

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

Chris Eliasmith

June 24, 2011

Contents

1	The science of cognition	8
1.1	The last 50 years	8
1.2	How we got here	12
1.3	Where we are	21
1.4	The house of answers	25
1.5	Nengo: An introduction	29
I	How to build a brain	36
2	An introduction to brain building	37
2.1	Brain parts	37
2.2	Theoretical neuroscience	44
2.3	A framework for building a brain	47
2.3.1	Representation	50
2.3.2	Transformation	58
2.3.3	Dynamics	62
2.3.4	The three principles	68
2.4	Levels	71
2.5	Nengo: Neural representation	77
3	Biological cognition – semantics	88
3.1	The semantic pointer hypothesis	89
3.2	What is a semantic pointer?	94
3.3	Semantics: An overview	95
3.4	Shallow semantics	99
3.5	Deep semantics for perception	102
3.6	Deep semantics for action	112

CONTENTS 2

3.7	The semantics of perception and action	120
3.8	Nengo: Neural computations	124
4	Biological cognition – syntax	135
4.1	Structured representations	135
4.2	Neural binding	137
4.2.1	Without neurons	137
4.2.2	With neurons	142
4.3	Manipulating structured representations	148
4.4	Learning structural manipulations	154
4.5	Clean-up memory and scaling	156
4.6	Example: Fluid intelligence	161
4.7	Deep semantics for cognition	167
4.8	Nengo: Structured representations in neurons	172
5	Biological cognition – control	178
5.1	The flow of information	178
5.2	The basal ganglia	180
5.3	Basal ganglia, cortex, and thalamus	184
5.4	Example: Fixed sequences of action	188
5.5	Attention and the routing of information	192
5.6	Example: Flexible sequences of action	201
5.7	Timing and control	206
5.8	Example: The Tower of Hanoi	210
5.9	Nengo: Question answering	218
6	Biological cognition – memory and learning	228
6.1	Extending cognition through time	228
6.2	Working memory	230
6.3	Example: Serial list memory	235
6.4	Biological learning	240
6.5	Example: Learning new actions	247
6.6	Example: Learning new syntactic manipulations	251
6.7	Nengo: Learning	264
7	The Semantic Pointer Architecture (SPA)	269
7.1	A summary of the SPA	269
7.2	An SPA unified network	272

7.3	Tasks	276
7.3.1	Copy drawing	276
7.3.2	Recognition	277
7.3.3	Reinforcement learning	278
7.3.4	Serial memory	278
7.3.5	Counting	279
7.3.6	Question answering	280
7.3.7	Rapid variable creation	281
7.3.8	Fluid reasoning	283
7.3.9	Discussion	283
7.4	A unified view: Symbols and probabilities	285
7.5	Nengo: Large-scale modeling	290
II	Is that how you build a brain?	295
8	Evaluating cognitive theories	296
8.1	Introduction	296
8.2	Quintessential cognitive criteria (QCC)	297
8.2.1	Representational structure	298
8.2.1.1	<i>Systematicity</i>	298
8.2.1.2	<i>Compositionality</i>	299
8.2.1.3	<i>Productivity</i>	300
8.2.1.4	<i>The massive binding problem</i>	302
8.2.2	Performance concerns	303
8.2.2.1	<i>Syntactic generalization</i>	303
8.2.2.2	<i>Robustness</i>	304
8.2.2.3	<i>Adaptability</i>	306
8.2.2.4	<i>Memory</i>	307
8.2.2.5	<i>Scalability</i>	309
8.2.3	Scientific merit	311
8.2.3.1	<i>Triangulation (contact with more sources of data)</i>	311
8.2.3.2	<i>Compactness</i>	312
8.3	Conclusion	313
9	Theories of cognition	315
9.1	The state of the art	315
9.1.1	ACT-R	317

9.1.2	Synchrony-based approaches	320
9.1.3	Neural blackboard architecture (NBA)	323
9.1.4	The integrated connectionist/symbolicist architecture (ICS)	327
9.1.5	Leabra	329
9.1.6	Dynamic field theory (DFT)	333
9.2	An evaluation	335
9.2.1	Representational structure	335
9.2.2	Performance concerns	338
9.2.3	Scientific merit	345
9.2.4	Summary	349
9.3	The same...	351
9.4	... but different	353
10	Consequences and challenges	362
10.1	Conceptual consequences	362
10.1.1	Representation	363
10.1.2	Concepts	366
10.1.3	Inference	369
10.1.4	Dynamics	371
10.2	Challenges for the SPA	373
10.2.1	Representation	375
10.2.2	Architecture	376
10.2.3	Scaling	378
10.3	Conclusion	380
A	Mathematical derivations for the NEF	381
A.1	Representation	381
A.1.1	Encoding	381
A.1.2	Decoding	382
A.2	Transformation	383
A.3	Dynamics	384
B	Mathematical derivations for the SPA	387
B.1	Binding and unbinding HRRs	387
B.2	Learning high-level transformations	389
B.3	Ordinal serial encoding model	390
B.4	Spike-timing dependent plasticity	391
B.5	Number of neurons for representing structure	392

CONTENTS	5
C SPA model details	394
C.1 Tower of Hanoi	394
Bibliography	397

Preface

I suspect that when most of us want to know “how the brain works”, we are somewhat disappointed with an answer that describes how individual neurons function, or an answer that picks out the parts of the brain that have increased activity while reading words, or an answer that describes what chemicals are in less abundance when someone is depressed. These are all parts of an answer, no doubt, but the parts need to be put together.

Instead, I suspect that the kind of answer that we would like is the kind of answer that early pioneers in cognitive science were trying to construct. Answers like those provided by Allen Newell (1990) in his book “Unified theories of cognition.” Newell was interested in understanding how the “wheels turn” and the “gears grind” during cognitive behavior. He was interested in the details of brain function during all the variety of tasks that we might call “cognitive”. Newell’s answer, while interesting and important, was ultimately not found to be very convincing by many. Perhaps this is why most researchers in the behavioral sciences – which I take to include psychology, neuroscience, psychophysics, cognitive science, linguistics, and parts of engineering, computer science, statistics, and so on – have since focused on characterizing the mechanisms of relatively small parts of the brain. And notably, many of these characterizations do not fit well with the answer Newell was attempting to propose.

Regardless, I think we may now be in a position to again try to provide the kind of answer Newell was after, and I think we may be able to integrate many of the pieces of the puzzle coming from these myriad different disciplines. In the grandest (and hence least plausible) sense, that is the purpose of this book. It is an attempt to provide basic principles and an outline of an architecture for building the wheels and gears needed to drive cognition, in a way that integrates much of the evidence from the behavioral sciences.

The architecture I propose and develop here is called the “Semantic Pointer Architecture” (SPA), for reasons that will become clear in chapter 3. I believe a central departure of the SPA from past attempts to articulate theories of cognition, is that it does not begin with cognition, it ends there. That is, it does not start with

the suggestion that cognitive systems are like computers, or that they are best described as abstract nodes connected together in a network. Instead, it begins with biology. That is why it is a theory of how to build a *brain*. It is in virtue of starting with biology that the SPA is able to draw on evidence from (and make predictions about) disciplines whose results are fundamentally the product of biological processes. To the best of my knowledge, there are currently no systematic methods for relating biological data to our high-level understanding of the complex dynamics, and sophisticated representational properties of cognitive systems.

Even in the likely scenario that the SPA is found wanting, it should at least make clear that starting with biology mandates that the specific computational and representational properties of neural systems be taken into account when developing an understanding of higher-order cognition. Not doing so makes it difficult, if not impossible, to relate the two (biology and cognition) at the end of the day. So, the architecture I propose adopts cognitively relevant representations, computations, and dynamics that are natural to implement in large-scale, biologically realistic neural networks. In short, the SPA is centrally inspired by understanding cognition as a biological process – or what I refer to as “biological cognition”.

Perhaps unsurprisingly, before I can describe this architecture, I need to provide an outline of what is expected of a cognitive theory and a systematic method for understanding large-scale biologically realistic systems: these are the purposes of chapters 1 and 2 respectively. In chapters 3-6 I describe the four central aspects of the SPA that allow it to capture biological cognition. In chapter 7 I provide a summary of the architecture, and a description of its application to constructing a single model that is able to perform a multitude of perceptual, motor, and cognitive tasks without user intervention. In the remainder of the book, I argue for a set of criteria for evaluating cognitive theories (chapter 8), I describe current state-of-the-art approaches to cognitive modeling, evaluate them with respect to those criteria, and compare and contrast them with the SPA (chapter 9), and I highlight future directions of the SPA and discuss several conceptual consequences of adopting this approach to understanding cognitive function.

Despite proposing a specific architecture, this book is also usable independently of those commitments. That is, there is a practical sense to the title of this book: I provide tutorials for constructing single neuron-level, spiking models at the end of each chapter. For those who wish, brain building can be more than just an abstract, theoretical pursuit. These tutorials employ the neural simulator Nengo, that has been developed by my lab. The first tutorial describes how to set up and run this graphical simulator. For those wishing to use this book as a text for a course, please visit <http://compneuro.uwaterloo.ca/cnrglab/> for further

resources and information.

- Query: Have a section on ‘all the math you need to read this book’: 1. discussion of scalars and vectors, as ‘arrows’ or just as points x,y on a graph, as collections of numbers; 2. distinguishing linear from nonlinear functions and functions in general; 3. integration 4. The math that goes beyond anything described here is in the appendices 5. anything on dynamics? super simple differential equation 6. dot product? 7. any other notation????

Acknowledgements

- TODO

Chapter 1

The science of cognition

Questions are the house of answers. – Alex, age 5

1.1 The last 50 years

“What have we actually accomplished in the last 50 years of cognitive systems research?” was the pointed question put to a gathering of experts from around the world. They were in Brussels, Belgium, at the headquarters of the European Union funding agency, and were in the process of deciding how to divvy up about 70 million euros of funding. The room went silent. Perhaps the question was unclear. Perhaps so much had been accomplished that it was difficult to know where to start an answer. Or, perhaps even a large room full of experts in the field did not really know any generally acceptable answers to that question.

The point of this particular call for grant applications was to bring together large teams of researchers from disciplines as diverse as neuroscience, computer science, cognitive science, psychology, mathematics, and robotics, in order to unravel the mysteries of how biological cognitive systems are so impressively robust, flexible, adaptive, and intelligent. This was not the first time such a call had been made. Indeed, over the course of the last four or five years this agency has funded a large number of such projects, spending close to half a billion euros. Scientifi-

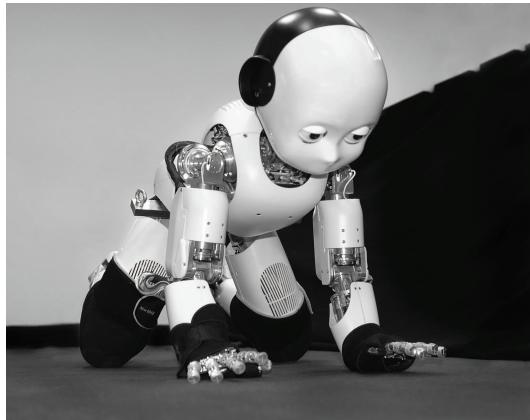


Figure 1.1: The iCub. The iCub is an example of a significant recent advance in robotics. See <http://www.robotcub.org/index.php/robotcub/gallery/videos> for videos. © The RobotCub Consortium. Image by Lorenzo Natale, reproduced with permission.

cally speaking, important discoveries have been made by the funded researchers. However, these discoveries tend not to be of the kind that tell us how to better construct integrated, truly *cognitive* systems. Rather, they are more often discoveries in the specific disciplines that are taking part in these “Integrated Projects.”

For instance, sophisticated new robotic platforms have been developed. One example is the iCub (figure 1.1), which is approximately the size of a two-year-old child, and has over 56 degrees of freedom.¹ The iCub has been adopted for use by researchers in motor control, emotion recognition and synthesis, and active perception. But, of course, iCub is not a cognitive system. It may be a useful testbed for a cognitive system, no doubt. It may be a wonder of high-tech robotics engineering, indeed. But, it is not a cognitive system.

So the pointed question still stands: “What have we actually accomplished in the last 50 years in cognitive systems research?” That is, what do we now know about how cognitive systems work, that we did not know 50 years ago? Pessimistically, it might be argued that we do not know too much more than we knew 50 years ago. After all, by 1963 Newell and Simon had described in detail the program GPS (General Problem Solver). This program, which was an extension of work that they had started in 1959, is the first in a line of explanations of human

¹A degree of freedom is an independent motion of an object. So, moving the wrist up and down is one degree of freedom, as is rotating the wrist, or moving a finger up and down.

cognitive performance that relied on production systems. Production systems are, historically, by far the most influential approach to building cognitive systems. Simply put, a production system consists of a series of productions, or if-then rules, and a control structure. The job of the control structure is to match a given input to the “if” part of these productions to determine an appropriate course of action, captured by the “then” part.

In 1963, GPS had all of these features, and it put them to good use. This “program that simulates human thought” was able to solve elementary problems in symbolic logic entirely on its own, and went through steps that often matched those reported by people solving the same problem. GPS could be given a novel problem, analyze it, attempt a solution, retrace its steps if it found a dead end (i.e., self-correct), and eventually provide an appropriate solution (Newell and Simon, 1963).

The success of GPS lead to several other cognitive architectures, all of which had production systems as their core. Best known amongst these are Soar (Newell, 1990) and ACT (Adaptive Control of Thought; Anderson, 1983). Despite the many additional extensions to the GPS architecture that were made in these systems, they share a reliance on a production system architecture. The dominance of the representational, computational, and architectural assumptions of production systems has resulted in such systems being called “classical” approaches to cognitive systems.

While classical approaches had many early successes, such as finding convincing solutions to cryptarithmetic problems and solving complex planning tasks, the underlying architecture does not seem well-suited to interacting with a dynamic, real-world environment, or explaining the evolution of real cognitive systems through time (Eliasmith, 1995). In fairness, more recent work on ACT and ACT-R (Anderson, 1996), has taken dynamics more seriously, explaining reaction times across a wide variety of psychological tasks (Anderson et al., 2004). Nevertheless, explaining reaction times addresses only a very small part of cognitive system dynamics in general, and the ACT explanations rely on a 50ms “cycle time,” which itself is not explained (Anderson, 2007, p. 61).

In fact, those centrally concerned with the dynamics of cognition largely eschew production systems (Port and van Gelder, 1995; Schöner, 2008). If you want to build a real-world cognitive system – one that actually interacts with the physics of the world – then the most unavoidable constraints on your system are dynamic interactions with the world through perception and action. Roboticists, as a result, seldom use production systems to control their systems. Instead, they carefully characterize the dynamics of their system, attempt to understand how to control

such a system when it interacts with the difficult-to-predict dynamics of the world, and look to perception to provide guidance for that control. If-then rules are seldom used. Differential equations, statistics, and signal processing are the methods of choice. Unfortunately, it has remained unclear how to use those same methods for characterizing high-level *cognitive* behavior – like language, complex planning, and deductive reasoning – behaviors which classical approaches have the most success at explaining.

In short, there is a broad gap in our understanding of real, cognitive systems: on the one hand there are the approaches centered on dynamic, real-world perception and action; on the other hand there are the approaches centered on higher-level cognition. Unfortunately, these approaches are not the same. Nevertheless, it is obvious that perception/action and high-level cognition are not two independent parts of one system. Instead, these two aspects are, in some fundamental way, integrated in cognitive animals such as ourselves. Indeed, a major theme of this book is that biology underwrites this integration. But for now, I am only concerned to point out that classical architectures are not obviously appropriate for understanding all aspects of real cognitive systems. This, then, is why we cannot simply say, in answer to the question of what has been accomplished in the last 50 years, that we have identified *the* (classical) cognitive architecture.

However, this does not mean that identifying such architectures is without merit. On the contrary, one undeniably fruitful consequence of the volumes of work that surrounds the discussion of classical architectures is the identification of criteria for what counts as a cognitive system. That is, when proponents of classicism were derided for ignoring cognitive dynamics, one of their most powerful responses was to note that their critics had no truly *cognitive* systems to replace theirs with. This resulted in a much clearer specification of what a cognitive system was. So, I suspect that there would be agreement amongst most of the experts gathered in Brussels as to what has been accomplished to these 50 years. Indeed, the accomplishments are not in the expected form of an obvious technology, a breakthrough method, or an agreed upon theory. Instead, the major accomplishments have been in clarifying what the questions are, in determining what counts as a cognitive system, and in figuring out how we are most likely to succeed in explaining such systems (or, perhaps more accurately, how we are *not* likely to succeed).

If true, this is no mean feat. Indeed, it is even more true in science than elsewhere that, as economic Nobel laureate Paul A. Samuelson has observed, “good questions outrank easy answers.” If we actually have more thoroughly identified criteria for distinguishing cognitive from non-cognitive systems, and if we really

have a good sense of what methods will allow us to understand how and why systems can successfully meet those criteria, we have accomplished a lot. Ultimately, only time will tell if we are on the right track. Nevertheless, I believe there is an overall sense in the field that we have a better idea of what we are looking for in an explanation of a cognitive system than we did 50 years ago – even if we do not yet know what that explanation is. Often, progress in science is more about identifying specific questions that have uncertain answers than it is about proposing specific answers to uncertain questions.

The goal of this first chapter, then, is to identify these cognitive criteria and articulate some questions arising from them. These appear in sections 1.3 and 1.4 respectively. First, however, it is worth a brief side trip into the history of the behavioral sciences to situate the concepts and methods that have given rise to these criteria.

1.2 How we got here

Previously, I identified the “classical” approach to understanding cognition. And, I contrasted this approach with one that is more centrally interested in the characterization of the dynamics of behavior. However, much more needs to be said about the relationship between classical and non-classical approaches in order to get a general lay-of-the-land in cognitive systems theorizing. Indeed, much more can be said than I will say here (see, e.g., Bechtel and Graham, 1999). My intent is to introduce the main approaches in order to: 1) identify the strengths and weaknesses of these approaches, both individually and collectively; 2) state and clarify the cognitive criteria mentioned earlier; and, ultimately, 3) outline a novel theory of biological cognition in later chapters.

In the last half century, there have been three major approaches to theorizing about the nature of cognition. Each approach has relied heavily on a preferred metaphor for understanding the mind/brain. Most famously, the classical approach (aka “symbolicism”, or Good Old-fashioned Artificial Intelligence (GO-FAI)), relies on the “mind as computer” metaphor. Under this view, the mind is the software of the brain. Jerry Fodor, for one, has argued that the impressive theoretical power provided by this metaphor is good reason to suppose that cognitive systems have a symbolic “language of thought” which, like a computer program-

ming language, expresses the rules that the system follows (Fodor, 1975). Fodor claims that this metaphor is essential for providing a useful account of how the mind works. Production systems, which I have already introduced, have become the preferred implementation of this metaphor.

