cognitive process can be usefully modeled in different ways by different models. And *all of those computational models are equally 'real' (or* 'useful' or 'good', depending on your preferred terminology). This is true even if the models give different quantitative predictions for the same situation.
The key is to remember that our criteria is not that models give exact quantitative predictions in a particular situation: it is that they do, overall, in many different situations, give us good results.

As research progresses, we do expect to find models which are more and more accurate over wider and wider domains; that is, after all, how science progresses. But we should not also be worried about whether our models are 'true'. If the best approach to the question of what things are 'real' is the same as determining if those things are 'predictively useful', then we do not need to do more than producing better and better computational models. As Dennett concludes, “is the view I am defending here a sort of instrumentalism or a sort of realism? I think the view itself is clearer than either of the labels” (Dennett, 1991, 120).

Model Structures vs. 'Real' structure

This view on 'patterns' in the world around us allows us to immediately call into question the idea that the structure of a model should be the same as the structure of the thing being modeled. Whether this means the causal structure or the physical structure is irrelevant: the plurality of Dennett's 'patterns' means that *there is no one 'real' structure of the thing being modeled.*

This is unfortunate, since it would be nice to rely on the simplistic idea that all of the components of our models *must have* a corresponding component in the real system. It would be nice to be able to say that if we have two models which are equally predictive but have very different internal structure, then we should just look at the components in each model and find out which ones do not 'exist' in the real world. After all, if we follow Dennett in saying that 'real' patterns are things that can be part of our predictive models, then *any component in any computational model that is usefully predictive is a pattern, and so therefore is 'real'.*

Are we then to abandon the idea that some models are 'just' predictive, without being explanations? What should we make of techniques such as 'functional measurement' (Anderson, 1976), where performance is modeled using a simple mathematical equation relating various postulated internal factors to a given output? This is exactly the sort of 'model' that does not seem to be satisfyingly 'explanatory'; it just gives equations saying “if a subject places more importance on A than B, then they will perform better on this task.” Surely we do not want to see this sort of model as having the same 'truth' as a more complex computational model that actually describes all of the internal processes involved, even if they are exactly as predictive.

The solution to this problem is to note that they are not exactly as predictive, and they never will be. The functional measurement model simply cannot be expanded to other domains: there is no way to start using it as it is for timing measurements, or for learning, or for anything other than the particular domain it was developed for. It is simply not a rich enough model. Computational models, however, give incredible freedom to the researcher to find new sorts of potential predictions and domains of applicability. It is this aspect of being able to expand the set of data being accurately predicted that we want in any scientific model, and it is on that criteria *alone* that we can say that a given model is 'better' than another. We simply cannot do it on our intuitions that a particular sort of model is a 'more satisfying' explanation than another sort of model.

Expanding the Data Set We believe that the more a model predicts (to a useful degree of accuracy), the more we should treat it as a real explanation of the cognitive phenomena in question. Dennett's afore-mentioned philosophical argument certainly leads us in this direction, but it would also be good to have confirmation from the practice of computational modeling that this principle is followed.

What we need to find are examples of a computational model that was developed for one domain that was then expanded into a new domain. A successful expansion of this sort should be an indication that the model is genuinely predictively useful, and thus we should have more confidence in its veracity. Conveniently, this is precisely what is happening with one of the most successful computational cognitive models: ACT-R.

We will discuss ACT-R in more detail at the end of this paper; for now we can simply note that it has had a long and successful history in its ability to match and predict human performance on a variety of high-level human reasoning tasks. This matching has been primarily done on reaction time and error rates (two of the key correspondences mentioned earlier). Recently, however, in (Anderson et al, 2004), they have shown that the exact same models can also match the observed fMRI data on blood-flow to various regions of the brain during these cognitive tasks. This striking result has caught the attention of a much wider audience, and is, anecdotally at least, being interpreted as a sign that ACT-R models can be seen as saying something about how brains 'really' work.
It is this idea of broadening the scope of the data set that is key in all sciences for a model to be accepted.

