Programming a Cognitive Architecture with Simulated Neurons

How to Build a Brain: A Neural Architecture for Biological Cognition

By Chris Eliasmith

Oxford: Oxford University Press, 2013. 456 pp. ISBN 978-0-19-026212-9

The book presents one perspective of cognitive science and cognitive architectures, and discusses the Semantic Pointer Architecture. This Architecture is built on what is called the Neural Engineering Framework, and can be accessed using Nengo, a downloadable software package. The software and overall idea are quite interesting. What is particularly valuable in it, is the ability to implement and run relatively complex systems based solely on simulated spiking neurons.

The above-mentioned Neural Engineering Framework is a mechanism for translating vectors into neural representations. Two or more vectors can then be combined via neurally instantiated operations to produce an output vector. Of particular interest is a reversible binding operator that can be used to combined two vectors, and then reproduce either of the inputs from the output. *Binding* is a widely known difficulty for neural systems. When combined with the representation of a symbol as a vector, the binding mechanism allows a system to implement rules, and other types of symbol manipulation.

In other words, this is really important work, and leads to real functioning systems; the software can be readily downloaded and is straightforward to use. The apotheosis of the book is Spaun, a neural system that performs several interesting tasks, such as learning to write prototypical digits. It is a system that implements several subsystems in an integrated fashion and it is the basis of the cognitive architecture.

We should mention that while it is interesting to build systems from simulated spiking neurons, there are unacknowledged yet important shortcomings, both as a biological model of neural processing, and with the overall system as a neuro-cognitive model. There are two basic assumptions of the Neural Engineering Framework that are flawed from a neuro-biological standpoint, I believe. First is the vector processing mechanism, and second is binding via convolution, or indeed binding via any other vector combination mechanism.

The Neural Engineering Framework assumes that the primary representation is a vector. Elements of this vector are then translated into simulated neurons. This unnatural assumption or indeed shortcoming is referred to in the final chapter, saying "there is always the challenge of determining

the appropriate dimensionality of the space to be presented by a population of cells." However, it is unlikely that the brain is actually made up vectors of orthogonal sets of neurons. Moreover, in the Framework, a great deal of processing is done by combining these input vectors to produce output vectors. This levelling is also neuro-biologically suspect.

A second problem with the Neural Engineering Framework is binding via circular convolution. Binding is important to deal with the combination of symbols that humans use, for example in natural language, but also to combine features of objects. The final chapter states: "The particular choice of circular convolution for binding is difficult to directly motivate from the data." And it is. This could be expanded to state that there is no evidence that any vector combination mechanism is ever used to bind items in the brain.

While Eliasmith notes the controversy over binding via synchrony, there is evidence for it (Usher and Donnelly, 1998). Moreover, synchrony does not need to perform all of the brains' binding tasks. It may be used, for instance, for object binding (the red square task), but not for, say, complex language processing. Binding via synchrony can thus be available for pre-linguistic humans, and for other animals. Another form of binding is via short-term synaptic changes. This is not mentioned in the book, and could resolve all of Jackendoff's problems, restated by Eliasmith. However, it should be again noted that, unlike synchrony, but much like convolution, there is also no neural evidence for short-term synaptic change as a binding mechanism.

A third problem with the approach is that learning is here an add-on. It is added as part of the move from the Neural Engineering Framework to the Semantic Pointer Architecture. As in reality learning is ubiquitous, it is constantly on, and is the real benefit to neural systems over symbolic systems, thus it should be central. The add-on approach is evident in the poorly acknowledged weaknesses of the proposed learning algorithm. Firstly, it seems unlikely that the brain relies entirely on reinforcement learning; there is a lot to learn, and not enough feedback from the environment required by reinforcement learning. The book does describe the system categorising the visual representation of digits; in a sense this is some kind of symbol grounding, but it has not yet learned the semantics of the

symbols. This is the second problem, there is no sense of number, and thus no deep semantics. In Semantic Pointer Architecture, indeed the symbols must be ground by the programmer. The scientific community is not going to be able to build a *brain* that does anything that is very useful. It is going to need to make the *substrate*, and have the substrate learn.

In the biological brain, learning occurs by short-term changes in synaptic weight, longer-term changes, synaptic growth and death, and neural growth and death. There is some evidence that this is all Hebbian, but the learning rule used in the Semantic Pointer Architecture cannot accommodate all of these neural learning events. Of course all of this learning is poorly understood, and it is difficult to implement usefully. This is why the Neural Engineering Framework ignored it. By ignoring learning, the Framework can support standard symbolic and statistical algorithms in simulated neurons. Unfortunately, learning is not easily correctly added. While the Semantic Pointer Architecture does add a reasonable neural learning algorithm, the algorithm only accounts for simple tasks, and is at best an approximation of one instance of the learning algorithms in the brain.

Neurons are powerful computational devices. They are Turing complete, which means any algorithm can be implemented in them. While it is interesting to run systems based entirely on simulated neurons, what would be important is to use neurons the way humans and other animals use them to implement the real cognitive architecture.

Finally, to finish the line of criticism, the book largely ignores the large neuro-biological simulation literature. There are many other systems that simulate cognitive phenomena with simulated neurons (e.g. Rolls 2008). And much of this literature is more closely tied to data on biological neurons than is the work in this book.

Overall, thus, the book lacks self-criticism, though happily criticising other approaches. For example, there are the criticisms above about vectors, and binding. Another example is that the book evaluates the Semantic Pointer Architecture as more biologically realistic than the well-known Leabra. However, one may say that despite Leabra's use of rate-based simulated neurons, it is still more biologically realistic, because it uses a realistic topology.

The Semantic Pointer Architecture is thus not a very solid cognitive architecture, but it is still useful. Cognitive architectures are supposed to provide all the essential mechanisms for cognition. But this architecture does not provide, among other things, a way of learning a new rule. While good Cognitive Architectures are not complete minds, they should be easily extendable. For example, a system has been implemented in Anderson's popular ACT cognitive architecture that shows it is more difficult to drive while using a mobile phone. It is not clear how such a complex system could be developed in the Semantic Pointer Architecture. However, it is nevertheless a good thing that Eliasmith and colleagues are continuing to work with it. Its rule based nature, and ability to encode simple sensory and motor domains make it useful. It will be used for new experiments, though it will probably not permit as many simulations as done in ACT.

The neural systems from the Semantic Pointer Architecture could be usefully integrated with other simulated neural systems; this is one of the great benefits of simulated neural systems. Building a full brain is going to be difficult, and systems based on the Semantic Pointer Architecture should help bridge the gap from the current state of the art to eventual full brain simulation systems.

To summarize, the book was interesting. Yet bit painful to read. There is a sound introductory discussion to a range of neural, cognitive, and cognitive architecture issues with pointers to relevant literature, but Eliasmith frequently says about other issues that "I am unaware of any literature..." when there is such a literature, and in some cases he either must have been aware or should have been aware of it.

The book is recommended to students of cognitive architecture and neural programming, but they should read it with a critical eye.

References

Feldman, J. (1982). Dynamical Connections in Neural Networks. *Biological Cybernetics, 46*, 27-39.

Rolls, E. (2008), *Memory, attention, and decision-making: A unifying computational neuroscience approach.* Oxford: Oxford University Press

Usher, M. and Donnelly, N. (1998). Visual Synchrony Affects Binding and Segmentation in Perception, *Nature, 394*, 179-182