A second major approach is “connectionism” (aka the Parallel Distributed Processing (PDP) approach or New-fangled Artificial Intelligence (NFAI)). In short, connectionists explain cognitive phenomena by constructing models that consist of large networks of typically identical nodes, that are connected together in various ways. Each node performs a simple input/output mapping. However, when grouped together in sufficiently large networks, the activity of these nodes is interpreted as implementing rules, analyzing patterns, or performing any of several other cognitively-relevant behaviors. Connectionists, like the symbolicists, rely on a metaphor for providing explanations of cognitive behaviors. This metaphor, however, is much more subtle than the symbolicist one; these researchers presume that the functioning of the mind is like the functioning of the brain. The subtlety of the “mind *as* brain” metaphor lies in the fact that connectionists also hold that the mind *is* the brain. However, when providing *cognitive* descriptions, it is the metaphor that matters, not the identity. In deference to the metaphor, the founders of this approach call it “brain-style” processing, and claim to be discussing “abstract networks” (Rumelhart and McClelland, 1986). In other words, their models are not supposed to be direct implementations of neural processing, and hence cannot be directly compared to the kinds of data we gather from real brains. This is not surprising since the computational and representational properties of the nodes in connectionist networks bear little resemblance to neurons in real biological neural networks.²

The final major approach to cognitive systems in contemporary cognitive science is “dynamicism”, and is often closely allied with “embedded” or “embodied” approaches to cognition. Proponents of dynamicism also rely heavily on a metaphor for understanding cognitive systems. Most explicitly, van Gelder employs the Watt Governor as a metaphor for how we should characterize the mind (van Gelder, 1995). In general, dynamicist metaphors rely on comparing cognitive systems to other, continuously coupled, nonlinear, dynamical systems (e.g. the weather). Van Gelder’s metaphor is useful because it has played a central role in his arguments for the novelty of dynamicist approaches, and the metaphor is a simple one.

The Watt Governor is a mechanism for controlling (i.e. “governing”) the speed

²As discussed in chapter 10 of (Bechtel and Abrahamsen, 2001).

of an engine shaft under varying loads. It is named after James Watt because he used it extensively to control steam engines, though the same mechanism was in use before Watt to control wind and water mills. Figure 1.2 depicts a Watt Governor. It consists of two weights attached to movable arms that are connected to a vertical shaft. The vertical shaft is driven by an engine, so as the engine spins the vertical shaft more quickly, centripetal forces cause the weights to be raised. The Governor is also connected to a throttle controlling the engine. Presuming the engine is being used to do work (e.g., driving a saw to cut wood), it is subject to varying loads (e.g., the presence or absence of a board to cut). With a constant load on the engine, the vertical shaft spins a constant speed and the weights maintain a given position. If the load decreases, the shaft speed increases, causing the weights to move higher, decreasing the throttle until the desired speed is again reached. Increasing the load has the opposite effect.

It is through his analysis of the best way to characterize this dynamic system that van Gelder argues for understanding cognitive systems as non-representational, low-dimensional, dynamic systems. Like the Watt Governor, van Gelder maintains, cognitive systems are essentially dynamic and can only be properly understood by characterizing their state changes through time. The “mind as Watt Governor” metaphor suggests that trying to impose any kind of discreteness, either temporal or representational, will lead to a mischaracterization of cognitive systems.

This same sort of analysis – one which highlights the importance of dynamics – highlights the essential coupling of cognitive systems to their environment (van Gelder and Port, 1995). Dynamic constraints are clearly imposed by the environment on the success of our behavior (we must see and avoid the cheetah before it eats us). If our high-level behaviors are built on our low-level competencies, then it is not surprising that identifying this important role of the environment has lead several researchers to emphasize the fact that real cognitive systems are embedded within a specific environment, with specific dynamics. Furthermore, they have argued, the nature of that environment can have significant impacts on what cognitive behaviors are realized (Clark, 1997; Hutchins, 1995). Because many of the methods and assumptions of dynamicism and embedded approaches are shared, in this discussion I group both under the heading of “dynamicism.”

Notably, each of symbolism, connectionism, and dynamicism, rely on metaphor not only for explanatory purposes, but also for developing the conceptual foundations of their preferred approach to cognitive systems. For symbolists, the properties of Turing machines become shared with minds because they are the same kind of computational system. For connectionists, the character of representa-

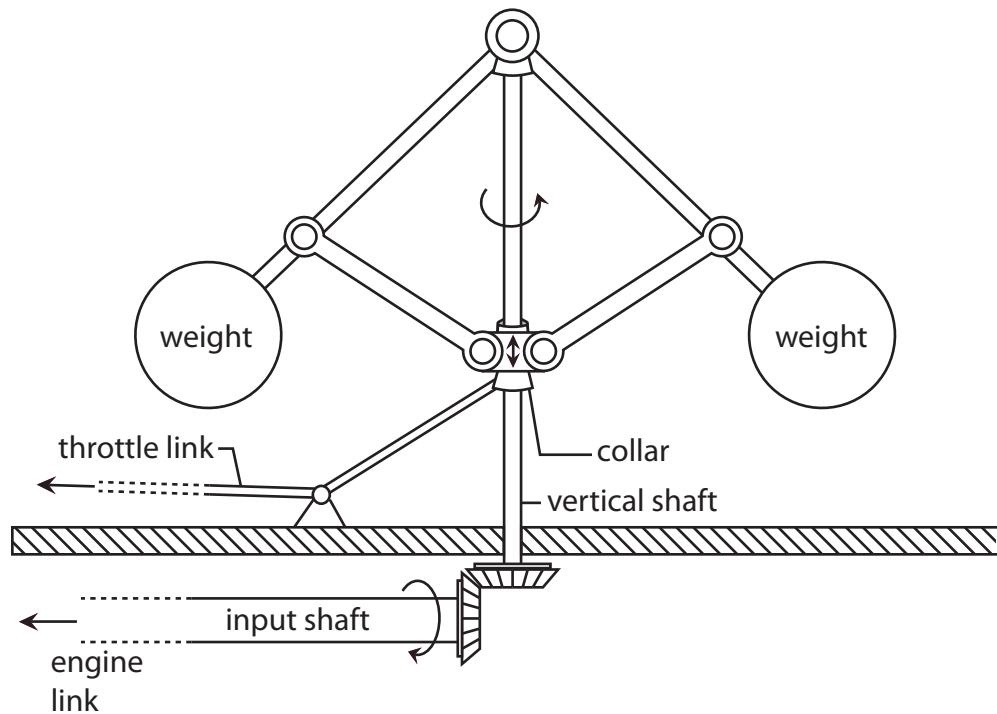


Figure 1.2: The Watt Governor. Two weights rotate around a vertical shaft driven by an engine. The centripetal force created by the rotation of the vertical shaft causes the weights to be pushed outwards and raises a collar which is attached to the weighted arms. The collar is linked to the engine throttle and as it rises it decreases the throttle, reducing the speed of the input shaft. The system settles into a state where the throttle is open just enough to maintain the position of the weights. Increasing or decreasing the load on the engine disturbs this equilibrium and causes the governor to readjust the throttle until the system stabilizes again.

tion changes dramatically under their preferred metaphor. Mental representations are taken to consist of “sub-symbols” associated with each node, while “whole” representations are real-valued vectors in a high-dimensional property space.³ Finally, because the Watt Governor is best described by the mathematics of dynamic systems theory, which makes no reference to computation or representation, dynamicists claim that our theories of mind need not appeal to computation or representation either (van Gelder, 1998).

I have argued elsewhere that our understanding of cognitive systems needs to move beyond such metaphors (Eliasmith, 2003). We need to move beyond metaphors because, in science, metaphorical thinking can sometimes unduly constrain available hypotheses. This is not to deny that metaphors are incredibly useful tools at many points during the development of scientific theory. It is only to say that, sometimes, metaphors only go so far. Take, for instance, the development of the current theory of the nature of light. In the nineteenth century, light was understood in terms of two metaphors: light as a wave, and light as a particle. Thomas Young was the best known proponent of the first view, and Isaac Newton was the best known proponent of the second. Each used their favored metaphor to suggest new experiments, and develop new predictions.⁴ Thus, these metaphors played a role similar to that played by the metaphors discussed above in contemporary cognitive science. However, as we know in the case of light, both metaphors are false. Hence the famed “wave-particle duality” of light: sometimes it behaves like a particle; and sometimes it behaves like a wave. Neither metaphor by itself captures all the phenomena displayed by light, but both are extremely useful in characterizing some of those phenomena. So, understanding what light *is* required moving beyond the metaphors.

I believe that the same is true in the case of cognition. Each of the metaphors mentioned above has some insights to offer regarding certain phenomena displayed by cognitive systems. However, none of these metaphors is likely lead us to all of the right answers. Thus, we ideally want a way of understanding cognitive systems that draws on the strengths of symbolism, connectionism, and dynamicism, but does not depend on the metaphors underlying any of these approaches.

In fact, I believe we are in a historically unique position to come to this kind

³See, for example, (Smolensky, 1988). Notably, there are also connectionist models that take activities of individual nodes to be representations. These are still treated unlike symbolic representations.

⁴For a detailed description of the analogies, predictions, and experiments, see (Eliasmith and Thagard, 1997).

of understanding. This is because we currently have more detailed access to the underlying biology of cognition than ever before. We can record large-scale blood flow in the brain during behavior using fMRI, we can image medium-sized networks of cells during the awake behavior of animals, we can record many individual neurons inside and outside of the functioning brain, and we can even record the opening and closing of individual channels (which are about 200 nanometers in size) on the surface of a single neural cell.

This kind of information is exactly what we need to free ourselves of high-level guiding metaphors. To be clear, I am not endorsing the claim that we should get rid of all metaphors. I by no means think that is plausible. Instead, I am advocating that we reduce our reliance on metaphors *as theoretical arbiters*. That is, we need to remove them from a role in determining what is a good explanation of cognition and what is not. If we can relate our cognitive explanations to specific biological measurements, then it is that data, not a metaphor, that should be the determiner of the goodness of our explanation. If we can identify the relationship between cognitive theories and biological data, then we can begin to understand cognitive systems for what they are: complex, dynamic, biological systems. The three past approaches I have discussed provide few hints as to the nature of this relationship.

One reason such hints are scarce is that, historically speaking, theories regarding cognitive systems mainly come from psychology, an offshoot of philosophy of mind in the late 19th century. At the time, very little was known of the biology of the nervous system. There was some suggestion that the brain was composed of neurons, but this was highly contentious. Certainly there was little understanding of how cells could be organized to control our simple behaviors, let alone our complex, cognitive ones.

So, cognitive theorizing has proceeded without much thought about biology. In the early days of psychology, most such theories were generated by the method of introspection: i.e., sitting and thinking very carefully, so as to discern the components of cognitive processing. Unfortunately, different people introspected different things, and so there was a crisis in “introspectionist” psychology. This crisis was resolved by “behaviorist” psychologists who simply disallowed introspection. The only relevant data for understanding cognitive systems was data that could be gleaned from the “outside”, i.e., from behavior.

While much is sometimes made of the difference between “philosophical” and “psychological” behaviorism, there was general agreement on this much: internal representations, states, and structures are irrelevant for understanding the behavior of cognitive systems. For psychologists, like Watson and Skinner, this was true

because only input/output relations are scientifically accessible. For philosophers, like Ryle, this was true because mental predicates (like “believes”, “wants”, etc.), if they were to be consistent with natural science, must be analyzable in terms of behavioral predicates. In either case, looking inside the “black box” that was the system of study, was prohibited.

Interestingly, engineers of the day respected a similar constraint. In order to understand dynamic physical systems, the central tool they employed was classical control theory. Classical control theory, perhaps notoriously, only characterizes physical systems in terms of their input/output relationship. As a result, classical control theory is limited to designing non-optimal, single-variable, static controllers and depends on graphical methods, rules of thumb, and does not allow for the inclusion of noise.⁵ While the limitations of classical controllers and methods are now well-known, they nevertheless allowed engineers to build systems of kinds they had not systematically built before: goal-directed systems.

While classical control theory was practically useful, especially in the 1940s when there was often a desire to blow things up (the most common goal at which such systems were directed), some researchers thought the theory had more to offer. They suggested that classical control theory could provide a foundation for describing living systems as well. Most famously, the interdisciplinary movement founded in the early 1940s known as “cybernetics” was based on precisely this contention.⁶ Cyberneticists claimed that living systems, like classical control systems, were essentially goal-directed systems. Thus, closed-loop control should be a good way to understand the behavior of living systems. Given the nature of classical control theory, cyberneticists focused on characterizing the input/output behavior of living systems, not their internal processes. Unfortunately for cyberneticists, in the mid-50s there was a massive shift in how cognitive systems were viewed.

With the publication of a series of seminal papers (including Newell et al. (1958), Miller (1956), Bruner et al. (1956), and Chomsky (1959)), the “cognitive revolution” took place. One simplistic way to characterize the move from behaviorism to “cognitivism” is that it became no longer taboo to look inside the black box. Quite the contrary: internal states, internal processes, and internal representations became standard fare when thinking about the mind. Classical control theory no longer offered the kinds of analytic tools that mapped easily onto this new way of conceiving complex biological behavior. Instead, making sense of the

⁵For a succinct description of the history of control theory, see (Lewis, 1992).

⁶For a statement of the motivations of cybernetics, see (Rosenblueth et al., 1943).

insides of that black box was heavily influenced by concurrent successes in building and programming computers to perform complex tasks. Thus, many early cognitive scientists saw, when they opened the lid of the box, a computer. As explored in detail by Jerry Fodor, “[c]omputers show us how to connect semantical [meaning-related properties] with causal properties for *symbols*” (Fodor, 1987, p. 18), thus computers have what it takes to be minds. Once cognitive scientists began to think of minds as computers, a number of new theoretical tools became available for characterizing cognition. For instance, the computer’s theoretical counterpart, the Turing machine, suggested novel philosophical theses, including “functionalism” (the notion that only the mathematical function computed by a system was relevant for its being a mind or not) and “multiple realizability” (the notion that a mind could be implemented (i.e. realized), in pretty much any substrate – water, silicon, interstellar gas, etc. – as long as it computed the appropriate function). More practically, the typical architecture of computers, the von Neumann architecture (which maps directly to the architecture of a production system), was thought by many to be relevant for understanding our cognitive architecture.

Eventually, adoption of the von Neumann architecture for understanding minds was seen by many as poorly motivated. Consequently, the early 1980s saw a significant increase in interest in the connectionist research program. As mentioned previously, rather than adopting the architecture of a digital computer, these researchers felt that an architecture more like that seen in the brain would provide a better model for cognition. It was also demonstrated that a connectionist architecture could be as computationally powerful as any symbolicist architecture. But despite the similar computational power of the approaches, the specific problems at which each approach excelled were quite different. Connectionists, unlike their symbolicist counterparts, were very successful at building models that could learn and generalize over the statistical structure of their input. Thus, they could begin to explain many phenomena not easily captured by symbolicists, such as object recognition, reading, concept learning, and other behaviors crucial to cognition.

For some, however, connectionists had clearly not escaped the influence of the mind as computer metaphor” after all, connectionists still spoke of representations, and thought of the mind as a kind of computer. Dynamicists, in contrast, suggested that if we want to know which functions a system can actually perform in the real world, we must know how to characterize the system’s dynamics. Since cognitive systems evolved in specific, dynamic environments, we should expect evolved control systems, i.e., brains, to be more like the Watt Governor – dynamic, continuous, coupled directly to what they control – than like a discrete-state Tur-

ing machine that computes over “disconnected” representations. As a result, dynamicists suggested that dynamic systems theory, not computational theory, was the right quantitative tool for understanding minds. They claimed that notions like “chaos”, “hysteresis”, “attractors”, and “state-space” underwrite the conceptual tools best-suited for describing cognitive systems. Notably absent from the list of relevant concepts are “representation” and “computation”.

As we have just seen, each of the three positions grew out of critical evaluation of previous positions. Connectionism was a reaction to the over-reliance on standard computer architectures for describing cognition. Dynamicism was a reaction to an under-reliance on the importance of time and our connection to the physical environment. Even symbolism was a reaction to its precursor, behaviorism, which had ruled out a characterization of cognition which posited states internal to the agent. Consequently, each of the metaphors has been chosen to emphasize different aspects of cognition, and hence driven researchers in these areas to employ different formalisms for describing their cognitive theories. Succinctly put, symbolicists use production systems, connectionists use networks of nodes, and dynamicists use sets of differential equations to explain our most complex behaviors.

While we can see the progression through these approaches as rejections of their predecessors, it is also important to note what was preserved. Symbolicism preserved the commitment to providing scientific explanations of cognitive systems. Connectionism retained a commitment to the notions of representation and computation so central to symbolist approaches. And dynamicism, perhaps the most self-conscious attempt to break away from previous methods, has not truly broken with the tradition of explaining the much the same set of complex behaviors. Rather, dynamicists have most convincingly argued for a shift in emphasis: they have made time a non-negotiable feature of cognition.⁷

Perhaps, then, a successful break from all of these metaphors will be able to relate each of the central methods to one another in a way that preserves the central insights of each. And, just as importantly, such a departure from past approaches should allow us to see how our theories relate to the plethora of data we have about brain function. I believe that the methods developed in this book can help us accomplish both of these goals, but I won’t make this argument until section 9.4. This is because such arguments cannot be made with any force until much

⁷Notably Newell, one of the main developers of production systems, was quite concerned with the timing of cognitive behavior. He dedicated a large portion of his magnum opus, *Unified Theories of Cognition*, to the topic. Nevertheless, time is not a necessary feature of production systems, and so his considerations seem, in many ways, after the fact (Eliasmith, 1996).

work is done. First, for instance, I need to use the historical background just presented to identify what I earlier claimed was the major advance of the the last 50 years: identifying criteria for evaluating the goodness of cognitive theories. Perhaps more importantly, I need to specify the methods that can underwrite such a departure – that is the objective of chapters 2-7.

1.3 Where we are

It will strike many as strange to suggest that there is anything like agreement on criteria for identifying good cognitive theories. After all, this area of research has been dominated by, shall we say, “vigorous debate” between proponents of each of the three approaches. For instance, it is true that there are instances of symbolicists calling connectionism “quite dreary and recidivist” (Fodor, 1995). Nevertheless, I believe that there is some agreement on what the target of explanation is. By this I do not mean to suggest that there is an agreed-upon definition of cognition. But rather that “cognition” is to most researchers what “pornography” was to justice Potter Stewart: “I shall not today attempt further to define [pornography]; and perhaps I could never succeed in intelligibly doing so. *But I know it when I see it.*”

Most behavioral researchers seem to know cognition when they see it, as well. There are many eloquent descriptions of various kinds of behavior in the literature, which most readers – regardless of their commitments to symbolism, connectionism, or dynamicism – recognize as cognitively relevant behavior. It is not as if symbolicists think that constructing an analogy is a cognitive behavior and dynamicists disagree. This is why I have suggested that we may be able to identify agreed-upon criteria for evaluating cognitive theories. To be somewhat provocative, I will call these the “Quintessential Cognitive Criteria”, or QCC for short. I should note that this section is only a first pass and summary of considerations for the proposed QCC. I return to a more detailed discussion of the QCC before explicitly employing them in chapter 8.

So what are the QCC? Let us turn to what have researchers said about what makes a system cognitive. Here are examples from proponents of each view:

- Dynamicism (van Gelder, 1995, p. 375-6): “[C]ognition is distinguished from other kinds of complex natural processes... by at least two deep features: on the one hand, a dependence on knowledge; and distinctive kinds of complexity, as manifested most clearly in the structural complexity of natural languages.”

- Connectionism (McClelland and Rumelhart, 1986, p. 13): Rumelhart and McClelland explicitly identify the target of their well-known PDP research as “cognition.” To address it, they feel that they must explain “motor control, perception, memory, and language”.
- Symbolicism (Newell, 1990, p. 15): Newell presents the following list of behaviors in order of their centrality to cognition: 1) problem solving, decision making, routine action; 2) memory, learning, skill; 3) perception, motor behavior; 4) language; 5) motivation, emotion; 6) imagining, dreaming, daydreaming.