Indeed, we might argue that “being predictive across many domains” is exactly what is meant by an *explanation.* It is exactly this sense of universality that leads us to look at such physical 'laws' as Newton's Laws of Motion, even though, technically speaking, those 'laws' are, as Griere points out, “neither universal nor necessary – they are not even true." (Giere, 1999, 90). Instead, they can perhaps better be seen as a set of principles for building predictive mathematical models of physical phenomena. But, due to the wide spread acceptance of their starting power for being useful in a wide variety of domains that people can treat them as 'Laws'. It is exactly this sort of process that we can envision for computational modeling.

The Scientific Method for Computational Modeling We thus arrive at the following structure for scientific inquiry using computational modeling. Here, we give an idealized version of the process; the second half of this paper deals with the practical considerations which must be taken into account by non-ideal cognitive scientists.

First, we identify a set of previously known real-world behavioral data. This forms the initial knowledge base to allow us to generate potential explanatory models. This should include a variety of types of information to be compared: different sorts of measurements, different domains, and different ways of looking at the raw behavioral data.

Second, we generated a large number of different computational models. These must be created from a variety of different approaches. Ideally, there will be models built utilizing a wide range of competing cognitive architectures, from neural networks to expert systems. Each of these models also gives rise to a vast array of 'similar' models that differ only by making small changes to the basic parameters within the system.

Third, we run each model. The result of this is a set of simulated behavioral data for each model (and each parameter variation of each model). This involves very little intellectual effort on the part of the researcher; the only difficulty here is the organization required to run many simulations on many computers and collect the data.

Fourth, we compare the model predictions to the initially known behavioral data. This results in the identification of the models which do, in fact, behave as they should. Importantly, it is unlikely that there will be only one model which matches the real behavior. In general, this stage will result in the identification of a number of different types of model that match, each of which matches over a (possibly complex) range of parameter settings. It is, of course, possible that no models will match, in which case we need to create new ones.

Fifth, we examine the competing successful models to identify other possible measurable domains where they will give differing predictions. This is a highly non-trivial task, given the complex and opaque nature of model behavior. The goal is to find ways to reduce the set of matching models. While doing this search, we also want to watch for sets of models which give matching predictions in all domains.

Sixth, we turn again to the real world to gather the data we need to distinguish between models. This new data can be considered to be an addition to the original data, and the whole process continues. Again, if we find that no models match, then we need to create new models. If too many match, then we need to identify new data that could make the distinction.

This process is exactly the classic ascending spiral view of scientific progress. Importantly, the use of computational models *is not a fundamental change to how science is done.* We are doing the exact same steps, but with a little more attention paid to various aspects, to reflect the fact that we are dealing with more complex and more precise hypotheses than ever before.

Scientific Revolutions

There are clear parallels in the above description of the procedure for computational modeling and Kuhn's "Structure of Scientific Revolutions" (1962). We must acknowledge that cognitive science is in a pre-paradigm stage, or, perhaps equivalently, in a crisis or multi-paradigm stage (Giere, 1999, 35). This is because there is no dominant 'paradigm' to guide the solution to new problems. We do not have the 'normal science' situation where "solutions to new problems are developed by modeling them on the exemplary solutions that underlie the general approach" (ibid).

Whether or not we will eventually find a dominant set of models for the majority of questions in cognitive science is a question best left to future historians. Instead, we need to embrace our current situation of having many possible modeling frameworks, and work from there. Vitally, this means we *need to use multiple computational models.* We do not need to have a situation where separate research groups specialize in separate types of computational modeling. Instead, those cognitive researchers interested in modeling as a tool need to compare many sorts of models. At this stage, it is likely that some modeling approaches will work in certain situations, and not in others, and this 'suitability space' is a large part of what current research will map out.

Evolution of Science

It should also be noted that this process of having multiple potential models and deciding among them based on their performance on matching a particular set of data and also on the effects of applying them to new domains is one familiar to any scientist. In particular, if we consider the transition from Ptolemy's epicycle model to Copernicus' heliocentric model, we see an interesting phenomenon. Importantly, Copernicus' model was *not* better at matching the observed motions of the planets. Indeed, the standard epicycle model (with its tens of nested circles, carefully calibrated to match the observed motion of a particular planet) was *more accurate* at predicting future motions of that planet.
However, they were *both* a 'good' match for many purposes.

