Available online at www.sciencedirect.com

ScienceDirect

Cognitive Systems Research 8 (2007) 227–236

Cognitive Systems RESEARCH

www.elsevier.com/locate/cogsys

Deconstructing and reconstructing ACT-R: Exploring the architectural space

Action editor: Danilo Fum

Terrence C. Stewart *, Robert L. West

Carleton Cognitive Modelling Lab, Institute of Cognitive Science, Carleton University, 1125 Colonel By Drive, Ottawa, Ontario, Canada K1S 5B6

Received 11 February 2007; accepted 15 June 2007
Available online 1 July 2007

Abstract

Evaluating variations in the structure of computational models of cognition is as important as evaluating variations in the numerical parameters of such models. However, computational models tend not to be organized in such a way as to directly support such research. To address this need, we have taken the well-known cognitive architecture ACT-R, reduced it to its fundamental components, and reconstructed it. Our new system, Python ACT-R, facilitates exploration of the space of possible models and architectures based on the core ACT-R theory. The result has enabled us to examine the possibility of using basic ACT-R components such as the declarative memory system in new ways; for example, as the basis for a new visual attention system. Python ACT-R allows the same model definition syntax to be used to define both ACT-R models and new ACT-R components, as well as making explicit the processes specified by the ACT-R theory.

Keywords: Cognitive modeling; ACT-R; Cognitive architectures; Philosophy of modeling

Cognitive modeling, as a science, involves not only building and evaluating models, but also effectively communicating these results and enabling further research with these models. Communication becomes increasingly difficult as the audience knows less and less about the particular approach that was used. For example, it is easier for neural network researchers to communicate to other neural network researchers than it is for them to communicate to production system researchers, and it is even harder to communicate to those who do not use cognitive modeling at all.

The concept of a cognitive architecture was put forward by Newell (1990) to deal with the problem that the behavioral sciences, and psychology in particular, study the mind by dividing it up into specialized sub-fields, without attempting to assemble the results into an integrated model of the mind. Cognitive architectures are meant to be a way to do this. Anderson (1993) further clarified the concept of cognitive architectures with his distinction between frameworks, theories, and models; where frameworks are general claims about cognition, theories are specific formulations about how the frameworks operate, and models are the theories applied to specific tasks and behaviors. Cognitive architectures are theories about how the mind integrates different processes to produce thoughts and behaviors.

Because cognitive architectures tend to be complex, they are often expressed as computer programs. Within the cognitive modeling community this is generally regarded as a good thing because it means that the theory is precisely specified, avoiding ambiguity and vague statements. However, outside the community it is often viewed with suspicion. Indeed, it is often felt that modelers merely write computer code that mimics the human data (i.e., that modeling is merely descriptive). In order for non-modelers

* Corresponding author.
E-mail addresses: terry@ccmlab.ca (T.C. Stewart), robert_west@carleton.ca (R.L. West).

to be convinced that this is not happening, we need clear descriptions of the architectures and how they are used.

In this paper, we start with an overall description of the process of modeling. We discuss the importance of examining the effects of the numerical parameters of models, and extend this to also cover the non-numerical parameters that specify the overall architecture of the model. Although it is as important to explore the architectural variations of a model as it is to explore the parameter-based variations, it is currently difficult to do so without extensive programming skills. The second half of this paper describes our progress in rebuilding the ACT-R cognitive architecture to support this sort of exploration. The result is a system called Python ACT-R. The goal of Python ACT-R is to make varying the architecture as easy as varying the numerical parameters of the architecture.

1. Modeling methodology

When creating a computational cognitive model of a phenomenon, we are not merely attempting to create a computer program which produces the same outputs for a given input as is empirically observed. If this is all that is required for a successful model, then we would be satisfied with trivial special-case models that merely encode a description of the phenomenon. Instead, a scientifically useful cognitive model will be based in a particular theory of the functional architecture that underlies the behavior itself. Importantly, this fundamental architecture must be able to support multiple phenomena, allowing for generalization to new situations.

In this section, we describe the approach to modeling that has led us to our re-conception of ACT-R, and which is the basis of our modeling work. For the purposes of this paper, we will focus on the concept of *variation* among models. That is, how does the performance of a model change as aspects of that model are changed? And how should this be interpreted within a scientific method? For further discussion of these issues, including more attention to the companion aspects of model measurement and communication, see our previous work (e.g., Stewart, 2006).

1.1. Model variation

It is common within the general practice of computational modeling for there to be a certain flexibility in terms of parameter settings. That is, the overall theory may specify the structures and the processes occurring in the model, but the particular numerical values for a learning parameter or a reaction speed may be left unspecified. The adjustment of these parameters affects the overall performance, and a major part of the modeling process tends to involve adjusting these parameters to find values for which the model behavior matches the empirical data.

This adjustment, however, opens the door to a common criticism of computational models. If there are enough parameters in the model it is possible that it can be adjusted to produce *any* possible (or plausible) result. This "parameter tweaking" has been used to call into question the entire process of computational modeling (e.g., Roberts & Pashler, 2000). If a theory is flexible enough to produce any reasonable outcome, then we must question the usefulness of that theory.

However, for most theories, we do not have a situation where *any* outcome is possible. Instead, parameters can be adjusted to produce a certain range of outcomes, constrained by the overall structure of the model. Traditionally, modelers have focused on finding the one parameter setting which "best fits" the observed data (usually by sampling a portion of the space of parameters). Unfortunately, reporting this one best parameter setting does not provide information about the flexibility of the model. However, when the same modeling architecture is used for a variety of different tasks, it is possible to identify particular parameter values that are consistently used. ACT-R researchers have identified particular "canonical" parameter values that should generally be used, and any deviation from these values should be seen as requiring further explanation. For example, few ACT-R models have a production firing time that is not exactly 50 ms. Other parameters, such as the latency and activation noise levels (which will be described later in this paper) tend to vary more widely between models.

In contrast, our approach is to identify *ranges* of parameter settings that produce models that match to *within a certain degree*. That is, instead of reporting that the model performs *best* if parameter *X* is set to 5, we instead indicate that the model performs *well* if the parameter is between 2 and 6, for example. This, in turn, requires a measurement of the degree of match between the model and reality. This can be a traditional measure such as mean-squared error or even correlation, but we recommend a measure of statistical equivalence that takes into account the confidence intervals of the model and real data (which are generally quite large). That is, instead of measuring the difference between the sample means of the model data and the human data, we would instead measure the maximum spread between the confidence intervals of the means of these two data sets. This avoids over-fitting to the sample. For more information, see Stewart (2006).

1.2. Exploring the model space

In addition to parameter values, this approach can also be applied to modifications to the underlying architecture itself (such as using slightly different structures or learning rules or other implementation details). Our approach is to consider these types of changes as qualitative, or non-numerical, parameter changes. The main issue with non-numerical parameter changes is the absence of well-defined, continuous parameter ranges. However, it should be noted that this is a problem for numerical parameters as well. After all, we are generally not proving that *every* parameter setting between 2 and 6 produces a good model; instead we