These examples do not provide criteria for identifying good cognitive theories directly, but rather attempt to identify which particular aspects of behavior must be explained in order to successfully explain cognition. However, from such lists we can begin to distill some of the QCC that capture the intuitions motivating these researchers. Notice that there are several commonalities among the lists. First, language appears in all three. This is no surprise as language is often taken to be the pinnacle of human cognitive ability. But, there are other important shared commitments evident in these lists. For instance, each identify the importance of adaptability and flexibility. For van Gelder adaptability is evident through his identification of the dependence of cognitive behavior on knowledge. For the other two, explicit mention of memory and learning highlight a more general interest in adaptability. As well, while not in this specific quote from van Gelder, dynamicism is built on a commitment to the centrality of action and perception to cognition (Port and van Gelder, 1995). Perhaps surprisingly, then, it is the connectionists and symbolist who clearly identify motor control and perception as important to understanding cognition. It is clear that all three agree on the important role of these, more basic, processes.

While these are simple lists, I believe we can see in them the motivations for subsequent discussions explicitly aimed at identifying what it takes for a system to be cognitive. Let us briefly consider some of these more direct discussions. One of the first, and perhaps the most well-known, is provided by Fodor and Pylyshyn in their 1988 paper “Connectionism and cognitive architecture: A critical analysis.” While mainly a critique of the connectionism of the day, this paper also provides three explicit constraints on what it takes to be a cognitive system. These are productivity, systematicity, and compositionality. Productivity is the ability of a system to generate a large number of representations based on a few basic representations (a lexicon) and rules for combining them (a grammar). Systematicity refers to the fact that some sets of representations (generated productively) come

together. For instance, they suggest that cognitive systems cannot represent “John loves Mary” without thereby being able to represent “Mary loves John”. Finally, compositionality is the suggestion that the meaning of complex representations is a direct “composition” (i.e. adding together) of the meanings of the basic representations. Any good theory, they claim, must explain these basic features of cognition.

More recently, Jackendoff has dedicated his book to identifying challenges for a cognitive neuroscience of cognition (Jackendoff, 2002). In it he suggests that there are four main challenges to address when explaining cognition. Specifically, Jackendoff’s challenges are: 1) the massiveness of the binding problem (that very many basic representations must be bound to construct a complex representation); 2) the problem of 2 (how multiple instances of one representational token can be distinguished); 3) the problem of variables (how can roles (e.g. “subject”) in a complex representation be generically represented); and 4) how to incorporate long-term and working memory into cognition. Some of these challenges are closely related to those of Fodor and Pylyshyn, and so are integrated with them as appropriate in the QCC (see table 1.1).

The Fodor, Pylyshyn, and Jackendoff criteria come from a classical, symbolist perspective. In a more connectionist-oriented discussion, Don Norman summarizes several papers he wrote with Bobrow in the mid-70s, in which they argue for the essential properties of human information processing (Norman, 1986). Based on their consideration of behavioral data, they argue that human cognition is: robust (appropriately insensitive to missing or noisy data, and damage to its parts), flexible, and relies on “content-addressable” memory. Compared to symbolist considerations, the emphasis in these criteria has moved from representational constraints to more behavioral constraints, driven by the “messiness” of psychological data.

Dynamicists can be seen to continue this trend towards complexity in their discussions of cognition. Take, for instance, Gregor Schöner’s discussion in his article “Dynamical systems approaches to cognition” (Schöner, 2008). In his opening paragraphs, he provides examples of the sophisticated action and perception that occurs during painting, and playing in a playground. He concludes that “cognition takes place when organisms with bodies and sensory systems are situated in structured environments, to which they bring their individual behavioral history and to which they quickly adjust” (p. 101). Again, we see the importance of flexibility and robustness, with the addition of an emphasis on the role of the environment.

All three perspectives, importantly, take the target of their explanation to be a large, complex system. As a result, all three are suspect of “toy” models, that

over-simplify a given cognitive task, or domain. Nevertheless, the complexity of the system of interest makes some simplifications crucial – much disagreement remains on which simplifications are or are not appropriate. However, practical considerations have so far limited all theories to presenting simpler models, but, crucially, all approaches take it that any good theory must be scalable to larger, more complex problems, at least in principle. This demand for “scalability”, in some ways, is an ultimate expression of the noted historical trend towards expecting ever greater amounts of complexity from our cognitive theories.

Before presenting a final summary of the QCC, I believe there are additional criteria that a good theory of cognition must meet. I suspect that these will not be controversial, as they are a selection of insights that philosophers of science have generated in their considerations of what constitutes a good scientific theory in general (Popper, 1959; Quine and Ullian, 1970; Kitcher, 1993; Craver, 2007). I take it as too obvious to bother stating that each approach assumes that we are trying to construct a scientific theory of cognitive systems.⁸ Nevertheless, these criteria may play an important role in distinguishing good cognitive theories from bad.

Two of the most important considerations for good scientific theories are those of unity and simplicity. Good scientific theories are typically taken to be unified: the more sources of data, and the more scientific disciplines that they are consistent with, the better the theory. One of the reasons Einstein’s theory of relativity is to be preferred over Newton’s theories of motion is that the former is consistent with more of our observations. I have called this criterion “triangulation” to emphasize the idea that a good theory contacts many distinct sources of data, in a consistent and unified way. In addition, good theories tend to simplicity. I have called this criterion “compactness” to emphasize that good theories can be stated compactly and without ad hoc additions. The reason that the heliocentric theory of our solar system is preferred over a geocentric one is that, in the latter, we need to specify not only the circular paths of the planets, but also the many infamous “epicycles” of each planet in order to explain their motions. That is, there is a degree of arbitrariness that enters the geocentric theory, making it less convincing. In contrast, the heliocentric theory needs to specify one simple ellipse for each planet. Thus, the same data is explained by the heliocentric theory in a more compact way.

Though this discussion has been brief, I believe that these considerations provide a reasonably clear indication of several criteria that researchers in the behav-

⁸Yes, I just stated it.

Table 1.1: Quintessential Cognitive Criteria (QCC) for theories of cognition.

1. Representational structure
a. Systematicity
b. Compositionality
c. Productivity (the problem of variables)
d. The massive binding problem (the problem of two)
2. Performance concerns
a. Syntactic generalization
b. Robustness
c. Adaptability
d. Memory
e. Scalability
3. Scientific merit
a. Triangulation (Contact with more sources of data)
b. Compactness

ioral sciences would agree can be used to evaluate cognitive theories – regardless of their own theoretical predispositions. As a result, table 1.1 summarizes the QCC we can, at least on the face of it, extract from this discussion. As a reminder, I do not expect the mere identification of these criteria to be convincing. A more detailed discussion of each is presented in chapter 8. It is, nevertheless, useful to keep the QCC in mind as we consider a new proposal.

1.4 The house of answers

If questions are the house of answers, then obviously we must state those questions in order to extract their contents. The QCC are concerned with *evaluating* a characterization of cognition, not *directing* the development of such a characterization. So in this section I identify four central questions that have been relatively persistent over the last 50 years. I take it that detailed answers to these questions will go a long way to providing a characterization of a cognitive system that addresses most, if not all, of the QCC. My answers are in chapters 3-6.

The questions are:

1. How are semantics captured in the system?
2. How is syntactic structure encoded and manipulated by the system?

3. How is information flexibly routed through the system in response to task demands?
4. How are memory and learning employed in the system?

A long-standing concern when constructing models of cognitive systems is how to characterize the relationship between the representations inside the system, and the objects in the external world that they purportedly represent. That is, how do we know what a representation means? Of course, for any system we construct, we can simply define what the meaning of particular representation is. Unfortunately, this is very difficult to do convincingly: consider trying to define a mapping between dogs-in-the-world and a state in your head that acts like our concept “dog”. The vast psychological literature on concepts points to the complexity of this kind of mapping. Most researchers in cognitive science are well aware of, and dread, addressing this problem, which is often called the “symbol grounding problem” (Harnad, 1990). Nevertheless, any implemented model of a cognitive system must make some assumptions about how the representations in that system get their meaning. Consequently, answering this first question will force anyone wishing to characterize cognition to, at the very least, state their assumptions about how internal states are related to external ones. Whatever the story, it will have to plausibly apply not only to the models we construct, but also the natural systems we are attempting to explain.

The second question addresses what I have already noted is identified nearly universally as a hallmark of cognitive systems: the manipulation of structured representations. Whether we think *internal* representations themselves are structured (or, for that matter, if they even exist), we must face the undeniable ubiquity of behavior involving the manipulation of language-like representations. Consequently, any characterization of cognition will have to tell a story about how syntactic structure is encoded and manipulated by a cognitive system. Answering this question will unavoidably address at least the first five criteria in the QCC.

Another broadly admired feature of cognitive systems is their incredible, rapid adaptability. People put into a new situation can quickly survey their surroundings, identify problems, and formulate plans for solving those problems – often within the space of a few minutes. Performing each of these steps demands coordinating the flow of huge amounts of information through the system. Even seemingly mundane changes in our environment, such as switching from a pencil to a marker while writing, pose difficult control problems. Such a simple switch can alter the weight of the hand, change what are valid configurations of the fingers, and modify the expected visual, auditory, and proprioceptive feedback signals. All

of these pieces of information can influence what is the best motor control plan, and so must be taken into account when constructing such a plan. Because the sources of such information can remain the same (e.g. visual information comes from the visual system), while the destination of the information may change (e.g., from arm control to finger control), a means of routing that information must be in place. More cognitively speaking, if I simply tell you that the most relevant information for performing a task is going to switch from something you are hearing to something you will be seeing, you can instantly reconfigure the information flow through your brain to take advantage of that knowledge. Somehow, you are re-routing the information you use for planning to come from the visual system instead of the auditory system. We do this effortlessly, we do this quickly, and we do this constantly.

The fourth question focuses our attention on another important source of cognitive flexibility: our ability to use past information to improve our performance on a future task. The time scale over which information might be relevant to a task ranges from seconds to many years. Consequently, it is not surprising that the brain has developed mechanisms to store information that also range over these time scales. Memory and learning are behavioral descriptions of the impressive abilities of these mechanisms. Considerations of memory and learning directly address several of the performance concerns identified in the QCC. Consequently, any characterization of a cognitive system has to provide some explanation for how relevant information is propagated through time, and how the system can adapt to its past experience.

So, I use these four questions to direct my description of a cognitive architecture. Unsurprisingly, there are already many answers to these questions. Most often there are multiple, conflicting answers that have resulted in a variety of contemporary debates regarding cognitive function. For example, with respect to syntax, there is a contemporary debate about whether cognitive representations are best understood as symbolic or subsymbolic. It might be natural, then, to try to characterize a new architecture by describing which side of such debates it most directly supports. However, Paul Thagard (2012) has recently suggested that the architecture I describe – the Semantic Pointer Architecture (SPA) – can be taken to resolve many of these debates by providing a synthesis of the opposing views. For example, instead of thinking that an approach must be symbolic or subsymbolic, or computational or dynamical, or psychological or neural, the SPA is all of these. The seven debates he highlights are shown in table 1.2.

Of course, much work must be done to make a convincing case that the SPA successfully provides a convincing resolution to some of these long-standing de-

Table 1.2: Seven contemporary debates resolved through a synthesis of opposing views by the Semantic Pointer Architecture.

	<i>Traditional View</i>	<i>Alternative View</i>
1	computational	dynamical
2	symbolic	subsymbolic
3	rule-governed	statistical
4	psychological	neural
5	abstract	grounded
6	disembodied	embodied
7	reflective	enactive

bates. As a result, I do not return to this claim until chapter 9. Nevertheless, it is helpful to make clear early on that I do not take the approach that I describe to be championing a particular cognitive approach. Instead, it more often borrows from, and hopefully unifies, what have often been depicted as competing ways of understanding cognitive function.

The general method I adopt for attempting to provide this unification consists in demonstrating how a single, underlying implementation can be concurrently described as both, say, computational and dynamical, or symbolic and subsymbolic, or psychological and neural, etc. This method essentially stems from the advice of Richard Feynman, the famous physicist. Shortly before he died, Feynman wrote a few of his last, perhaps most dear, thoughts on the blackboard in his office. After his death, someone had the foresight to take a photograph of his blackboard for posterity (see <http://abbynuss1.tripod.com/id32.htm>). In large letters, in the top left corner, he wrote:

What I cannot create I do not understand.

I would like to propose to adopt this as a motto for how best to answer our questions about cognition. It suggests that *creating* a cognitive system would provide one of the most convincing demonstrations that we truly understand such a system. From such a foundation, we would be in an excellent position to evaluate competing descriptions, and ultimately evaluate answers to our questions about cognition. I will not argue for this point in great depth (though others have (Dretske, 1994)). But this, in a nutshell, is the goal I'm striving towards. In the next chapter, I begin with a description of neural mechanisms, so we can work our way towards an understanding of biological cognition.

Of course, in this case, as in the case of many other complex physical systems, “creating” the system amounts to creating *simulations* of the underlying mechanisms in detail. As a result, I also introduce a tool for creating neural simulations that embodies the principles described throughout the book. This tool is a graphical simulation package called Nengo (<http://www.nengo.ca/>), that can be freely downloaded and distributed. At the end of each chapter, I provide a tutorial for creating and testing a simulation that relates to a central idea introduced in that chapter. It is possible to understand all aspects of this book without using Nengo, and hence skipping the tutorials. But, I believe a much deeper understanding is possible if you “play” with simulations to see how complex representations can map to neural spike patterns, and how sophisticated transformations of the information in neural signals can be accomplished by realistic neuronal networks. After all, knowing how to build something and building it yourself are two very different things.

1.5 Nengo: An introduction

Nengo (Neural ENGineering Objects) is a graphical neural simulation environment developed over the last several years by my research group. In this tutorial, I describe how to install and use Nengo, and guide you through simulating a single neuron. The simulation you will build is shown in Figure 1.3. Anything you must do is placed on a single bulleted line. Surrounding text describes what you are doing in more detail. All simulations can also simply be loaded through the *File* menu. They are in the “Building a Brain” subdirectory of the standard Nengo installation.

A quick visual introduction to the simulator is available through the videos posted at <http://compneuro.uwaterloo.ca/cnrglab/videos.html> (???web page not yet accessible).

Installation

Nengo works on Mac, Windows, and Linux machines and can be installed by downloading the software from <http://nengo.ca/>.

- To install, unzip the downloaded file where you want to run the application from.

- In the unzipped directory, double-click ‘Nengo’ to run the program on Mac and Linux, or double-click ‘Nengo.bat’ to run the program on Windows.

An empty Nengo world appears. This is the main interface for graphical model construction.

Building a model

The first model we will build is very simple: just a single neuron.

- A column of icons should occupy the left side of the screen. This is the *Template Bar*. If the template bar is not displayed, click the icon showing a pair of scissors over a triangle which is located at the top-right corner of the screen.
- From the template bar, click the icon above the label ‘Network’ and drag the network into the workspace. Alternatively, right-click anywhere on the background and choose *New Network*.
- Set the *Name* of the network to ‘A single neuron’ and click *OK*.

All models in Nengo are built inside ‘networks’. Inside a network you can put more networks, and you can connect networks to each other. You can also put other objects inside of networks, such as neural populations (which have individual neurons inside of them). For this model, you first need to create a neural population.

- Click the *Ensemble* icon in the template bar and drag this object into the network you created. In the dialog box that appears, you can set *Name* to ‘neuron’, *Number of nodes* to 1, *Dimensions* to 1, *Node Factory* to ‘LIF Neuron’, and *Radius* to 1. Click *OK*.

A single neuron is now created. This leaky integrate-and-fire (LIF) neuron is a simple, standard model of a spiking single neuron. It resides inside a neural ‘population’, even though there is only one neuron. To delete anything in Nengo, right-click on it and select ‘Remove model’. To readjust your view, you can zoom using the scroll wheel and drag the background to shift within the plane.

- To see the neuron you created double-click on the ‘population’ you just created.

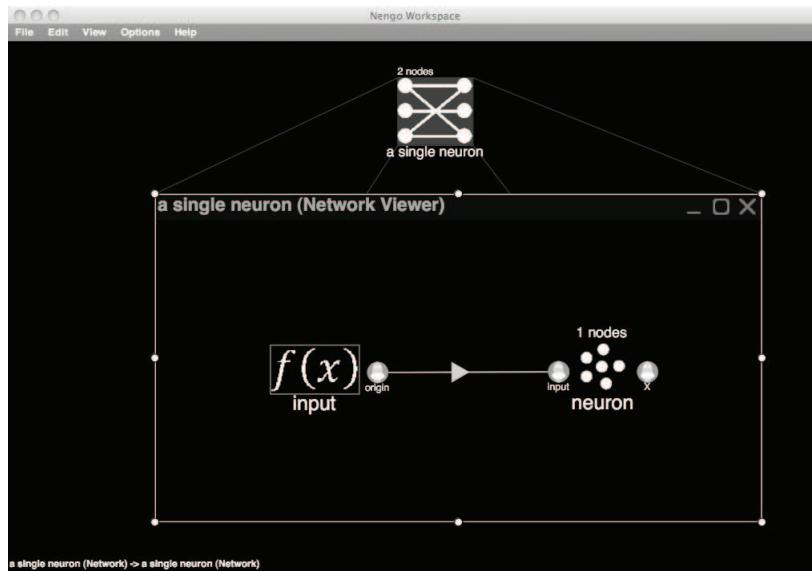


Figure 1.3: A single neuron Nengo model. This is a screen capture from Nengo that shows the finished model described in this section.

This shows a single neuron called ‘node0’. To get details about this neuron, you can right-click it and select *Configure* and look through the parameters, though they won’t mean much without a good understanding of single cell models. In order to simulate input to this neuron, you need to add another object to the model that generates that input.

- Close the *NEFEnsemble Viewer* (with the single node in it) by clicking the ‘X’ in the upper right of that window.
- Drag the *Input* component from the template bar into the *Network Viewer* window. Set *Name* to ‘input’, and *Output Dimensions* to 1. Click *Set Functions*. In the drop down select *Constant Function*. Click *Set*. Set *Value* to 0.5. Click *OK* for all the open dialogs.

A network item that provides a constant current to inject into the neuron has now been created. It has a single output labeled ‘origin’. To put the current into the neuron, you need to create a ‘termination’ (i.e., input) on the neuron.

- Drag a *Termination* component from the template bar onto the ‘neuron’ population. Set *Name* to ‘input’, *Weights Input Dim* to 1, and *tauPSC* to 0.02. Click *Set Weights*, double-click the value and set it to 1. Click *OK*.

A new element will appear on the left side of the population. This is a *decoded termination* which takes an input and injects it into the population. The *tauPSC* specifies the ‘synaptic time constant’, which I will discuss in the next chapter. It is measured in seconds, so 0.02 is equivalent to 20 milliseconds.

- Click and drag the ‘origin’ on the input function you created to the ‘input’ on the neuron population you created.

Congratulations, you’ve constructed your first neural model. Your screen should look like Figure 1.3.

Running a model

You want to make sure your model works, so you need to use to the parts of Nengo that let you ‘run’ the model. There are two ways to run models in Nengo. The first is a ‘non-interactive mode’, which lets you put ‘probes’ into parts of the network. You then run the network for some length of time, and those probes gather data (such as when neurons are active, what their internal currents and voltages are, etc.). You can then view that information later using the built in data viewer or with another program such as Matlab®. For future reference, to access this method of running, you can right-click on the background of the network and use the ‘Run’ command under the ‘Simulator’ heading.