This is exactly the approach we should take with our models. Even if model A is technically a closer fit to a given situation than model B, *this should not mean that we abandon model* B. Instead, we decide on what is a 'good enough' match before running our experiments, based on the practical uses of our predictions, and the observed

underlying variability. If both models fit well enough, then we need to ignore the fact that one of them will, perhaps just by chance, be a mathematically 'better' fit. We can then turn to new sources of data to separate the two.

It is this line of reasoning that led to the heliocentric planetary model being seen as a more 'accurate' model of planetary motion, *even though its predictions were actually less accurate.* The heliocentric model gives a framework that can be used for new planetary bodies, and is thus more usable in a wider variety of situations. Our approach to computational modeling should follow the same route.

Of course, it should be remembered that Ptolemy's model was a necessary and useful model when it was developed. We always want to make use of the best *known* model, even though we are quite sure that it is not the best *possible* model. Even though the structure of Ptolemy's approach has turned out to be not useful in terms of modern models, it was a necessary and important step in the evolution of planetary models. For this reason, we should not be worried about ensuring that the structure of our models is in some sense 'correct' or 'plausible'; these features will come for free as we continue to develop models which are suitable across more domains.

Separating Hypothesis and Prediction It is worth pointing out one small change encountered by the approach advocated here. In particular, by identifying the computational model with the hypothesis, and by accepting the opaqueness of the model, we are in effect separating the hypothesis and the prediction. Often in science, the hypothesis is exactly the same as the prediction, and this allows us to conclude that if the prediction is true, then the hypothesis is true, and if the prediction is false, then the hypothesis is false.

However, we have a situation where the hypothesis (H) implies a prediction (P), and we then check the truth of our prediction. So, we have H->P, and if we determine that P is false, then we can clearly also conclude that H is false. However, if P is true (ie if the model is predictive), then we have an interesting situation.

We clearly cannot simply conclude that H is true (ie that the model is 'true' or 'correct'); this has been mentioned earlier. This is because H->P does not mean that !H->!P. Finding one predictive model does not mean that other models are going to be less predictive. Certainly, it is a good first step, but it is quite possible that almost any other reasonable model could give a similar result.

This means that we should not follow the typical approach of testing a single hypothesis at a time. Instead, we use the multiple models approach and have hundreds or thousands of models being tested. We have H1->P1, H2->P2, H3->P3, and so on. The ideal situation is that we can find one model (or one type of model) which gives accurate predictions, and we can at least show that none of the other models we tried worked either.

Most importantly, we cannot take the increasing common approach of deciding upon the one hypothesis that we think is 'right' one, and running our experiment to confirm or

deny this hypothesis. The experiment is meant to decide between competing hypotheses, and is not a situation where we should be 'hoping' one way or another, and thus introducing potential biases. This is particularly a problem in computational modeling, where traditionally researchers have had favored modeling architectures. While this approach may be appropriate when initially developing new sorts of models, we must be more willing to make use of models from more than one modeling camp.

This also means that we must perform our modeling in such a way that it is *easy for other researchers to do the exact same test using new models.* Since we clearly cannot test against all possible models, it is likely that future researchers will want to continue to explore a given situation with new models, or make slight changes to the experimental situation to attempt to tease apart the results of currently equivalent models. With computational modeling, this process can be made to be extremely simple, as the source code to our systems *exactly specifies what it is we have done.* We can thus easily allow for complete replicability in our research. This is exactly in line with Lane & Gobet (2003)'s call for the publication of automated systems for re-running key experiments with every publish computational model.

Multiple-Model Approaches A key

aspect of the approach to computational modeling advocated in this paper is that we need to try multiple sorts of models at the same time. This is not a common phenomenon in cognitive science as it is currently practiced. This is likely due to the traditional difficulty faced in creating computational models in the first place: there is a significant amount of effort put into the creation and programming and debugging of a single model, making it difficult to justify creating five or six different families of model for a particular phenomenon. Furthermore, it is just not a common practice, so there is little incentive to do so.

There are two recent factors that may push towards changing this state of affairs. First, computing tools have become significantly easier to use in the past decade. Higher-level programming languages and extensive well-designed code libraries allow more complex programs to be written more quickly and with fewer specialized skills. Secondly, there has been an incredible growth in the number of computational models of aspects of cognition.

Together, these factors change the situation for a cognitive scientist interested in doing computational modeling. Instead of designing their models from scratch, they can more easily build up a model for their particular situation using pre-existing software. This greatly reduces the effort required to make use of a model and makes the use of multiple models more possible.

The Practice of Computational Modeling The preceeding

sections of this paper lay out the general theoretical argument for how computational modeling should be performed. However, putting that theory into practice leads to a whole other set of concerns. While this sort of modeling does fit well

into the general theory of science, these practical details can make the process of computational modeling into something unfamiliar to most researchers.

The following sections give tips and techniques for practical computational modeling.

Developing Models The first stage involves generating a number of models for the phenomenon of interest. As argued earlier, showing that one particular model 'matches' the real-world data is not sufficient. We gain a much more powerful conclusion if we can show that one particular model does fit well, while a large cross-section of other plausible models do not give good results. This then means that we need to create all of these models to be tested. How can this be done in a manner that is suitable for the time and skill constraints that most cognitive researchers will be under?

First, it is important to start with models that are as simple as possible. This should include models that are too simple to be at all accurate. For example, in (Stewart, West, & Coplan, 2004), we looked at models of children forming friendships. The simplest of these was simply a statement that children randomly choose people to be their friends, based on no information at all, and it keeps changing from moment to moment. Surprisingly, this model was sufficent to accurately model the results of a standard measurement of the number of 'Popular', 'Rejected', 'Neglected', and 'Controversial' children in a given group. This model was then able to serve as a baseline for comparison for other models, which were able to deal with other sorts of data as well.

Second, we need to make extensive use of existing types of computational models. There currently exists a vast array of different sorts of models suitable for different situations. For any given cognitive phenomena, we can build up possible models using these existing systems as building blocks. For example, in (West, Stewart, Lebiere & Chandrasekharan, 2005), we created models of Rock-Paper-Scissors players using ACT R, Q-Learning, Recursive Neural Networks, and Assosiative Neural Networks. By using and adapting these existing sorts of models, we are in effect trying to use various pre-existing 'principles', in the sense described above. Since a major aspect of our inquiry into these cognitive 'principle' is to learn how to use them to generate predictive models, we clearly must try to make use of them in this way. We can thus find that some types of models tend to work in certain situations, while others are better for other phenomena, which is a major aspect (and perhaps the major goal) of cognitive science.

We have to make sure that not only do we use a large number of different types of models, but we also need to examine a large number of different parameter settings for these types of models. Remembering our terminology that a 'model' is a computer program *with all its parameters fixed,* we need to make sure that we explore the behavior of the range of models with different parameter values. This means that we cannot simply decide that we will only look at a given set of models constrained by setting, say, the 'learning rate' to 0.1. Even if there is strong evidence from previous research saying that this type of model tends to work well near that range, we must at least examine the behavior of the system outside that space. We do not need to

exhaustively search the parameter space, but we certainly cannot arbitrarily ignore large swaths of models.

Source Code

This requirement of using a large number of fundamentally different models seems to put an inordinate load on the researcher. Surely we are not expecting cognitive scientists to spend months implementing various algorithms, as this is clearly not an efficient use of time. Instead, we need to be able to quickly build models suitable for the current situation using pre-existing software libraries. We do not need to reimplement everything from first principles. Just as we do not need to write our own programming language, we do not need to write our own computational modeling libraries.

Traditionally, this has been a difficult task in computational modeling. Even in such a computing-oriented field as Artificial Life, it has been observed that

> Sharing work has been so difficult that researchers tend to build their own animat minds and worlds from scratch, often duplicating work that has been done elsewhere. There have been a number of attempts to re-use animat or agent minds and worlds, but the model of re-use often requires installation, or even a particular programming language. … Often, the only person who ever does experiments with an animat or agent is its author. In this field it has become acceptable not to have direct access to many of the major systems under discussion. (Humphrys & O'Leary, 2002)

This is clearly unacceptable. Humphrys and O'Leary argue that the solution is that people who develop modeling architectures should set their software up to run on a publicly accessible computer, so that other researchers can log in remotely and use the system. This approach is somewhat cumbersome and has yet to catch on, but it is certainly possible and straightforward for any algorithm developer to make their source code available to other researchers. Doing this has a number of scientific advantages.