The other way to run models is to start ‘interactive plots’, which is more hands-on, so I will use it in the examples in this book. An example of the output generated by interactive plots for this model is shown in Figure 1.4.

- Click on the *Interactive Plots* icon (the double sine wave in the top right corner).

This pulls up a new window with a simplified version of the model you made. It shows the ‘input’ and ‘neuron’ network elements that you created earlier. They are the elements that you will interact with while running the simulation. This window also has a ‘play’ button and other controls that let you run your model. First, you need to tell the viewer which information generated by the model you would like to see.

- Right-click on the ‘neuron’ population in the interactive plots window and select *spike raster*.

A spike raster shows the times that the neurons in the population fire an action potential spike (signifying a rapid change in its membrane voltage), which is largely how neurons communicate with one another. You can drag the background, and any of the objects around within this view to arrange them in a useful manner.

- Right-click on the ‘neuron’ population and select *value*.

This graph will show what effects this neuron will have on the current going into a neuron that receives its output spikes shown in the raster. Each small angular pulse can be thought of as a postsynaptic current or PSC that would be induced in the receiving cell.

- Right-click ‘input’ and select *control*.
- Right-click on ‘input’ and select *value*.

This shows a controller for manipulating the input, and the value of your control input over time.

- Click the play arrow in the bottom right.

The simulation is now running. Because the neuron that you generated was randomly chosen, it may or may not be active with the given input. Either way, you should grab the slider control and move it up and down to see the effects of increasing or decreasing input. Your neuron will either fire faster with more input (an ‘on’ neuron) or it will fire faster with less input (an ‘off’ neuron). Figure 1.4 shows an on neuron with a fairly high firing threshold (just under 0.69 units of input). All neurons have an input threshold below which (or above which for off neurons) they will not fire any spikes.

You can test the effects of the input and see if you have an on or off neuron, and where the threshold is. You can change the neuron by pausing the simulation, and returning to the main Nengo window. To randomly pick another neuron, do the following in the main window:

- Right-click on the ‘neuron’ population and select *Configure*.
- Click on the gray rightward pointing arrow that is beside *i neurons (int)*. It will point down and *i 1* will appear.
- Double-click on the 1 (it will highlight with blue), and hit *Enter*. Click *Done*.

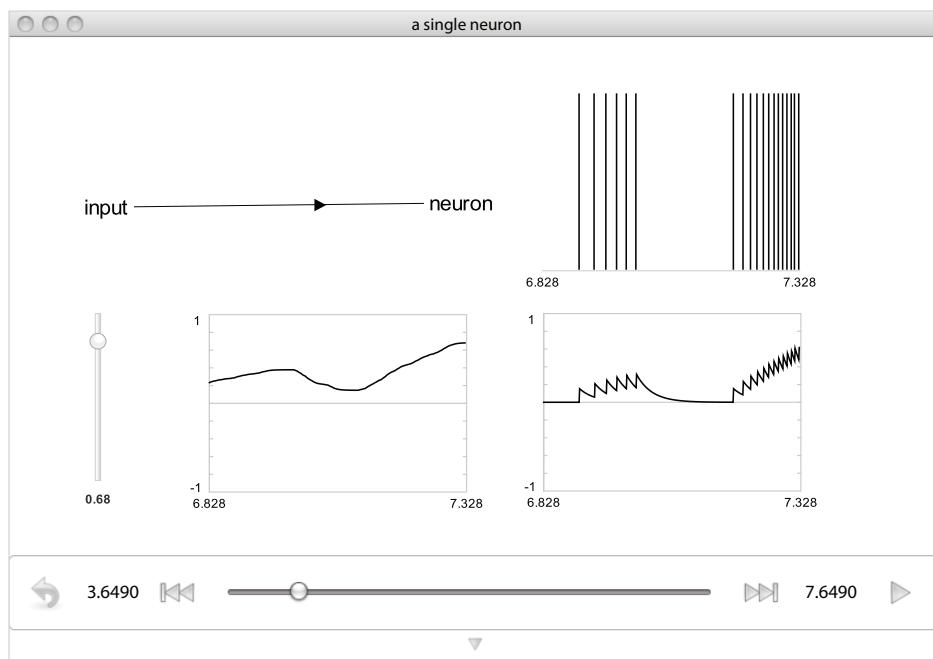


Figure 1.4: Running the single neuron model. This is a screen capture from the interactive plot mode of Nengo running a single neuron.

You can now return to the interactive plots and run your new neuron by hitting play. Different neurons have different firing thresholds. As well, some are also more responsive to the input than others. Such neurons are said to have higher sensitivity, or ‘gain’. You can also try variations on this tutorial by using different neuron models. Simply create another population with a single neuron and choose something other than ‘LIF neuron’ from the drop down menu. There are a wide variety of neuron parameters that can be manipulated. In addition, it is possible to add your own neuron models to the simulator, inject background noise, and have non-spiking neurons. These topics are beyond the scope of this tutorials, but relevant information can be found on our website <http://compneuro.uwaterloo.ca/cnrglab/>.

Congratulations, you have now built and run your first biologically plausible neural simulation using Nengo. You can save and reload these simulations using the *File* menu.

Part I

How to build a brain

Chapter 2

An introduction to brain building

Before turning to my main purpose of answer the four questions regarding semantics, syntax, control, and memory and learning, I need to provide an introduction to the main method I will be relying in providing answers to these questions. As I argued in the last chapter, I believe the best answers to such questions will be biologically-based. In the sections that follow, I provide an overview of the relevant biology, and introduce the Neural Engineering Framework (NEF), which provides basic methods for constructing large-scale neurally realistic models of the brain. My purpose here is to lay a foundation for what is to come: a neural architecture for biological cognition. Additional details on the NEF can be found in Eliasmith and Anderson (2003).

2.1 Brain parts

Brains are absolutely fantastic devices. For one, they are incredibly efficient. Brains consume only about 20 W of power – the equivalent of a compact fluorescent light bulb. To put this power efficiency in some perspective, one of the world’s most powerful supercomputers “roadrunner” at Los Alamos labs in the United States, which, as far as we know, is unable to match the computational power of the mammalian brain, consumes 2.35 MW (about 100,000 times more). Brains are also relatively small compared to the size of our bodies. A typical brain weighs between 1 and 2 kg and comprises only 2% of our body weight. Nevertheless, brains account for about 25% of the energy used by the body. This is

especially surprising when you consider the serious energy demands of muscles, which must do actual physical work. Presumably, since the brain is such a power hog, it is doing something important for our survival or it would not have been preserved by evolution. Presumably, that “something important” is somewhat obvious: brains control the four Fs (feeding, fleeing, fighting, and, yes, reproduction); brains provide animals with behavioral flexibility that is unmatched by our most sophisticated machines; and brains are constantly adapting to the uncertain, noisy, and rapidly changing world in which they find themselves embedded.

We can think of this incredibly efficient device as something like a soft pillow crammed inside our skulls. While the texture of brains has often been compared to that of a thick pudding, it is more accurate to think of the brain as being a large sheet, equivalent in size to about four sheets of writing paper, and about 3 mm thick. In almost all animals, the sheet has six distinct layers which are composed of three main elements: 1) the cell bodies of neurons; 2) the very long thin processes used for communication; and 3) glial cells, which are a very prevalent but poorly understood companion to neurons. In each square millimeter of human cortex there are crammed about 170,000 neurons. So, there are about 25 billion neurons in human cortex. Overall, however, there are approximately 100 billion neurons in the human brain. The additional neurons come from “subcortical” areas, which include cerebellum, basal ganglia, thalamus, and the brainstem, among others. To get a perspective on the special nature of human brains, it is worth noting that monkey brains are approximately the size of one sheet of paper, and rats have brains the size of a Post-it note.

In general, it is believed that what provides brains with their impressive computational abilities is the organization of the connections among individual neurons. These connections allow cells to collect, process, and transmit information. In fact, neurons are specialized precisely for communication. In most respects, neurons are exactly like other cells in our bodies: they have a cell membrane, a nucleus, and they have similar metabolic processes. What makes neurons stand out under a microscope, are the many branching processes which project outwards from the somewhat bulbous, main cell body. These processes are there to enable short and long distance communication with other neural cells. This is a hint: if we want to understand how brains work, we need to have some sense of how neurons communicate in order to compute.

Figure 2.1 outlines the main components and activities underlying cellular communication. The cellular processes that carry information to the cell body are called dendrites. The dendrites carry signals, in the form of an ionic current to the main cell body. If sufficient current enters the cell body at a given point in

time, a series of nonlinear events will be triggered that result in an action potential, or voltage “spike,” that will proceed down the output process of the neuron, which is call an axon. Neural spikes are very brief events, lasting for only a few milliseconds (see figure 2.2), which travel in a wave-like fashion down the axon until they reach the end of the axon, called the bouton, where they cause the release of tiny packets of chemicals called neurotransmitters. Axons end near the dendrites of subsequent neurons. Hence, neurotransmitters are released into the small space between axons and dendrites, which is called the synaptic cleft. The neurotransmitters very quickly cross the cleft and bind to special proteins in the cell membrane of the next neuron. This binding causes small gates, or channels, in the dendrite of the next neuron to open, which allows charged ions to flow into the dendrite. These ions result in a current signal in the receiving dendrite, called the postsynaptic current (PSC; see figure 2.2), which flows to the cell body of a subsequent neurons, and the process continues as it began.

A slightly simpler description of this process, but one which retains the central relevant features, is as follows:

1. Signals flow down the dendrites of a neuron into its cell body.
2. If the overall input to the cell body from the dendrites crosses a threshold, the neuron generates a stereotypical spike that travels down the axon.
3. When the spike gets to the end of the axon, it causes chemicals to be released that travel to the connected dendrite and, like a key into a lock, cause the opening of channels that produce a postsynaptic current (PSC) in the receiving dendrite.
4. This current is the signal in step 1., that then flows to the cell body of the next neuron.

As simple as this story first sounds, it is made much more complex in real brains by a number of factors. First, neurons are not all the same. There are hundreds of different kinds of neurons that have been identified in mammalian brains. The neurons can range in size from 10^{-4} to 5 m in length. The number of inputs to a cell can range from about 500 or fewer, to well over 200,000. The number of outputs, that is, branches of a single axon, cover a similar range. On average, cortical cells have about 10,000 inputs and 10,000 outputs. Given all of these connections, it is not surprising to learn that there are approximately 72 km of fiber in the human brain. Finally, there are hundreds of different kinds of

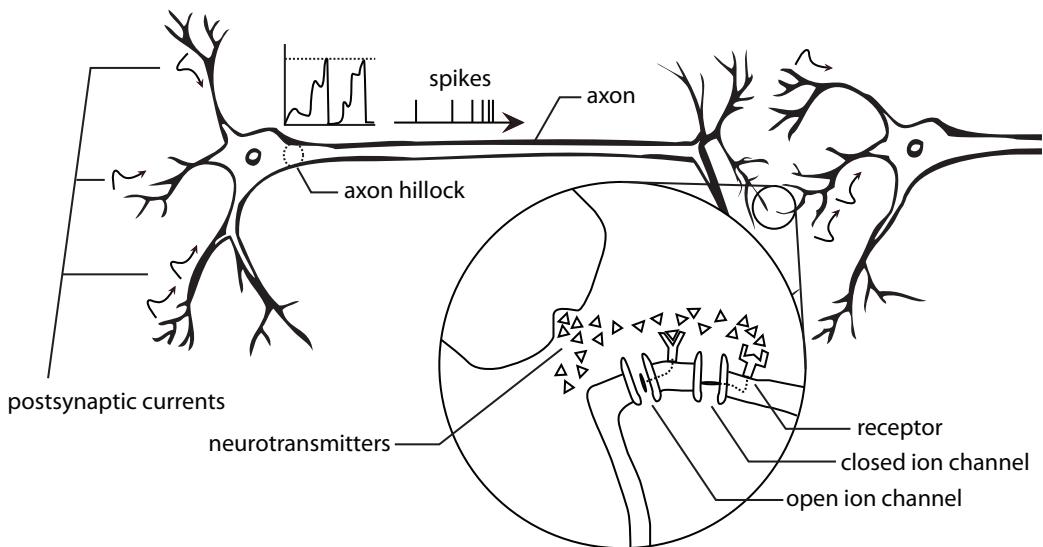


Figure 2.1: Cellular communication. This figure diagrams the main components and kinds of activities of typical neural cells during communication. The flow of information begins at the left side of the image, with postsynaptic currents (PSCs) travelling along dendrites towards the cell body of the neuron. At the axon hillock, these currents result in a voltage spike being generated if they cause the somatic voltage to go above the neuron's threshold. The spikes travel along the axon and cause the release of neurotransmitters at the end of the axon. The neurotransmitters bind to matching receptors on the dendrites of connected neurons, causing ion channels to open. Following the opening of ion channels, postsynaptic currents are induced in the receiving neuron and the process continues.

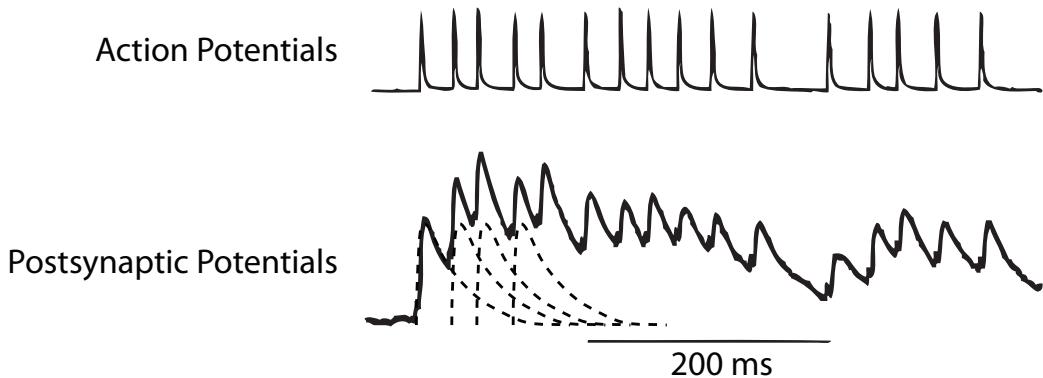


Figure 2.2: Electrical activity in two connected neurons. The top trace shows a series of neural spikes recorded intracellularly from a single rat neuron, that projects this signal to a receiving neuron. The bottom trace shows a series of postsynaptic potentials (PSPs, caused by PSCs) generated at a single synapse in the receiving neuron in response to the input action potentials. The dotted lines indicate the single PSPs that would result from a spike in isolation. The PSP trace is approximately a sum of the individual PSPs. (Adapted from Holmgren et al. (2003), figure 4, with permission).

neurotransmitters and many different kinds of receptors. Different combinations of neural transmitters and receptors, can cause different kinds of currents to flow in the dendrite. As a result, a single spike transmitted down an axon can be received by many different neurons, and can have different kinds of effect on each neuron depending on the mediating neurotransmitter and receptors.

The variability found in real biological networks makes for an extremely complex system, one which, at first glance, seems designed to frustrate any analysis. Typically, in mathematics, homogeneous systems (those that look similar no matter where you are in the system) are much easier to understand. The brain, in contrast, is clearly highly heterogeneous: there are hundreds of kinds of neurons, many with different kinds of intrinsic dynamical properties. Even neurons of the same kind often respond differently, i.e., generate different patterns of spikes to exactly the same current injected into their soma. And, even neurons that share response properties and physiological type, can still differ in the number and kinds of channels in their dendrites, meaning that the same pattern of spikes coming from preceding neurons to two otherwise similar neurons will generate different responses. For example, figure 2.3 shows a variety of responses to directly in-

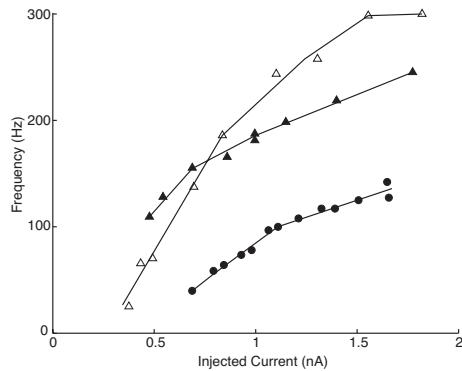


Figure 2.3: Response functions of regular-spiking neurons found in the mammalian neocortex. These curves demonstrate that the same kinds of neurons can have very different responses even for the same current injected directly into the soma (adapted from McCormick et al. (1985) with permission).

jected current from the most common kinds of cells found in cortex.

Experimental neuroscientists who record the activity of single neurons in response to perceptual stimuli shown to an animal, will tell you that no two neurons seem to respond in the same manner. Often, the responses of a given cell is captured in terms of its “tuning curve”. Sometimes, these tuning curves will look much like the response functions shown in figure 2.3 (with a stimulus parameter instead of current on the x -axis). However, tuning curves can have many different shapes. An example tuning curve from a cell in primary visual cortex is shown in figure 2.4. In general, a tuning curve is a graph that shows the frequency of spiking of a neuron in response to a given input stimuli. In this figure, we can see that as the orientation of a presented bar changes, the neuron responds more or less strongly. Its peak response happens at about 0 degrees. Nevertheless, there is some information about the orientation of the stimulus available from this activity whenever the neuron responds.

This tuning curve, while typical of cells in primary visual cortex, includes an additional implicit assumption. That is, this is not the response of the cell to any oriented bar at any position in the visual field. Rather, it is a response to an oriented bar in what is called the “receptive field” of the cell. Since this is a visual cell, the receptive field indicates which part of possible visual input space the neuron responds to. So, to completely capture the “tuning” of the cell to visual stimuli, we ideally want to combine the receptive field and tuning curve. In the remainder of this book, I will use the notion of tuning curve in this more

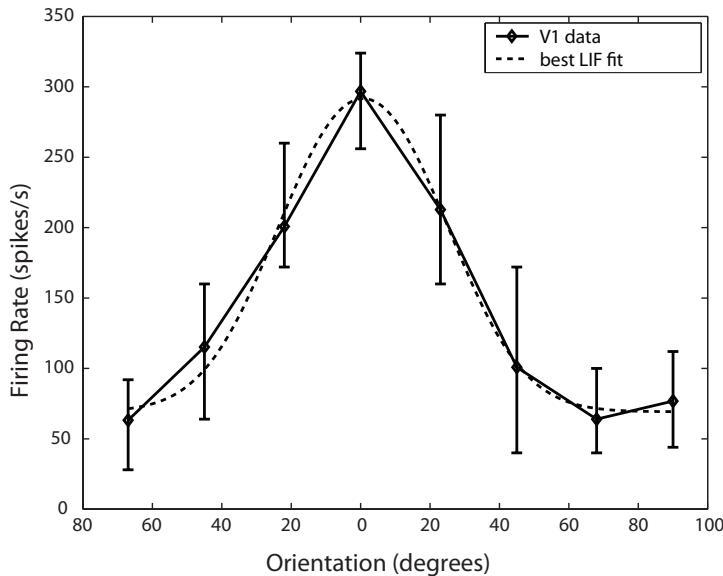


Figure 2.4: Example tuning curve for a cell from a macaque monkey in primary visual cortex (data provided by Dan Marcus). The x -axis indicates the angle of rotation of a bar, and the y -axis indicates the firing rate of the neuron when shown that stimulus. The graph is a result of many trials at each rotation, and hence has error bars that indicate the standard deviation of the response. The dashed line indicates the best fit of a leaky integrate-and-fire (LIF) neuron model to the data (adapted from Eliasmith & Anderson (2003), with permission).

general sense. That is, what I subsequently define as a tuning curve often includes information about the receptive field of the cell in question. Returning to the issue of heterogeneity, the reason that experimental neuroscientists would suggest that no two cells are the same, is partly because their tuning curves (coupled with their receptive fields) seem to never perfectly overlap. As a result, while neurons near to the one shown in figure 2.4 may share similar tuning properties, the peak, width, and roll-off of the graph will be somewhat different.