First, computational models of cognitive behavior tend to be rather complicated. It is certainly not possible to adequately describe them within the constraints of a 6-page paper. Indeed, the only full and complete description of the model *is the source code itself.* Not making this available is equivalent to not giving an adequate description of one's experimental methodology or stimuli, and certainly hampers scientific progress.

Secondly, making this code available allows us to achieve an oft-neglected ideal of science: perfect replication. By making the complete software available, anyone, anywhere *can exactly replicate* all of the data produced in a computational modeling experiment. Ideally, this can include all of the statistical data analysis and even graph generation. Lane and Gobet (2003) discuss the generation of automated test suites which can replicate the core important findings for a model, and which can be automatically run when one is making modifications to a model so as to confirm that the new model still has the important features of the old one. Furthermore, these tests can be run to compare multiple models to each other *using exactly the same tests,* with no extra effort on the part

of the researchers. This is especially important, given the fact that there are more architectures out there than there are current comparisons between architectures (Guilot and Meyer, 2000).

Finally, having the source code available (for both the model and the data analysis) means that errors can be identified. We certainly should not expect that computational modellers are somehow immune to the sort of software bugs which plague all programmers. Certainly, the reuse of code can mean that some of the key architectures can be rigorously tested, and this will help alleviate the problem somewhat, but there is still plenty of room for simple errors that can sneak through. This is not a problem unique to computational modeling; indeed, all sciences have the potential for undetected errors in analysis or methodology. But, the availability of source code can make computational modeling into a domain where *these sorts of errors can be found.* This unique ability needs to be highlighted, and we need to recognize that these sorts of errors will happen. Finding them and correcting them should be a natural part of the field.

From personal experience in the field, and from anecdotal evidence from other researchers, it is clear that we are often hesitant to make source code to our models available due to them being 'messy' or 'unreadable'. Some would argue that this means we should spend more time getting our source code in order before publishing (Lane and Gobet, 2003). Instead, in may be better to just make the code available anyway. After all, given the experimental nature of model development, it is quite likely that the first version of a new modeling system will be a horrid monstrosity of badly-written code.
Indeed, it may be that it is better for *someone else* to re-implement the model into a clean and more widely useful format (possibly even in a different language). Having the original code available, however ugly it may be, allows for a reimplementation that can be directly tested against the original. It can even be convenient to have multiple implementations of the same modeling system, each suitable for different sorts of situations (see Stewart & West, 2005 for an example of this).

Combinatorial Explosion Once

we have decided upon a number of differing modeling architectures to use, we now have the problem of running and testing them. The first major difficulty here is one of combinatorial explosion. As discussed previously, every combination of parameter settings *gives a different model that must be tested independently.* In general, a particular type of model will have multiple parameters, leading to an *infinite number* of potential models.

Now, we are clearly not looking for the *best* model among that infinite set: that would require either a lucky bit of mathematical trickery, or an infinite amount of computing power, neither of which we can rely on. Instead, we are looking for areas of the parameter space which tend to produce models that match well to the real data. If we are dealing with modeling architectures which are well-known, and have suggested parameter settings, then we can certainly make use of this to narrow our search, but this is still going to leave us with many possibilities.

The first thing we may do is to divide each parameter into discrete chunks. We cannot study all the models with parameter values between 0 and 1 (as there are an infinite number), be it is possible to examine the models with the values of 0, 0.25, 0.5, 0.75, and 1. Doing this, of course is dangerous: there might be a set of excellent models with parameter values between 0.31 and 0.32, but we will completely miss them. Indeed, there might also be parameters which are better to sample exponentially (0.001, 0.01, 0.1, and 1). How can we even know what the upper and lower bounds should be?

These questions do not have answers we can know a priori. This requires us to do two things: to make a decision based on our best experience, and to fully document that decision when presenting our results. Fortunately, if we follow the advise above and publish our source code as well, this allows other researchers to explore the gaps that we have definitely left behind.

However, even if we make this sort of decision, we are still likely to be left with thousands or millions of models to investigate. Just five parameters with five settings each gives us 3,125 models, and that just includes the models of this one type. Furthermore, each model will need to be run more than once, in order to deal with the variability of its behavior. Is this possible with the computing resources that we generally have available? Will we all need large supercomputing clusters in order to do this sort of research?

Fortunately for us, this computing problem has a feature which allows us to get away with not having an expensive set of hardware. We may need to run these models thousands of times, but we can take advantage of the parallel nature of the situation. Unlike computing problems such as factoring large matrices, which requires a large degree of communication between any of the various parts of the problem, each of the thousands of different models *is completely independent of the others.* This means that rather than having to have one powerful computer, we can have a large number of rather normal computers, and get them all to work on the problem in parallel.

This sort of software goes a long way towards providing sufficient computing power for modeling work. However, there is another, more experimental approach. The same software for creating the parallel computing cluster can also be set up to *intelligently direct the search for useful models.* That is, instead of simply trying each of the desired parameter settings in turn, it can use a *Genetic Algorithm* (or an *Evolutionary Strategy)* to ignore areas of the parameter space that seem to be producing poor results, and focus on areas with more promising results.

This approach is rather controversial. The standard worry with Genetic Algorithms is that they may find one set of 'good' results, but may end up completely missing another set of models which may be even better. However, recent observations involving Neutral Network theory indicate that the more complex a problem is, the *less* likely we are to encounter this issue (Huynen et al, 1996). At the very least, Genetic Algorithms can be used to quickly identify if there are 'interesting' areas of the parameter space which we can then rigorously explore by the previous exhaustive search method. This helps to

alleviate the problem of having to decide on a particular sampling of the parameter values.

Evaluating Models

Given the above considerations, it is possible to run the thousands of models it is necessary to evaluate. Since there are so many models, we would also like our software to automatically do the sort of analysis we need. As argued previously, the major measure for model 'fit' is via equivalence testing. Unfortunately, this is a relatively uncommon form of statistics, which can make it difficult to perform. We also want to be able to determine Relative Likelihood ratios, another rare type of statistics.

Once this is all accomplished, we must still be careful in expressing the results of our studies. Importantly, we are not looking for a single 'best' model. Much current work involves a lot of effort finding the parameter values which pick out the one model which 'best' matches the real-world data. This is not the approach we should be taking, as it can simply be a result of over-fitting. Instead, we can find that, for a given set of data, some models 'match' while many others do not match. Using the equivalence testing, we can conclude that a particular model (or collection of models) fits, while all the other models *that we tried* do not fit. Or, if none of the models fit, we can turn to relative likelihood testing to identify those models that are a 'closer' match than the others (with the caveat that being a closer match does not necessarily mean that the model itself is somehow ' similar' structurally to a model that would match). Importantly, these are all comparative statements, and we are only comparing to models within the set of the ones we have identified. This further highlights the point that we must make it as easy as possible for other researchers (or for ourselves) to continue the exact same sorts of comparisons with new models, and even to expand the set of comparison data. This is required for this sort of modeling work to be effective at expanding our knowledge.

ACT-R: A Case Study This

document has described a wide variety of suggestions and techniques for integrating computational modeling into cognitive science. To summarize this and to give it a concrete example, it is useful to examine current research in the light of these ideas.

ACT-R (Anderson & Lebiere, 1998) is one of, if not the, most successful computational modeling frameworks currently in existence. In our terminology, it consists of a set of principles which allow for the creation of models of high-level human thought. It consists of a set of tools for modeling declarative memory, procedural memory, and basic visual/ motor abilities. This framework has been used for a wide variety of tasks, from mathematical reasoning to using computer user interfaces, and the results tend to accurately match and predict human performance.

Principles

ACT-R research has been very explicit that it is not a single particular model, but rather a modeling architecture, which allows for the creation of predictive models. This is exactly in line with the idea of looking for *principles* which guide the creation of models.

Parameters

To make models for different tasks and situations, one specifies a collection of 'productions' (aka rules) and a pre-existing set of information in declarative memory. This can be seen as building different types of ACT-R models. Once these are decided upon, there are still a collection of *parameters* that can be adjusted. Changing these parameters produces different models with different behaviour.