We can begin to get a handle on the heterogeneity observed in neural systems by noticing that there are two main sources for the variability: intrinsic and extrinsic. Intrinsic heterogeneity, such as different responses to the same injected current, will have to be captured by variability in the models of individual cells. Extrinsic heterogeneity, such as the variability of tuning curves, can be understood more as a consequence of where a particular cell sits in the network. That is, the

reason neighboring cells have different tuning curves is not merely because of intrinsic heterogeneity, but also because they are receiving slightly different inputs than their neighbors, or treating the same inputs slightly differently. So, even if all of the cells were intrinsically homogeneous, their tuning curves might look very different depending on the processing of the cells before them. This difference between extrinsic and intrinsic heterogeneity will prove important in understanding the sources of different kinds of dynamics observed in real neural networks.

Of course, this distinction does not in any way mitigate the fact of heterogeneity itself. We must still overcome the observed diversity of neural systems, which has traditionally been seen as a barrier to theoretical analysis. In the next sections we will see that despite this complexity, it is possible to suggest and quantify underlying principles which do a good job of describing the functional properties of neural networks that display broad heterogeneity. In fact, we can come to understand why this massive heterogeneity that we observe might provide for more robust computations than those of a system whose individual components were identical (see section 2.3.1).

2.2 Theoretical neuroscience

In recent years, most of the behavioral sciences have seen a heavy influx of influence from the neurosciences. Most obviously, the psychology of a mere three decades ago was almost bereft of brain-related talk. Now, however, one of the fastest growing areas of psychology is cognitive neuroscience, whose practitioners regularly talk of neuroanatomy and neurophysiology, and employ various brain imaging and measurement techniques, like fMRI and EEG. The same trend can be found in many other disciplines as well, including linguistics, economics, and philosophy (often called “experimental philosophy”, or “X-phi”).

While the growing influence of neuroscience is undeniable, it would be a mistake to think that the field of neuroscience itself is monolithic. Quite the contrary: the tens of thousands of posters and presentations at the Society for Neuroscience meeting held every year are divided in to many sections, such as molecular neuroscience, cellular neuroscience, systems neuroscience, neuroanatomy, developmental neuroscience, behavioral neuroscience, cognitive neuroscience, and so on. Often, the researchers in one area are not able to understand or communicate effectively with those from another, despite the fact that ultimately, everyone in the building wants to understand how the brain works.

This, of course, is not a problem unique to neuroscience. Many other areas of

biology suffer the same difficulties, as do other sciences such as geology, chemistry, and even physics. However, one main advantage that, for example, physics has in mitigating these challenges is a widely shared technical vocabulary for describing both problems and solutions in the field. The development and application of these quantitative tools have helped physics develop rapidly, leading us to exciting new discoveries, as well as deep, challenging problems. The subfield of physics most centrally concerned with the development of such tools is theoretical physics.

Interestingly, an analogous subfield has been developing in neuroscience over the last few decades as well, and it is often appropriately called “theoretical neuroscience” (though perhaps equally as often called “computational neuroscience”). In fact, the analogy between theoretical neuroscience and theoretical physics is both strong, and useful for understanding its importance to neuroscience. For instance, both are centrally interested in quantifying the phenomena under study. This does not mean merely statistically quantifying the data generated by the phenomena, but rather coming up with quantitative descriptions of the deterministic regularities and mechanisms giving rise to that data. Take, for instance, one of the greatest advances in theoretical physics, the development of Newton’s three laws of motion. The second, perhaps most famous, law is that “The alteration of motion is ever proportional to the motive force impressed; and is made in the direction of the right line in which that force is impressed” Newton (1729, p. 19). In short form: $F = ma$. The purpose of this statement is to make a clear, straightforward hypothesis about motion. I describe similar principles in the sections 2.3.1-2.3.3 for neural representation, computation, and dynamics.

A second analogy between these two subfields is that both are interested in summarizing an enormous amount of data. Newton’s second law is intended to apply to all forms of motion, be it rectilinear, circular, or what have you. Measuring all such forms of motion, and describing the results statistically would not be nearly as concise. Similarly, theoretical neuroscientists are attempting to understand the basic principles behind neural function. Often, they would like their mathematical descriptions to be as general as possible, although there is some debate regarding whether or not the kind of unification being striven for in physics should be a goal for theoretical neuroscience.

A third crucial analogy between theoretical physics and theoretical neuroscience is that the disciplines are speculative. Most such quantitative descriptions of general mechanisms go beyond the available data. As such, they almost always suggest more structure than the data warrants, and hence more experiments to perform. Looking again to Newton’s second law, it is clear that Newton intended it

to be true for all velocities. However, special relativity has subsequently demonstrated that the law is measurably violated for large velocities. With speculation comes risk. Especially for a young field like theoretical neuroscience, the risk of being wrong is high. But, the risk of becoming lost in the complexity of the data without such speculation is much higher. The real fruit of a field like theoretical neuroscience – fruit realized by theoretical physics – is (hopefully) the identification of a set of principles that lets massive amounts of data be summarized, thereby suggesting new questions and opening lines of communication between different sub-disciplines.

Of course, there are crucial disanalogies between these fields as well. For instance, we can think of physics as being interested in questions of *what there is*, while neuroscience is interested in questions of *who we are*. As well, neuroscience is a much younger discipline than physics, and so the methods for making measurements of the system of interest are still rapidly developing.

Nevertheless, the similarities can help us understand why theoretical neuroscience is important for the development of neuroscience. Like theoretical physics, theoretical neuroscience can help in at least two crucial ways. Specifically, it should:

1. Quantify, and hence make more precise and testable, hypotheses about the functioning of neural systems;
2. Summarize large amounts of experimental data, and hence serve to unify the many sources of data from different “neuro-” and behavioral sub-disciplines.

The first of these stem from the commitment to using mathematics to describe principles of neural function. The second is crucial for trying to deal with the unavoidable complexities of neural systems. This is a challenge not faced to the same degree by many physicists.¹ Characterizing a system of billions of parts as if each is identical, and as if the connections between all of them are approximately the same can lead to very accurate characterizations of physical systems (e.g. the ideal gas law). The same is not true of neural systems. Large neural systems where all of the parts are the same, and interact in the same way simply do not exist.

Thus, the links between the “lowest” and “highest” levels of characterizing neural systems are complex and unavoidable. It is perhaps in this kind of circumstance that quantification of the system of interest plays its most crucial role. If

¹This observation should make it obvious that I am in no way suggesting the behavioral sciences should become physics, a mistake which has been made in the past Carnap (1931).

we can state our hypotheses about neural function at a “high” level, and quantify the relationship between levels, then our high-level hypothesis will connect to low-level details. In fact, ideally, a hypothesis at any level should contact data at all other levels. It is precisely this kind of unification of experimental data that is desperately needed in the behavioral sciences to support cross-disciplinary communication, and eventually a mature theory of biological cognition.

This ideal role for theoretical neuroscience has not yet been realized. Perhaps this is because there has historically been more of a focus on “low” levels of neural systems (i.e. single cells or small networks). This is perfectly understandable in light of the complexity of the system being tackled. Nevertheless, I believe we are now in a position to begin to move past this state of affairs.

In the context of this book, I will adopt methods from theoretical neuroscience that I believe currently have the best potential to realize this ideal role. At the same time I hope the contents of this book will prod theoretical neuroscientists to consider expanding the viable areas of application of their methods to all of the behavioral sciences. In the next section I introduce a series of theoretical principles developed in the context of traditional theoretical neuroscience. In subsequent chapters I suggest a way of applying these same principles to large-scale, cognitive modeling. This should help to not only test, but also to refine such principles, and it will help make our cognitive models subject to data from all of the various disciplines of the behavioral sciences.

2.3 A framework for building a brain

In 2003, Charles H. Anderson and I wrote a book called *Neural Engineering*. In it, we presented and demonstrated a mathematical theory of how biological neural systems can implement a wide variety of dynamic functions. We now refer to this theory as the Neural Engineering Framework (NEF). As with most work in theoretical neuroscience, we focused on “low-level” systems, including parts of the brainstem involved in controlling stable eye position, parts of the inner ear and brainstem for controlling a vestibulo-ocular reflex (VOR), and spinal circuits in the lamprey for controlling swimming.

In more recent work, these methods have been used by us and others to propose novel models of a wider variety of neural systems, including the barn owl

auditory system (Fischer, 2005; Fischer et al., 2007), parts of the rodent navigation system (Conklin and Eliasmith, 2005), escape and swimming control in zebrafish (Kuo and Eliasmith, 2005), tactile working memory in monkeys (Singh and Eliasmith, 2006), and simple decision making in humans (Litt et al., 2008) and rats (ref Laubach???). We have also used these methods to better understand more general issues about neural function, such as how neural systems might perform time derivatives (Tripp and Eliasmith, 2010), how the variability of neural spike trains and the timing of individual spikes relates to information that can be extracted from spike patterns (Tripp and Eliasmith, 2007), how we can ensure that biological constraints such as Dale’s Principle – the principle that a given neuron typically has either excitatory or inhibitory effects but not both – are respected by neural models (Parisien et al., 2008). ???MacNeil learning paper ref, when accepted???

One reason the NEF has such broad application is because it does not make assumptions about what the brain actually does. Rather, it is a set of three principles that can help determine *how* the brain performs a given function. For this reason, John Miller once suggested that the NEF is a kind of “neural compiler”. If you have a guess about the high-level function of the system you are interested in, and you know (or assume) some information about how individual neurons respond to relevant input, the NEF provides a way of connecting populations of neurons together to realize that function. This, of course, is exactly what a compiler does for computer programming languages. The programmer specifies a program in a high-level language like Java. The Java compiler knows something about the low-level machine language implemented in a given chip, and it translates that high-level description into an appropriate low-level one.

Of course, things are not so clean in neurobiology. We do not have a perfect description of the machine language, and our high-level language may be able to define functions that we cannot actually implement in neurons. Consequently, building models with the NEF can be an iterative or bootstrapping process: first you gather data from the neural system and have a hypothesis about what it does; then you build a model using the NEF and see if it behaves like the real system; then, if it does not behave consistently with the data, you alter your hypothesis or perform experiments to figure out why the two are different. Sometimes a model will behave in ways that cannot be compared to data because the data does not exist. In these lucky cases, it is possible to make a prediction and perform an experiment. Of course this process is not unique to the NEF. Instead, it will be familiar to any modeller. What the NEF offers is a systematic method for performing these steps in the context of neurally realistic models.

It is worth emphasizing that, also like a compiler, the NEF does *not* specify what the system does. This specification is brought to the characterization of the system by the modeler. In short, the NEF is about *how* brains compute, not *what* they compute. The bulk of this book is about the “*what*”, but those considerations do not begin until chapter 3.

In the remainder of this section, I provide an outline of the three principles of the NEF. There are, of course, many more details (at least a book’s worth!). So, a few points are in order. First, the methods here are by no means the sole invention of our group. We have drawn heavily on other work in theoretical neuroscience. To keep this description as brief as possible, I refer the reader to other descriptions of the methods that better place the NEF in its broader context (Eliasmith and Anderson, 2003; Eliasmith, 2005b; Tripp and Eliasmith, 2007). Second, the original *Neural Engineering* book is a useful source for far more mathematical detail than I provide here. However, the framework has been evolving, so that book should be taken as a starting point. Finally, some mathematics can be useful for interested readers, but I have placed most of it in the appendices to emphasize a more intuitive grasp of the principles. This is at least partially because the Nengo neural simulator has been designed to handle the mathematical detail, allowing the modeller to focus effort on capturing the neural data, and the hypothesis she or he wishes to test.

The following three principles form the core of the NEF:²

1. Neural representations are defined by the combination of nonlinear encoding (exemplified by neuron tuning curves, and neural spiking) and weighted linear decoding (over populations of neurons and over time).
2. Transformations of neural representations are functions of the variables represented by neural populations. Transformations are determined using an alternately weighted linear decoding.
3. Neural dynamics are characterized by considering neural representations as state variables of dynamic systems. Thus, the dynamics of neurobiological systems can be analyzed using control (or dynamics systems) theory.

In addition to these main principles, the following addendum is taken to be important for analyzing neural systems:

- Neural systems are subject to significant amounts of noise. Therefore, any analysis of such systems must account for the effects of noise.

²For a quantitative statement of the three principles, see Eliasmith and Anderson (2003, pp. 230-231).

The ubiquity of noise in neural systems is well documented, be it from synaptic unreliability (Stevens and Wang, 1994; Zucker, 1973), variability in the amount of neurotransmitter in each vesicle (Burger et al., 1989), or jitter introduced by axons into the timing of neural spikes (Lass and Abeles, 1975). Consequently, there are limits on how much information can be passed by neurons: it seems that neurons tend to encode approximately 2-7 bits of information per spike (Bialek and Rieke, 1992; Rieke et al., 1997). Here I do not consider this addendum separately, though it is included in the formulation of each of the subsequent principles, and their implementation in Nengo.

Let me now describe each of the three principles in turn. To make the application of these principles concrete, I adopt the example of a “controlled integrator”. This neural circuit can be thought of as a simple memory circuit which can be loaded with a memory, hold the memory over time, and then erase the memory (see figure 2.8).

2.3.1 Representation

A central tenet of the NEF is that we can adapt the information theoretic account of *codes* to understanding representation in neural systems. Codes, in engineering, are defined in terms of a complimentary encoding and decoding procedure between two alphabets. Morse code, for example, is defined by the one-to-one relation between letters of the Roman alphabet, and the alphabet composed of a standard set of dashes and dots. The encoding procedure is the mapping from the Roman alphabet to the Morse code alphabet and the decoding procedure is its inverse (i.e., mapping dots and dashes to the Roman alphabet).

In order to characterize representation in a neural system, we can identify the relevant encoding and decoding procedures and their relevant alphabets. The encoding procedure is straightforward to identify: it is the mapping of stimuli into a series of neural spikes. Indeed, encoding is what neuroscientists typically talk about, and encompasses many of the mechanisms I have discussed in section 2.1. When we show a brain a stimulus, some neurons or other “fire” (i.e. generate action potentials). This pattern of firing is often depicted by neuroscientists as a “spike raster” (see the spike rasters in figure 2.7 for a simple example). Spike rasters show the time of occurrence of an action potential from a given neuron – in essence, the only response to the stimulus that it sends to other neurons. The raster

in figure 2.7 shows responses from just two neurons, while figure 2.7 shows the responses from thirty. Often, rasters will show the responses of the same neuron over many different trials to demonstrate the variability of the response to the same stimulus, although I will typically show the responses of many neurons on one trial.³ The precise nature of this encoding from stimulus to spikes has been explored in-depth via quantitative models (see Appendix A.1.1).

Unfortunately, neuroscientists often stop here in their characterization of representation, but this is insufficient. We also need to identify a decoding procedure, otherwise there is no way to determine the relevance of the purported encoding for “downstream” parts of the system. If no information about the stimulus can be extracted from the spikes of the encoding neurons, then it makes no sense to say that they represent the stimulus. Representations, at a minimum, must potentially be able to “stand-in” for the things they represent. As a result, characterizing the decoding of neural spikes into the physical variable they represent is as important as characterizing the process of encoding physical variables into neural spikes.

Quite surprisingly, despite typically nonlinear encoding (i.e., mapping a continuously varying parameter like stimulus intensity into a series of discontinuous spikes), a good linear decoding can be found.⁴ To say that the decoding is “linear” means that the responses of neurons in the population are weighted by a constant (the decoder) and summed up to give the decoding. There are several established methods for determining appropriate linear decoders given the neural populations that respond to certain stimuli (see Appendix A.1.2 for the one I will use in this book).

There are two aspects to the decoding of the neural response that must be considered. These are what I call the “population” and “temporal” aspects of decoding. The “population” decoding accounts for the fact that a single stimulus variable is typically encoded by many different (i.e. a population of) neurons. The “temporal” decoding accounts for the fact that neurons respond *in time*, to a typically *changing* stimulus. Ultimately, these two aspects determine one combined decoding. However, it is conceptually clearer to consider them one at a time.

As depicted in figure 2.5, population decoders are determined by finding the weighting of each neuron’s tuning curve, so that their sum represents the input

³Because we can control the trial-to-trial variability of our single neuron models, we often keep it low to be able to better see the typical response. Nevertheless, variability is often central to characterizing neural responses, and it can be captured effectively in Nengo using the available noise parameters (see e.g., ref???laubach paper). And, as mentioned earlier, variability and noise are central to the formulation of the NEF principles.

⁴This is nicely demonstrated for pairs of neurons in (Rieke et al., 1997, pp. 76-87).

signal over some range. That is, each neuron is weighted by how useful it is for carrying information about the stimulus in the context of the whole population. If it's very useful, it has a high weight, if not it has a lower weight. Finding the exact decoding weights in the NEF is accomplished by solving an optimization problem to minimize the representational error (see Appendix A.1.2). To be clear, each neuron's decoder is equal to the value of its weight, and it is the sum of the weighted responses that gives the population decoding.

As can be seen in figure 2.5, as more neurons are added to the population, the quality of the representation improves. This is because the neuron responses are nonlinear, but the ideal representation is linear (specifically, the estimate of the input should exactly equal the input). Once the decoders have been found, they remain fixed for any input value. So again, it is clear from this figure that different inputs will be encoded with different accuracies (as can be inferred from the “ripples” in the thick black lines of figure 2.5). Typically, the input will be changing over time, so we must also determine how to decode over time.

Figures 2.6 and 2.7 depict temporal decoding for different numbers of neurons. For the NEF, temporal encoding and decoding are determined by the biophysics of cellular communication. In short, the postsynaptic current (PSC) discussed in section 2.1, is used to convert spikes generated by a model of a single cell into an estimate of the input signal over time (see figure 2.6). Specifically, the spike patterns of the neurons being driven by the input are decoded by placing a PSC at each spike time and summing the result. More precisely, the “on” neuron PSCs are multiplied by +1 and the “off” neuron PSCs are multiplied by -1 (this is a linear decoding since we are multiplying a constant function (the PSC) by a weight (± 1) and summing).

When we apply this same principle to many neurons, as in figure 2.7, the set of PSCs induced by a given neuron need a specific weight, which is precisely the population decoder discussed earlier. Combining these decoding weights with a standard PSC model completely defines a linear “population-temporal” decoder. In this way, many neurons can “cooperate” to give very good representations of time-varying input signals. Notice that adding additional neurons accomplishes two things. First, it allows the nonlinearities of the tuning curves to be linearized as previously discussed with respect to population decoding. Second, it allows the very “spiky” decoding in figure 2.6 to become smoothed as the PSCs are more evenly distributed over time. These concepts are illustrated in the tutorial described in section 2.5.