Much effort has been put in ACT-R to keep the number of parameters low, which is an oft-neglected important issue for models. There has also been some success in identifying particular suggested values for at least two of these parameters. The default action time (almost always set to 50 msec) regulates the speed of the model, giving highly accurate predictions for how long humans will take to perform various tasks. The activation noise also has a suggested value (anywhere from 0.2 to 0.5), but recent results have shown that for many interactive models, behavior can change significantly depending on this parameter. This effect is an area of ongoing research in the field.

Unfortunately, the story is less promising for the other ACT-R parameters. We generally find that the 'best' models for different tasks have wildly differing latency and goal parameters. To deal with this problem, we need more information on how the model performance differs across these parameter settings (something which currently tends not to be included in published results).

Evaluation

When data from ACT-R models are compared to real-world results, the current standard practice is to use statistical correlation. This gives a single r-value, indicating how strongly associated the two sets of data are. Unfortunately, this is not a particularly useful measure on its own. Certainly, two models can be compared in this manner (although relative ratios would be preferable), but it does not give us any demonstration that the model's behavior is similar to that of the human subjects. For that sort of result, we need to use a technique like equivalence testing.

Furthermore, there is a distinct lack of studies involving both ACT-R models and *other sorts of models.* Most likely, this is due to the other suitable architectures being less widely understood, and more difficult to make use of. This then means *we need to further develop the alternative modeling systems.* Systems such as Clarion (Sun, 1997) can certainly be used for many of the high-level tasks for which cognitive scientists tend to use ACT-R. Instead, we currently have a situation where many cognitive modellers focus entirely on ACT-R models, and thus never do the cross-model comparisons which must be done. This total investment into one sort of modeling is something that is understandable to a certain degree, as ACT-R has a difficult learning curve. Fortunately, projects such as Python ACT-R (Stewart & West, 2005) are striving to reduce this problem, and we should thus see more of these broad comparisons in the future.

Comparison

One of the most important features of ACT-R models is that the interface system which allows the model to interact with some environment (used primarily for studies involving

looking at a computer screen, manipulating the mouse, and otherwise interacting with different user interfaces) *also allows human subjects to use the exact same interface.* This is clearly the idea approach for ensuring that the models and the humans are performing comparable tasks.

Broadening the Scope For

much of the initial stages of ACT-R development, the key data being predicted were timing measurements and error rates. This guided the development of ACT-R as a whole. Recently, however, there has been a perfect example of predicting new sorts of data using exactly the same model. In (Anderson et al, 2004), the traditional ACT-R system is used to predict blood-flow measurements in fMRI data. That is, by identifying which parts of the model are most active at what times, and from previous data relating various parts of the model to various neural correlates, matches can be made to the spaciotemporal data from subjects in fMRI machines performing the same tasks.

Code Publishing

There is currently an online repository for ACT-R models associated with publications (at <http://act-r.psy.cmu.edu/models/>). This is certainly an important step, but has not yet become standard practice.

References

Anderson, JR, Bothell, D., Byrne, MD, Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. Psychological Review 111, (4). 1036-1060.

Anderson, JR & Lebiere, C. (1998). The atomic components of thought. Mahwah, NJ: Erlbaum.

Anderson, NH (1976). How functional measurement can yield validated interval scales of mental quantities. Journal of Applied Psychology, 61:677-692.

Axelrod, R., The evolution of strategies in the iterated prisoner's dilemma, in Davis, L. (ed.), Genetic algorithms and simulated annealing, Research notes in AI, Pitman/ Morgan Kaufmann, 1987, 32-41

Cole, M., & Means, B. (1981). Comparative studies of how people think: An introduction. Cambridge, MA: Harvard University Press.

Cooper, R., Fox, J., Farringdon, J., & Shallice, T. (1996). "A systematic methodology for cognitive modeling" Artificial Intelligence, 85, 3-44.

Dennett, D. (1991). "Real Patterns," Journal of Philosophy 88: 27-51.

Fisher, RA (1958). Statistical Methods for Research Workers, 13th Edition. Hafner Publishing.

Fodor, J. (2005). “What We Still Don't Know about Cognition”

Giere, R. (1988). Explaining Science: A Cognitive Approach. Chicago: University of Chicago Press

Giere, R. (1999). Science without Laws. Chicago: University of Chicago Press.

Glover, S. & Dixon, P. (in press). Likelihood ratios: A simple, intuitive statistic for empirical psychologists. Psychonomic Bulletin and Review.