Having specified the encoding and decoding used to characterize time-varying representation, we still need to specify the relevant alphabets. While the specific

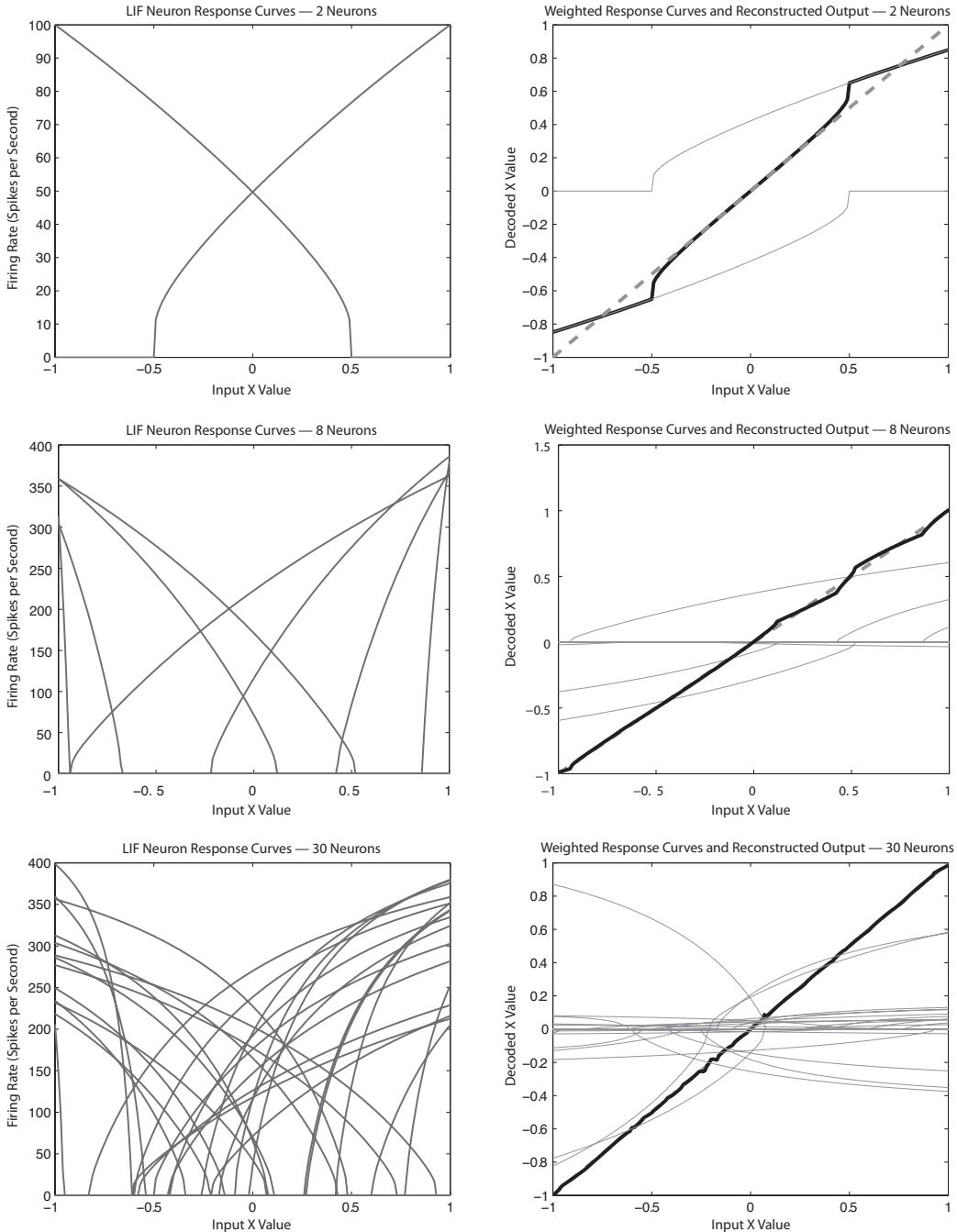


Figure 2.5: Finding decoders for a set of neurons. This set of figures demonstrates how neuron tuning curves can be individually weighted by decoders to approximate an input variable, x , over a range of values. The left column shows the tuning curves of the neurons. The right column shows linearly weighted versions of these curves in gray and the sum of these weighted curves as a thick black line. As the number of neurons is increased, the approximation to ideal representation (the straight dashed line) improves.

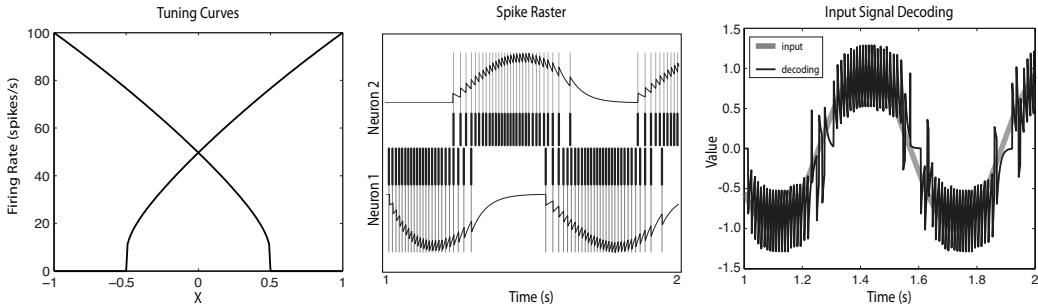


Figure 2.6: Encoding and decoding of an input signal by two neurons. The neurons fire at rates specified by their tuning curves, shown here to be symmetric about zero. With a sinusoidally varying input, these tuning curves cause the neurons to fire as shown by the black vertical lines in the spike raster. The continuous black lines in the middle plot show the postsynaptic currents that would be generated in the dendrites of a receiving neuron by the spike rasters. In the last panel, the PSCs are summed to give an estimate (black) of the original input signal (gray). The estimate is poor with two neurons, but can be made much better with additional neurons (see figure 2.7).

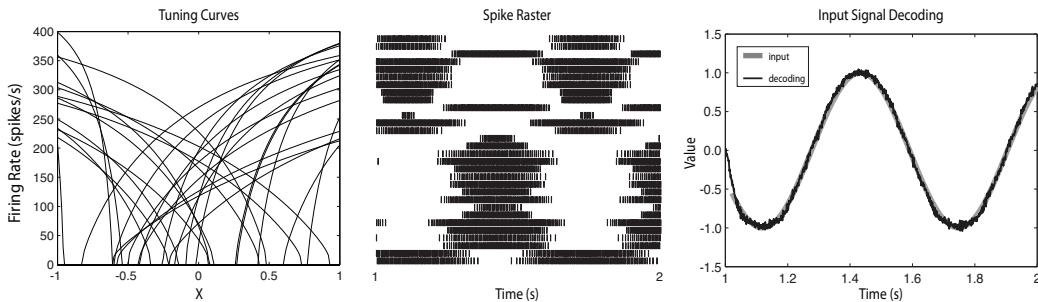


Figure 2.7: Encoding and decoding of an input signal by 30 neurons. As above in 2.6, the tuning curves of the neurons characterize their general response to input, the spike raster depicts the firing pattern of each neuron in response to a sinusoidal input signal, and the final panel shows the decoded estimate of the input signal. The temporal decoders are PSCs and the population decoders are optimal weights (determined as depicted in figure 2.5).

cases will diverge greatly, we can describe the alphabets generally: neural responses (encoded alphabet) code physical properties (decoded alphabet). Slightly more specifically, the encoded alphabet is the set of temporally patterned neural spikes over populations of neurons. This is reasonably uncontroversial.

However, it is much more difficult to be specific about the nature of the alphabet of physical properties. We can begin by looking to the physical sciences for categories of physical properties that might be encoded by nervous systems. Indeed, we find that many of the properties that physicists traditionally use to describe the physical world do seem to be represented in nervous systems: there are neurons sensitive to displacement, velocity, acceleration, wavelength, temperature, pressure, mass, etc. But, there are many physical properties not discussed by physicists that also seem to be encoded in nervous systems: such as red, hot, square, dangerous, edible, object, conspecific, etc.⁵ It is reasonable to begin with the hypothesis that these “higher-level” properties are inferred on the basis of representations of properties more like those that physicists talk about.⁶ In other words, encodings of “edible” depend, in some complex way, on encodings of “lower-level” physical properties like wavelength, velocity, etc. The NEF itself does not determine precisely what is involved in such complex relations, although I do suggest that it provides the necessary tools for describing such relations. I return to these issues – related to the meaning (or “semantics”) of representations – throughout much of the book, starting in chapter 3.

For now, we can be content with the claim that whatever is represented, it can be described as some kind of structure with units. A precise way to describe structure is to use mathematics. Hence, this is equivalent to saying that the decoded alphabet consists in mathematical objects with units. The first principle of

⁵I am allowing any property reducible to or abstractable from physical properties to count as physical. My suspicion is that this captures all properties, but that is a conversation for another book. Notice, however, that many typical physical properties, like temperature, are both abstracted from lower-level properties and are part of physics. So, abstracting from physical properties does not make properties non-physical. In philosophical terms, this is the claim that properties “supervening” on physical properties are physical.

⁶Clearly, this hypothesis can be wrong. It may turn out that no standard physical properties are directly encoded by neurons, but in the end this probably does not matter. We can still *describe* what they encode in terms of those properties, without being inaccurate (though our terminology may be slightly cumbersome; i.e. we may speak of “derivatives-of-light-intensity”, and so on). In short, it makes sense to begin to describing the properties encoded by neurons in terms of familiar physical properties, because so much of our science is characterized in terms of those properties, we are familiar with how to manipulate those properties, and in the end we need a description that works (not necessarily the only, or even best possible, description).

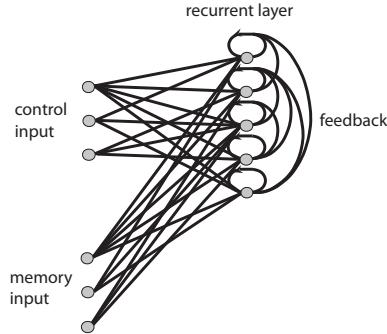


Figure 2.8: The architecture of a controlled integrator. This network can act like a loadable and erasable memory. Both the control input and the memory input are connected to the recurrent layer of neurons and the feedback to the recurrent layer is the product of both inputs. The NEF allows the neural connection weights in this network of spiking neurons to be determined, once we have defined the desired representations, computations, and dynamics of the system.

the NEF, then, provides a general characterization of the encoding and decoding relationship between mathematical objects with units and patterns of spikes in populations of neurons.

To make this characterization more concrete, let us turn to considering the example of the controlled integrator (see figure 2.8).

One of the simplest mathematical objects is a scalar. We can characterize the horizontal position of an object in the environment as a scalar, whose units are degrees from mid-line. There are neurons in a part of the monkey's brain called the lateral intraparietal (LIP) cortex, that are sensitive to this scalar value (Andersen et al., 1985). Indeed, these parts of the brain seem to act as a kind of memory for object location, as they are active even after the object has disappeared. I should be clear that I do not think a simple controlled integrator that remembers only a scalar value maps well to these specific neurons, but a similar architecture with a more complex representation has been used to model many properties of LIP activity (Eliasmith and Anderson, 2003). What is relevant for this example is that, as a population, neurons in this area *encode* an object's position over time.

To summarize, the representation of object position can be understood as a scalar variable, whose units are degrees from mid-line (decoded alphabet), that is encoded into a series of neural spikes across a population (encoded alphabet). Using the quantitative tools mentioned earlier, we can determine the relevant decoder (see appendix A.1.2). Once we have such a decoder, we can then estimate

what the actual position of the object is given the neural spiking in this population, as in figure 2.7. Thus we can determine precisely how well, or what aspect of, the original property (in this case the actual position) is represented by the neural population. We can then use this characterization to understand the role that the representation plays in the system as a whole.

One crucial aspect of this principle of representation, is that it can be used to characterize arbitrarily complex representations. The example I have described here is the representation of a scalar variable. However, this same principle applies to representations of vectors (such as movement or motion vectors found in motor cortex, brainstem, cerebellum, and many other areas), representations of functions (such as stimulus intensity across a spatial continuum as found in auditory systems, and many working memory systems), representations of vector fields (such as the representation of a vector of intensity, color, depth, etc. at each spatial location in the visual field as found in visual cortex), and representations of composable symbol-like objects (as argued for throughout this book). Suggesting that one of these kinds of representation can be found in a particular brain area, crucially, does not rule out the characterization of the same areas in terms of other kinds of representations. This is because we can quantitatively define a “representational hierarchy” that relates such representations to one another. I return to this issue in section 2.4.

A second crucial aspect of this principle is that it distinguishes the mathematical object being represented from the neurons that are representing it. I refer to the former as the “state space”, and the latter as the “neuron space”. There are many reasons it is advantageous to distinguish the neuron space from the state space. Most such advantages should come clear in subsequent chapters, but perhaps most obviously, distinguishing these spaces naturally accounts for the well-known redundancy found in neural systems. Familiarity with Cartesian plots makes us think of axes in a space as being perpendicular. However, the common redundancy found in neural systems suggests that their “natural” axes are in fact not perpendicular, but rather slanted towards one another (this is sometimes called an “overcomplete” representation). Distinguishing the state space, where the axes typically are perpendicular, from the neuron space, where they are not, captures this feature of neurobiological representation.

A third crucial aspect of this principle is that it embraces the heterogeneity of neural representation. There are no assumptions about the encoding neurons being similar in any particular respect. In other work, we have shown that such a representation is nearly as good as an optimally selected representation, both in terms of capacity and accuracy Eliasmith and Anderson (2003, pp. 206-217).

And, such a representation does not need mechanisms for optimal selection of new neurons (the neural properties are simply randomly chosen, as in figure 2.7). In practice, this means that the typically heterogeneous tuning curve data from a particular neural system can be matched in a model (since the principle puts no constraints on tuning curve distribution), and then optimal decoding can be subsequently determined.

The tutorial at the end of this chapter demonstrates how to build and interact with simulations of heterogeneous scalar and vector representations in Nengo. The central features of the representation principle are further highlighted there with concrete examples.

2.3.2 Transformation

A representational characterization is not be useful if it does not help us understand how the system functions. Conveniently, this characterization of neural representation paves the way for a general characterization of how these representations can be transformed. This is because, like representations, transformations (or computations) can be characterized using decoding. But, rather than using the “representational decoder” discussed above, we can use a “transformational decoder”. We can think of the transformational decoder as defining a kind of *biased* decoding. That is, in determining a transformation, we extract information *other than* what the population is taken to represent. The bias, then, is away from a “pure”, or representational, decoding of the encoded information.

For example, if we think that the quantity x is encoded in some neural population, when defining the representation we determine the representational decoders that estimate x . However, when defining a computation we identify transformational decoders that estimate some function, $f(x)$, of the represented quantity. In other words, we find decoders that, rather than extracting the signal represented by a population, extract some transformed version of that signal. The same techniques used to find representational decoders are applicable in this case, and result in decoders that can support both linear and nonlinear transformations (see Appendix A.2). Figure 2.9 demonstrates how a simple nonlinear function can be computed in this manner.

Perhaps surprisingly, this is all there is to neural transformation. Instead of decoding a representation of x from a spike train, we decoding a function of x (i.e. $f(x)$) from that same spike train. Importantly, this understanding of neural

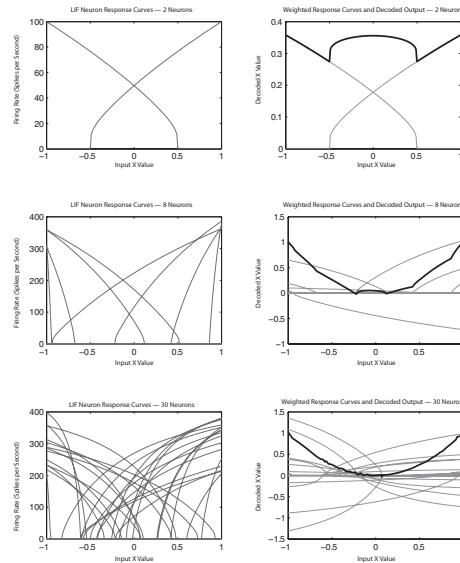


Figure 2.9: A nonlinear function of the input computed over time. The left column shows the tuning curves of the neurons which are identical to the curves shown in figure 2.5. The right column shows linearly weighted versions of these curves in gray and the sum of these weighted curves as a thick black line. As can be seen, the choices of linear weights used result in a quadratic function over the range of -1 to 1 . Under this decoding scheme, the network is thus computing x^2 .

computation applies at all levels of the representational hierarchy and accounts for complex transformations. For example, it can be used to define inference relations, be they statistical (Eliasmith and Anderson, 2003, chp. 9), or more linguistic (see section 4.3). So, although linear decoding is simple, it can support the kinds of complex, nonlinear transformations needed to articulate descriptions of cognitive behavior. A main purpose of this book is to articulate one such description through the identification of relevant transformations, as discussed in later chapters.

For present purposes, let us again consider the simple, specific example of a controlled integrator. The reason this integrator is “controlled” is because, unlike a standard integrator, we can change the dynamics of the system by changing the input. Comparison of ideal controlled, and ideal non-controlled integrators are shown in figure 2.10. It is evident there that the dynamics of the system in the controlled integrator is a function of its input. Specifically, the “control variable” is able to make the integrator act as a standard integrator, or exponentially forget its current state.

In order to build such a system, it is necessary to compute the product (i.e., a nonlinear function) of the current state (i.e., the memory) and the control variable. Consequently, to implement such a system in a neural network, it is necessary to transform the represented state space of the recurrent layer (see figure 2.8) in such a way that it estimates this nonlinear function. To do so, it is important note that the state space of the recurrent layer represents both the memory and the control inputs. Hence, it contains a 2D vector representation $\mathbf{x} = [x_1, x_2]$. To compute the necessary transformation of this state space, we can find transformational decoders to estimate the function $f(\mathbf{x}) = x_1 x_2$, just as we earlier found the decoders to estimate the function $f(x) = x^2$ for a 1D scalar (see figure 2.9). The tutorial in section 3.8 demonstrates how to construct a network that performs scalar multiplication in this manner.