Guillot, A. and Meyer, JA (2000). From SAB94 to SAB2000: What's New, Animat ? In From Animals to Animats 6. Proceedings of the Sixth International Conference on Simulation of Adaptive Behavior. The MIT Press.

Harnad, S. (1990) The Symbol Grounding Problem. Physica D 42: 335-346.

Humphrys, Mark and O'Leary, Ciarán (2002), Constructing complex minds through multiple authors, Proc. 7th Int. conf. on Simulation of Adaptive Behavior (SAB-02)

Huynen, MA, Stadler, PF, and Fontana, W (1996). Smoothness within ruggedness: The role of neutrality in adaptation. Proc. christmas. Acad. science. USA, 93:397-401.

Klahr, D. (1976). Steps toward the simulation of intellectual development. In LB Resnick (Ed.), The nature of intelligence. Hillsdale, NJ: Lawrence Erlbaum Assoc.

Kuhn, TS (1962), The Structure of Scientific Revolutions, Chicago, University of Chicago Press.

Lane, PCR, & Gobet, F. Developing reproducible and comprehensive computational models, Artificial Intelligence, 144:251-263, 2003.

Lombrozo, T. (2005). Why Adaptionist Explanations are So Seductive. 27th Annual Conference of the Cognitive Science Society.

Moore, EF (1956). Gedanken-experiments on machines. In Automata Studies, ed. CE Shannon, J. McCarthy. Princeton, NJ: Princeton

Putnam, H. (1961). "Brains and Behavior"

Rescorla, RA, & Wagner, AR (1972). A theory of Pavlovian conditioning: Variations in the effectiveness of reinforcement and nonreinforcement. In AH Black & WF

Prokasy (Eds.), Classical conditioning II: Current theory and research (pp. 64-69). New York: Appleton-Century-Crofts.

Reynolds, C. 1987. Flocks, herds, and schools: A distributed behavioral model. ACM Computer Graphics, SIGGRAPH'87, 21(4):25-34.

Rumelhart, DE, Hinton, GE, and Williams, RJ (1986) Learning representations by back-propagating errors. Nature, 323, 533--536.

Searle, John. 1980a. "Minds, Brains, and Programs." *Behavioral and Brain Sciences* 3, 417-424.

Sibley, DE & Kello, CT (2004). Computational explorations of double dissociations: Modes of processing instead of components of processing. Cognitive Systems Research, 6, 61-69.

Simon HA & Wallach, C. (1999). Editorial: Cognitive modeling in perspective. Kognitionswissenschaft 8(1): 1-4

Stewart, TC (2000). Learning in Artificial Life: Conditioning, Concept Formation, and Sensorimotor Loops. Masters of Philosophy Thesis, University of Sussex.

Stewart, TC and West, RL (2005) Python ACT-R: A New Implementation and a New Syntax. 12th Annual ACT-R Workshop.

Stewart, TC, West, R., and Coplan, R. (2004) A Dynamic, Multi-Agent Model of Peer Group Formation. Sixth International Conference on Cognitive Modeling.

Sun, R. (1997). An agent architecture for on-line learning of procedural and declarative knowledge. Proceedings of ICONIP'97, pp. 766-769, Springer-Verlag.

Verschure et al. "Distributed adaptive control: the self-organization of structured behavior," Robotics and Autonomous Systems 9: 181-196. 1992.

Wallach, D. (1998). Complexe Regelungsprozesse. Eine kognitionswissenschaftliche Analysis (Complex control processes. A cognitive science analysis). Wiesbaden: Deutscher Universitäts-Verlag.

West, R., Stewart, TC, Lebiere, C., and Chandrasekharan, S. (2005) Stochastic Resonance in Human Cognition: ACT-R Versus Game Theory, Associative Neural Networks, Recursive Neural Networks, Q-Learning, and Humans. 27th Annual Meeting of the Cognitive Science Society.

Woodward, J. (2003). "Scientific Explanation", The Stanford Encyclopedia of Philosophy, Edward N. Zalta (ed.).

Zaykin DV, Zhivotovsky LA, Westfall PH, Weir BS: Truncated product method for combining P-values. Genetic Epidemiology 2002, 22:170-185.