Before moving on to a consideration of dynamics, it is important to realize that this way of characterizing representation and computation does not demand that there are “little decoders” inside the head. That is, this view does not entail that the system itself needs to decode the representations it employs. In fact, according to this account, there are no directly observable counterparts to the representational or transformational decoders. Rather, as discussed in section 2.3.4, they are embedded in the synaptic weights between connected neurons. That is, coupling weights of connected neurons indirectly reflect a particular population decoder, but they are not identical to the population decoder. This is because connection weights are best characterized as determined by both the decoding of the

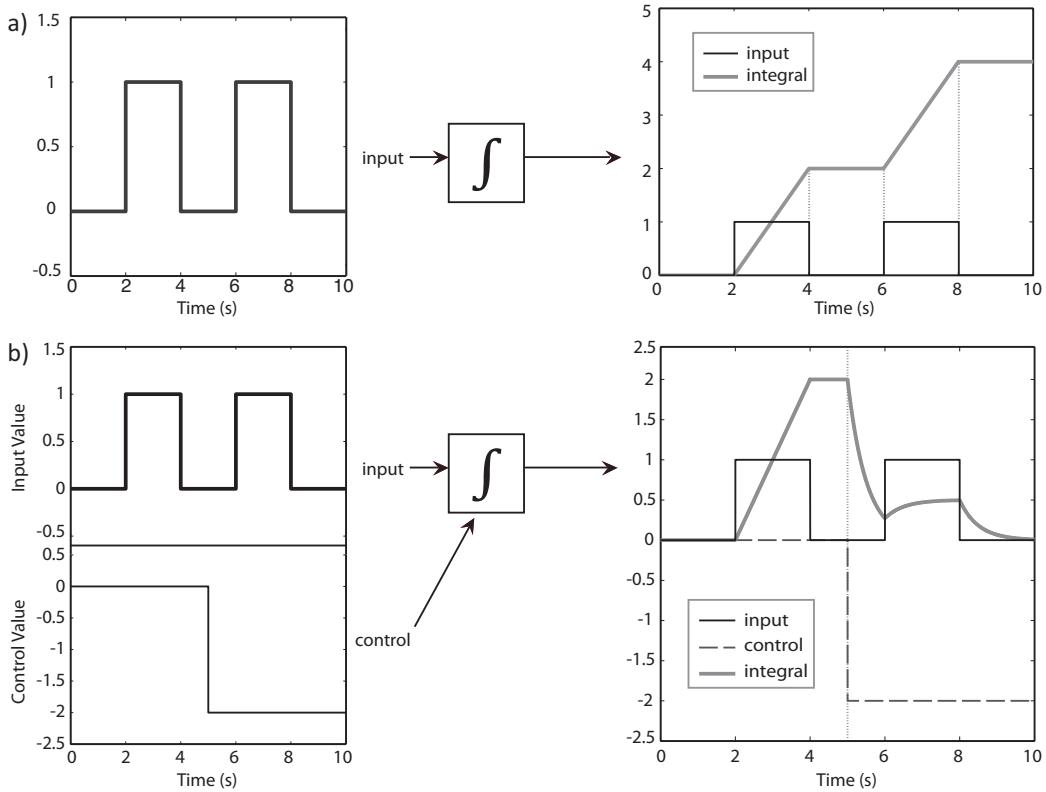


Figure 2.10: Comparison of a controlled and uncontrolled integrator. All x -axes represent time and y -axes represent the values of functions. a) A standard integrator continuously adds its input to its current state. These dynamics do not change regardless of the input signal. b) A controlled integrator acts like a standard integrator if the control variable is equal to zero. Values less than zero cause the integrator to exponentially forget its current state. The speed of forgetting is proportional to the value of the control variable.

incoming signal and the encoding of the outgoing signal. Practically speaking, this means that changing a connection weight both changes the transformation being performed and the tuning curve of the receiving neuron. As is well known from both connectionism and theoretical neuroscience, this is exactly what happens in such networks. In essence, the encoding/decoding distinction is not one that neurobiological systems need to respect in order to perform their functions, but it is extremely useful in trying to understand such systems and how they do, in fact, manage to perform those functions. Consequently, decoders – both transformational and representational – are theoretical constructs. The only place something like decoding actually happens is at the final outputs of the nervous system, such as at the motor periphery.

2.3.3 Dynamics

In the history of cognitive science, computation and representation have always been central players in our cognitive theories, but dynamics less so. And, while it may be understandable that dynamics were initially ignored by those studying cognitive systems as purely computational systems, it would be strange indeed to leave dynamics out of the study of minds as physical, neurobiological systems. Even the simplest nervous systems performing the simplest functions demand temporal characterizations: moving, eating, and sensing in a constantly changing world are clearly dynamic processes. It is not surprising, then, that single neural cells have almost always been modeled by neuroscientists as essentially dynamic systems. In contemporary neuroscience, researchers often analyze neural responses in terms of “onsets”, “latencies”, “stimulus intervals”, “steady states”, “decays”, etc. – these are all terms describing temporal aspects of a neurobiological response. The fact is, the systems under study in neurobiology are dynamic systems and as such they make it very difficult to ignore time.

Notably, modern control theory was developed precisely because understanding complex dynamics is essential for building something that works in the real world. Modern control theory permits both the analysis and synthesis of elaborate dynamic systems. Because of its general formulation, modern control theory applies to chemical, mechanical, electrical, digital, or analog systems. As well, it can be used to characterize non-linear, time-varying, probabilistic, or noisy systems. As a result of this generality, modern control theory is applied to a huge variety of control problems, including autopilot design, spacecraft control, design

of manufacturing facilities, robotics, chemical process control, electrical systems design, design of environmental regulators, and so on. It should not be surprising, then, that it also proves useful for characterizing the dynamics of complex neurobiological systems.

Central to employing modern control theory for understanding the dynamics of a system is the identification of the “system state variable” ($\mathbf{x}(t)$) in figure 2.11). It is not a coincidence that the terminology introduced earlier has us calling the represented mathematical objects of a neural population the “state space”. This is because the third principle of the NEF is the suggestion that the representations of neural populations can be characterized as the state variables of a dynamical system using control theory.

However, things are not quite so simple. Because neurons have intrinsic dynamics dictated by their particular physical characteristics, we must adapt standard control theory to neurobiological systems (see figure 2.11). Fortunately, this can be done without loss of generality for linear and nonlinear dynamic systems (see Appendix A.3). Notably, all of the computations needed to implement such systems can be implemented using transformations as defined earlier in principle 2. As a result, we can directly apply the myriad techniques for analyzing complex dynamic systems that have been developed using modern control (and dynamic systems) theory to this quantitative characterization of neurobiological systems.

To get a sense of how representation and dynamics can be integrated, let us revisit the simple controlled integrator. As shown in figure 2.10, a standard integrator constantly sums its input. If we call the input $u(t)$ and the state of the controlled $x(t)$, we can write this relation as

$$\dot{x}(t) = Ax(t) + Bu(t) \quad (2.1)$$

which is the scalar version of the system shown in figure 2.11a. This equation says that the change in $x(t)$ (written $\dot{x}(t)$) at the next moment in time is equal to its current value plus the input $u(t)$ (times some number B). For the rest of the discussion, we can let $B = 1$.

If we also let $A = 0$, then we get a standard integrator: the value of the state $x(t)$ at the next moment in time will be equal to itself plus the change $\dot{x}(t)$. That change is just equal to the input $u(t)$ (since $0x(t) = 0$), so the value of $x(t)$ will result from constantly summing the input, just as in a standard integrator. In short,

$$x(t) = \int u(t) dt. \quad (2.2)$$

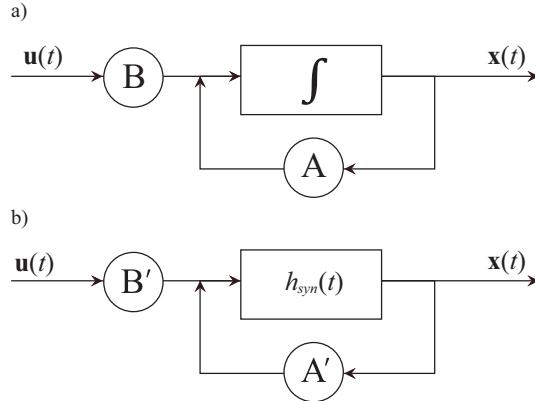


Figure 2.11: A control theoretic description of neurobiological systems. a) The canonical diagram for any linear system in control theory. \mathbf{A} determines what aspects of the current state affect the future state. \mathbf{B} maps the input to the system into its state space. The transfer function is perfect integration. b) The equivalent description for a neurobiological system. The matrices \mathbf{A}' and \mathbf{B}' take into account the effects of the transfer function, $h_{syn}(t)$, which captures the dynamics of neurons by modelling the postsynaptic current (PSC).

In many ways, this simple integrator is like a memory. After all, if there is no input (i.e., $u(t) = 0$) then the state of the memory will not change. This captures the essence of an ideal memory – its state stays constant over time with no input.

However, as noted in figure 2.11, neuron dynamics are not well-characterized by perfect integration. Instead, they have dynamics identified by the function $h_{syn}(t)$ in that figure. Specifically, this term captures the postsynaptic current (PSC) dynamics of the neurons in question, several examples of which are shown in figure 2.12. As mentioned earlier, these dynamics depend on the kind of neurotransmitter being used. As a result, the “translation” from a control theoretic description to its neural equivalent must take these differences into account. Specifically, that translation must tell us how to change A and B in figure 2.11a into A' and B' in figure 2.11b, given the change from integration to $h_{syn}(t)$.

For example, if we apply this translation to the neural integrator, the feedback matrix A' is equal to 1 and the input matrix B' is equal to the time constant of the PSC. This makes sense because the dynamics of the neural system decay at a rate captured by the PSC time constant, so the feedback needs to “remind” the system of its past value at a similar rate. Setting A' and B' to these values will result in behavior just like that described for a perfect integrator in equation 2.2.

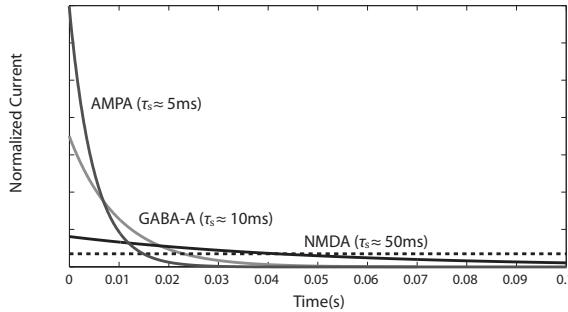


Figure 2.12: Example postsynaptic currents for different neurotransmitters. This figure shows a simple postsynaptic current model as a decaying exponential in response to a single neural spike. The time constant of the exponential is different for different receptors, as shown. The longer the time constant, the slower the decay. Compared to the dynamics of a perfect integrator (dashed line), PSCs lose information about their initial state over time. Each PSC here is normalized to have an area of 1 under the curve, so the peak value is not indicative of the strength of response but of the relative weighting of earlier versus later aspects of the response.

As mentioned before, this translation is defined in general for dynamical systems in appendix A.3.

The problem with this as a characterization of memory is that a simple integrator always just adds new input to the current state. If you can only remember one thing, and I ask you to remember the word “cat”, and then later ask you to remember the word “dog”, we would not expect you to report some kind of “sum” of “cat” and “dog” as the contents of your memory when queried later. Instead, we would expect you to *replace* your memory of “cat” with “dog”. In short, there must be a way to empty the integrator before a new memory is remembered.

This is why we need to control the value of A in the above system. Consider for a moment the effect of changing the value of A in equation 2.1 to lie anywhere between -1 and 1. Let us suppose that the current value of $x(t)$ is not 0, and there is no input. If A is positive, then all future values of $x(t)$ will move increasingly far away from 0. This is because the change in $x(t)$ will always be positive for positive $x(t)$ and negative for negative $x(t)$. In fact, this movement away from zero will be exponential because the next change is always a constant fraction of the current state (which keeps getting bigger) – adding (or subtracting) a fraction of something to itself over and over results in an exponential curve. Now suppose

that A is negative. In this case, the opposite happens. The current state will move exponentially towards 0. This is because the next state will be equal to itself minus some fraction of the current state.

To build a controlled integrator that acts as an erasable memory, we can thus implement the dynamical system in equation 2.1, with an additional input that controls A . We can use principle 3 to determine how to map this equation into one that accounts for neural dynamics. We then need to employ principle 2 to compute the product of A and $x(t)$ to implement this equation (as in the previous section). And, we need to employ principle 1 to represent the state variable, the input variable, and the control variable in spiking neurons. The result of employing all three principles is shown in figure 2.13. Ultimately, the principles tell us how to determine the appropriate connection weights between spiking neurons in the model (recurrent or otherwise), to give the desired dynamical behavior.

It is perhaps worth emphasizing that while the controlled integrator is simple – it computes only products and sums, it represents only scalars and a 2D vector, and it has nearly linear time-invariant dynamics – it employs each of the principles of the NEF. Furthermore, there is nothing about these “simplicities” that are a consequence of the principles employed. NEF models can compute complex non-linear functions, have sophisticated high-dimensional representations, and display more interesting, nonlinear dynamics.

It is perhaps worth pointing out here that such sophistication, while available in the NEF, is not necessary for giving us a deeper understanding of neural function. For example, Ray Singh and I used two simple (non-controlled) neural integrators, coupled them together, and provided a novel explanation for observed spiking patterns in monkey cortex during a working memory task (Singh and Eliasmith, 2006). Specifically, the first integrator integrated a brief input stimulus to result in a constant output (i.e. a memory of the input), and the second integrator integrated the output of the first, to give a “ramp” (i.e. a measure of how long since the input occurred). We demonstrated that projecting these two outputs onto a population of cells representing a 2-dimensional vector gave all of the observed classes of spiking single-cell responses in a working memory experiment performed on monkey frontal cortex, including responses with unusual dynamics (Romo et al., 1999a). A recent analysis of a similar, but larger data set verified that this kind of model – one with linear dynamics in a higher-dimensional space – captures 95% of the observed responses of the over 800 observed neurons (there were 6, not 2 dimensions required for this model; Machens et al., 2010). Another very similar model (which uses coupled *controlled* integrators) captures the population dynamics, and single cell spiking patterns observed in rat medial prefrontal

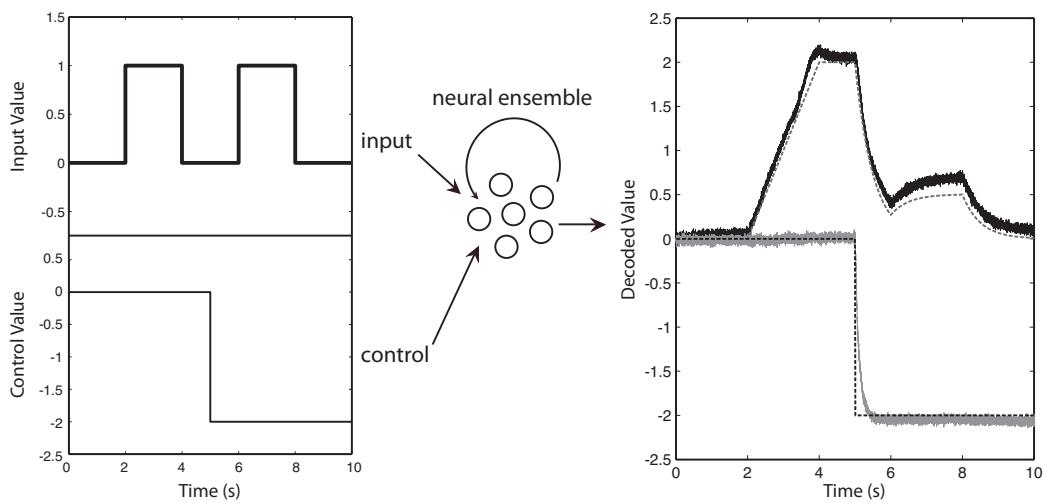


Figure 2.13: A recurrent spiking network of several hundred neurons implementing a controlled integrator. Input and control signals, shown on the left, are connected to a neural ensemble which also has recurrent connections (shown in the middle). The neural ensemble contains spike-based representations of both the control signal and the value of the current state, $x(t)$. Decodings of these representations are shown in the rightmost graph: the solid black line is the decoded representation of $x(t)$ (neural spikes are not shown for clarity), the solid gray line is the representation of the control signal, and dotted lines depict the ideal values of the integral and the control. The tutorial at the end of chapter 5 demonstrates how to build this controlled integrator in Nengo.

cortex similarly well, though on a very different task (Laubach et al., 2010). In both cases, a surprisingly good explanation of the complex, detailed spiking patterns found in “higher” cortical areas (like the prefrontal cortex) was discovered despite the simplicity of the model.

In many ways, it is the successes of these simpler models, built using the principles of the NEF, that make it reasonable to explore the available generalizations of those principles. Ultimately, it is this generality, I believe, that puts the NEF in a unique position to help develop and test novel hypotheses about biological cognition. Developing and testing one such hypothesis is the purpose of the remainder of this book, beginning with the next chapter.

2.3.4 The three principles

Hopefully it is clear that this presentation of the NEF is somewhat superficial. It is not intended to satisfy those deeply familiar with theoretical neuroscience. I have side-stepped issues related to optimal decoding, nonlinear versus linear decoding, information transfer characteristics, nonlinear dendritic function, rate versus timing code considerations, and so on. While I believe the NEF satisfactorily addresses these issues, my purpose here is to give the necessary background to make the methods plausible, usable, and not unnecessarily complex.

In that spirit, it is perhaps useful to summarize the principles diagrammatically, to both clarify how they relate to neurobiology and to demonstrate what they contribute to an analysis of neural systems. Figure 2.14 shows what I have called a “generic neural subsystem”. The purpose of identifying such a subsystem is to note that such “blocks” can be strung together indefinitely to describe neural function. This subsystem is generic because it has input spikes, which generate postsynaptic currents (PSCs) that are then weighted and passed through a neural non-linearity that then generates outputs spikes. This characterization is generic across almost all areas of the mammalian brain.⁷

What the NEF adds to the standard neural-level description of a population of neurons is the decomposition of synaptic weights into various elements, and a mapping of the dynamics onto biophysical properties of neurons. Again consider

⁷Some notable exceptions are retinal processing, some dendro-dendritic interactions, and gap junctions. An extension to the generic neural subsystem as drawn here can capture such interactions by incorporating optional elements that include these other potential sources of dendritic current. This added complexity obscures the simplicity and broad generality of the subsystem, however.

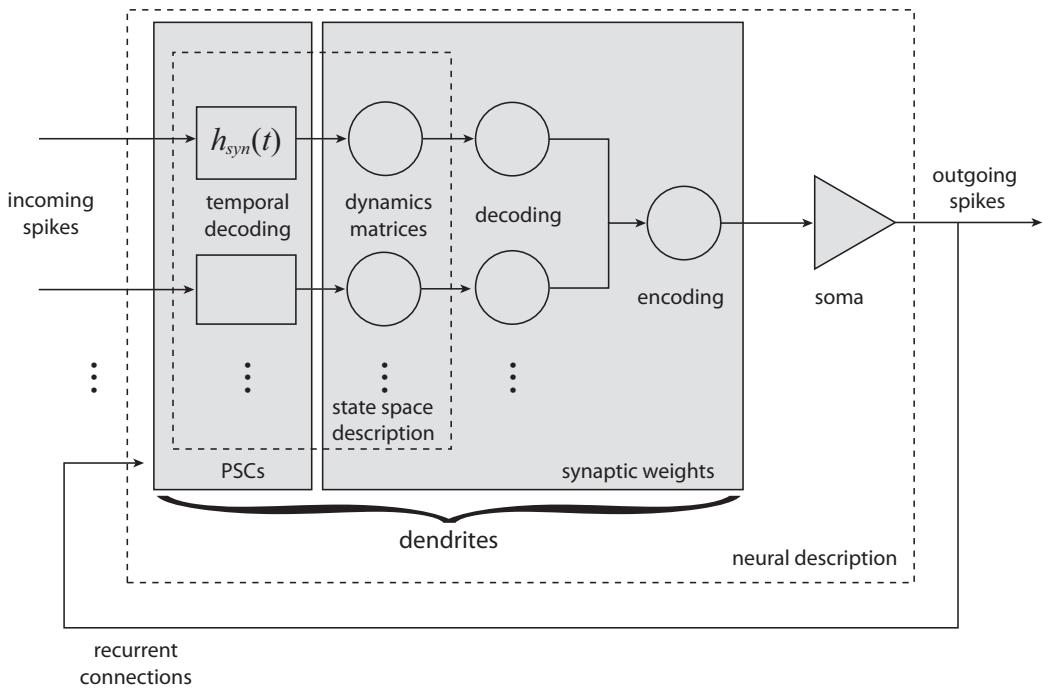


Figure 2.14: A generic neural subsystem. A synthesis of the preceding characterizations of representation (encoding/decoding), computation (biased decoding), and dynamics (captured by $h_{syn}(t)$ and the dynamics matrices, A' and B'). Dotted lines distinguish neural and state space descriptions. Encoding and decoding relate these two descriptions to one another.

the neural integrator as an example of the system shown in figure 2.11. There, the A' matrix specifies the dynamics matrix which connects outgoing spikes of the population back to itself (the recurrent connection in figure 2.14, equal to 1 for the integrator). The B' matrix specifies the dynamics matrix that connects the incoming spikes to this population (equal to τ_{syn} for the integrator). Associated with both of these dynamics matrices are the decoding and encoding elements, which in this case are the same and specified by the neuron responses (encoders) and the optimal linear weights for estimating the input (decoders). Finally, the dynamics of the system are capture by the PSC model being used (examples are shown in figure 2.12). Having specified all of these elements allows us to generate the synaptic connection weights necessary to implement the defined higher-level dynamical system in a population of spiking neurons.

In this way, the subsystem in figure 2.14 captures the contributions of the principles. That is, together the principles determine the synaptic weights. That determination depends on the contributions of each of the principles independently: the representation principle identifies encoders and representational decoders; the transformation principle identifies transformational decoders; and the dynamics principle identifies the dynamics matrices (i.e. the matrices identified in figure 2.11). The synaptic weights themselves are the product of these elements.

Crucially, while this theoretical characterization of the subsystem is generic, its application is not. To determine decoders, the specific tuning curves of neurons in the circuit play a role. To determine dynamics, the kinds of neurotransmitters found in a given circuit are crucial. To determine what kind of spiking occurs, the choice of a single cell model is important. To determine encoders, both the single cell model and the tuning curves need to be known. To determine transformational decoders and dynamics, a high-level hypothesis about the system function is needed. All of these considerations can vary widely depending on which brain areas are being considered.

I mentioned several examples of the broad application of the NEF at the beginning of section 2.3. As noted, these applications have focused on detailed neural models. A benefit of this focus has been that the NEF has been used to account for and predict a myriad of detailed neural properties, such as changes in single cell tuning curves under different environmental circumstances (Conklin and Eliasmith, 2005), the variety of single cell dynamics observed during working memory (Singh and Eliasmith, 2006) and other delayed activity tasks (ref Laubach???, recovery from ablation of single cells, the effects of systematic input perturbation, and the normal functioning of the neurobiological integrator controlling eye position (ref macneill???) among many others. In short, the credentials of the NEF as

a method for generating models with close connections to detailed neural data are good.

As well, these applications make it clear that the NEF bears little resemblance to traditional connectionism: NEF neurons spike, they are highly heterogeneous, they have a variety of different dynamics, they are differentially affected by different neurotransmitters – they are not “units” or “nodes”, but *neurons*. In addition, the NEF does not rely on learning to design models (although learning can be included if appropriate). Consequently, the NEF can be used to construct arbitrarily “deep” (i.e. with any number of layers), and arbitrarily connected models, without running afoul of the many difficulties of learning such a model. This is because NEF networks can be designed from the top down (given a functional hypothesis, how might neurons be organized to realize that function). This top-down approach has been used in all of the previously mentioned applications of the NEF to specific systems.

One drawback of these biologically focused applications of the NEF is that the kinds of functions addressed are simple, making it less than obvious how the NEF might be relevant for understanding complex behavior. It has been suggested that a detailed understanding of neural implementation is completely irrelevant for understanding cognition (Fodor and Pylyshyn, 1988b). I, and many others, disagree, though I won’t argue the point here. But if we disagree, we must also admit that it is not obvious what the relation *is* between these simple neural systems and cognition, and presumably we want to understand the cognitive forest *as well as* the neural trees.

Before confronting the task of applying the NEF to biological cognition in the next chapter, I believe it is conceptually important to clarify a notion I will be using throughout the remainder of the book. A notion I have been using somewhat loosely to this point: the notion of “levels”.

2.4 Levels

At one point in the development of science, it was widely thought that the sciences could be identified with the level of nature which they characterized. Physics was the most fundamental, followed closely by chemistry, and subsequently biology, psychology, sociology and so on (Oppenheim and Putnam, 1958; Carnap, 1931; Hempel, 1966). The suggestion was that lower-level sciences could reductively describe the higher-level sciences. Societies, after all, are composed of people, who are made up of biological parts, which are largely driven by chemical processes, which can be understood as interactions between the fundamental parts of

nature. This uncomplicated view has unfortunately not withstood the test of time (Bechtel, 1988, 2008; Craver, 2007). However, the notion that there are “levels” of some kind has proven too useful to discard. Instead it has become a perennial and complex problem to determine what, exactly, the relation between the sciences, and the entities they talk about, is (Fodor, 1974; Wilson; Batterman, 2002).

Jerry Fodor has famously suggested that, since the reduction of one science to another has failed, the sciences must be independent (Fodor, 1974). Consequently, he has argued that to understand cognitive systems, which lie in the domain of psychology, appeal to lower-level sciences is largely useless. The only thing such disciplines, including neuroscience, can do is to provide an implementational story that bears out whatever psychological theory has been independently developed. There are many difficulties with such a view, but the one that stands out in the present context is that cleaving neuroscience from psychology simply throws out an enormous amount of empirical data about the system we are attempting explain. If we were certain that we could achieve an adequate, purely psychological-level description of the system, this strategy might be defensible. However, we are in no such position.

There are other possible relations between levels other than the extremes of their being reducible to one another, or their being independent. I suspect that the reason that both the reducibility and the independence views seem implausible is because they share a very strong interpretation of the term “level” in nature. Specifically, both assume that such levels are *ontological* (Oppenheim and Putnam, 1958). That is, that the levels themselves are intrinsic divisions in the structure of the natural world. The reason we might assume we can reduce theories about people to theories about particles, is because people seem ultimately describable as a bunch of particles stuck together in some (perhaps very complex) way. The reason we might assume theories about people are independent of theories about particles, is because we have given up on reduction, but still believe that both people and particles are equally real and so believe that both kinds of theory are equally scientific.

If, instead, we think of levels from an *epistemological* perspective, that is, as different *descriptions* of a single underlying system, neither view seems plausible. Sometimes we may describe a system as a person without much consideration of their relationship to individual particles. Other times we may be concerned about the decomposition a biological person, leading to what looks like a kind of reductive description. The reasons for preferring one such description over another can be largely practical: in one case we need to predict the overall behavior of the whole system; in another we need to explain how changes in a component influ-

ence complex interactions. In either case, we need not take these different perspectives on the underlying system as something we need reify (i.e., make real), especially at the expense of other perspectives. Rather, we can understand levels to be kinds of description that are chosen for certain explanatory and predictive purposes.

Crucially, this does not mean that levels are in any problematic sense “made up”: they can still be right and wrong (depending on whether they are consistent with empirical evidence), and they can still be about real things (i.e., things whose existence does not depend on our existence). Instead, “levels as kinds of description for purposes” are flexible in a manner that acknowledges our limited intellectual capacities, or maybe simply our limited knowledge. It is sometimes important to make assumptions or simplifications in order to make predictions with our theoretical descriptions. In psychology, we may talk about people as if they were unchanging sets of particles because most of our predictions are sufficiently accurate regardless of that assumption. The psychological level may thus pick out descriptions that employ a consistent set of assumptions which are effective at maintaining the explanatory and predictive power of those descriptions..

In short, I have suggested that levels are pragmatically identified sets of descriptions that share assumptions. Perhaps it is useful to call this position “descriptive pragmatism” for short. Importantly, I think descriptive pragmatism does a reasonable job of capturing a variety of intuitions about “levels” evident in scientific practice. For example, descriptive pragmatism can help us understand why levels, spatial scale, and complexity seem inter-related. Consider: as we increase the number of objects in a system, we need to increase the amount of space necessary to encapsulate the system, and the number of possible interactions goes up rapidly. So, given our constant intellectual capacity, typically, larger spatial scales allow for more complexity and hence concurrently demands higher-level descriptions to address the behavior of the entire system. And, more complexity often means more assumptions need to be made to keep the description simple enough for it to be practical to us. A lower-level description of the entire system will require fewer assumptions, but soon outstrip our intellectual capacity to represent (which is perhaps why we often turn to simulations for our lower-level descriptions).

Similarly, this view of levels is consistent with the notion that part/whole relations, especially in mechanisms, often help pick out levels. Most systems composed of parts have parts that interact. If the parts of a system are organized in such a way that their interaction results in some regular phenomena, they are called “mechanisms” (Machamer et al., 2000). Mechanisms are often analyzed

by decomposing them into their simpler parts, which may also be mechanisms (Bechtel, 2005). So, within a given mechanism, we again see a natural correspondence between spatial scale, complexity and levels. Note, however, that even if two distinct mechanisms can be decomposed into lower level mechanisms, this does not suggest that the levels *within* each mechanism can be mapped *across* mechanisms (Craver, 2007). This, again, suggests that something like descriptive pragmatism better captures scientific practice than more traditional ontological views (Bechtel, 2008), as it allows different practical considerations to be used to identify decompositions across different mechanisms.

Descriptive pragmatism about levels may initially seem to be a more arbitrary characterization of levels but, there is no reason not to demand systematic identification and quantification of the relations between identified levels. The descriptions of inter-level relations we employ can still be mathematical, after all. In fact, mathematical descriptions can help clarify the assumptions made when identifying levels. So, for example, we might talk in terms of mathematical objects that represent concepts instead of mathematical objects that capture individual neuron function because successful explanation or prediction of certain behavior can proceed in terms of concepts without detailed consideration of individual neuron function. The nature of “without detailed consideration” is specified by assumptions that underly our mathematical theory of concept representation that relates it to individual neuron function. And, identifying those assumptions is paramount to identifying the level of description being employed.

I should note that I in no way take this brief discussion to sufficiently specify a new characterization of levels. There are many, much deeper considerations of levels that I happily defer to (e.g., Bechtel, 2008; Craver, 2007; Hochstein, 2011). My purpose is to clarify what general characterization of levels lies behind my use of the term throughout the book, so as to avoid confusion about what I might mean. The main point to be taken from this discussion is simply that we need not think of levels as being either reductive (Oppenheim and Putnam, 1958) or independent (Fodor, 1974). There are accounts, instead, which identify levels with something more like degrees of descriptive detail.

With this background in mind let me return to specific consideration of the NEF. As described in section 2.3.1 on the principle of representation, the principle applies to the representation of all mathematical objects. Since such objects can be ordered by their complexity, we have a natural and well-defined meaning of a representational hierarchy: a hierarchy whose levels can be understood as kinds of descriptions with specific inter-relations. Table 2.1 provides the first levels of such a hierarchy.

Table 2.1: A representational hierarchy. Each row has the same type of encoding/decoding relations with neurons. Higher rows can be constructed out of linear combinations of the previous row.

Mathematical Object	Dimension	Example
Scalar (x)	1	Light intensity at x, y
Vector (\mathbf{x})	N	Light intensity, depth, color, etc. at x, y
Function ($x(\mathbf{v})$)	∞	Light intensity at all spatial positions
Vector Field ($x(\mathbf{r}, \mathbf{v})$)	$N \times \infty$	Light intensity, depth, color, etc. at all spatial positions
:	:	:

This hierarchy maps well to notions of levels as spatial scale, complexity, or part/whole relations. If we hold the precision of the representation constant, then the higher levels in the hierarchy will require more neurons, and hence typically be ascribed larger areas of cortex (relative few cells are needed to encode light intensity at a single spatial position, compared to encoding light intensity over the entire visual field). Relatedly, these higher-level representations will be able to under-write more complex behaviors (object recognition will often require detailed intensity information across large portions of the visual field). Finally, the hierarchy clearly defines how the higher-levels are built up out of the lower levels. Hence mechanisms trafficking in one level of representation will often be decomposable into mechanisms trafficking in lower-level representations.

Given the tight relationship between NEF principles 1 and 2, it should not be surprising that what goes for representations also goes for transformations. High-level computations will be carried out over larger spatial scales, can be more complex, and relate to high-level mechanisms. Furthermore, because principles 1 and 2 are used to implement principle 3 (dynamics are defined by transformations of representations), the same can be said for dynamics. Picking which level is appropriate for a given system will depend on what explanations we are trying to provide about the system. Hence, descriptive pragmatism is a good way to understand such a hierarchy, and how it is used.

Notice also that all of the levels of this hierarchy are described using the same encoding/decoding relationship with individual neurons. For this reason, I will often talk of neuron-level descriptions as being lower-level than these representational descriptions. In other words, the neuron space/state space distinction is also a distinction between levels of description. The representational hierar-

chy itself lies within the state space half of this distinction. The relation between the neuron and state space levels (i.e., nonlinear encoding and linear decoding) is different than that between levels within the representational hierarchy (i.e. perfect linear encoding and decoding). But, both are quantitatively defined, and the increase of levels with spatial scale, complexity, and part/whole relations still obtains. Consequently, it still makes sense to talk of all of these as levels of the same system.

The fact that all of these levels can be described quantitatively and in a standard form suggests that the NEF characterization provides a unified means of describing neurobiological systems. In addition, it makes clear how we can “move between” levels, and precisely how these levels are *not* independent. They are not independent because empirical data that constrains one description is about *the same system* described at a different level. So, when we move between levels, we must either explicitly assume away the relevance of lower (or higher) level data, or we rely on the hypothesized relationship between the levels to relate that data to the new level. Either way, the data influences our characterization of the new level (since allowable assumptions are part and parcel of identifying a level).

One final subtlety in the NEF characterization of levels in neural systems is worth noting. Specifically, it is consistent with this characterization that what we may naturally call a high-level neural representation (e.g., the representation of phonemes), is a relatively simple mathematical object (e.g., a scalar specifying only one of forty possible values). On the face of it, this characterization of phonemes as simple mathematical objects seems to contradict my earlier claim that, in general, higher-level representations are more complex mathematical objects. However, this observation misses the fact that the complexity of the representation has been “shifted” into the *units* of the object. That is, phonemes themselves are very complicated objects that require many sophisticated computations to extract (they depend not only on complex interactions of frequencies, but auditory and motor contexts). So the fact that the unit of the object is “phoneme” captures enormous amounts of complexity. Consequently, the representational hierarchy applies in cases when the units of the component representations stay constant (as in the example provided in table 2.1). If we change units, we may still proceed “up” levels even though the mathematical objects themselves get simpler. Of course, in such a case, the relationship between units at different levels must also be specified to properly characterize the relationship between these levels. This relationship is most often a transformational one, and hence not one that the NEF specifies in general (recall from section 2.3 that the NEF is about *how*, not *what*, neural systems compute).

So, the NEF provides a consistent and general way to talk about levels in the behavioral sciences. However, only some of the inter-level relations are defined by the principles of the framework. This seems appropriate given our current state of ignorance about how best to decompose the highest-level neural systems. In the next chapter, I begin to describe a specific architecture that addresses what kinds of functions brains may actually be computing in order to underwrite cognition. The NEF provides a method for specifying that description at many levels of detail, while the architecture itself helps specify additional inter-level relations not addressed by the NEF.

2.5 Nengo: Neural representation

In this tutorial, we examine two simple examples of neural representation. These examples highlight two aspects of how principle 1 in section 2.3.1 characterizes neural representation. First, the examples make evident how the activity of neural populations can be thought of as representing a mathematical *variable*. Second, we examine a simple case of moving up the representational hierarchy to see how the principle generalizes from simple (i.e., scalar) to more complex (i.e. vector) cases.

Representing a scalar

We begin with a network that represents a very simple mathematical object, a scalar value. As in the last tutorial, we will use the interactive mode to examine the behavior of the running model. Let us begin.

- In an empty Nengo world, drag a *Network* component from the template bar into the workspace. Set the *Name* of the network to ‘Scalar Representation’ and click *OK*.
- Drag an *Ensemble* into the network. A configuration window will open.

Here the basic features of the ensemble can be configured.

- Set *Name* to ‘x’, *Number of nodes* to 100, *Dimensions* to 1, *Node factory* to ‘LIF Neuron’, and *Radius* to 1.

The name is a way of referring to the population of neurons you are creating. The number of nodes is the number of neurons you would like to have in the population. The dimension is the number of different elements in the vector you would

like the population to represent (a 1 dimensional vector is a scalar). The node factory is the kind of single cell model you would like to use. The default is a simple leaky integrate-and-fire (LIF) neuron. Finally, the radius determines the size of the n -dimensional hypersphere that the neurons will be good at representing. A 1-dimensional hypersphere with a radius of 1 is the range from -1 to 1 on the real number line. A 2-dimensional hypersphere with the same radius is a unit circle.

- Click the *Set* button. In the panel that appears, you can leave the defaults (τ_{RC} is 0.02, τ_{Ref} is 0.002, *Max rate* low is 100 high is 200, *Intercept* is low -1.0 and high is 1.0).

Clicking on *Set* allows the parameters for generating neurons in the population to be configured. Briefly, τ_{RC} is the RC time constant for the neuron membrane, usually 20ms, τ_{Ref} is the absolute refractory period (the period during which a neuron cannot spike after having spiked), *Max rate* has a high and low value, in Hertz, which determines the range of firing rates neurons will have at the extent of the radius (the maximum firing rate for a specific neuron is chosen randomly from a uniform distribution between low and high), *Intercept* is the range of possible values along the represented axis where a neuron ‘turns off’ (for a given neuron its intercept is chosen randomly from a uniform distribution between low and high).

- Click *OK*. Click *OK* again, and the neurons will be created.

If you double-click on the population of neurons, each of the individual cells will be displayed.

- Right-click on the population of neurons, select *Plot*→*Constant Rate Responses*.

The ‘activities’ graph which is now displayed shows the ‘tuning curves’ of all the neurons in the population. This shows that there are both ‘on’ and ‘off’ neurons in the population, that they have different maximum firing rates at $x = \pm 1$, and that there is a range of intercepts between $[-1, 1]$. These are the heterogeneous properties of the neurons that will be used to represent a scalar value.

- Right-click on the population of neurons, select *Plot*→*Plot distortion: X*.

This plot is an overlay of two different plots. The first, in red and blue compares the ideal representation over the range of x (red) to the representation by the population of neurons (blue). If you look closely, you can see that blue does not lie exactly on top of red, though it is close. To emphasize the difference between the

two plots, the green plot is the distortion (i.e. the difference between the ideal and the neural representation). Essentially the green plot is the error in the neural representation, blown up to see its finer structure. At the top of this graph, in the title bar, the error is summarized by the MSE (mean squared error) over the range of x . Importantly, MSE decreases as the square of the number of neurons (so RMSE is proportional to $1/N$), so more neurons will represent x better.

- Right-click on the population and select *Configure*. Any of these properties can be changed for the population.

There are too many properties here to discuss them all. But, for example, if you click on the arrow beside *i neurons*, double-click the current value. You can now set it to a different number and the population will be regenerated with the new number of neurons.

Let us now examine the population running in real time.

- Drag a new *Input* from the template bar into the Scalar Representation network.
- Set *Name* to 'input', make sure *Output Dimensions* is 1 and click *Set Functions*.
- Select *Constant Function* from the drop down menu click *Set* and set *Value* to 0. Click *OK* on all the open configuration windows.

The function input will appear. We will now connect the function input to the neural population as in the tutorial in section 1.5. This essentially means that the output from the function will be injected into the soma of each of the neurons in the population, driving its activity.

- Drag a *Termination* component onto the 'x' population. Set *Name* to 'input', *Weights Input Dim* to 1, and *tauPSC* to 0.02. Click *Set Weights*, double-click the value and set it to 1. Click *OK* twice.
- Click and drag the 'origin' on the input function you created to the 'input' on the 'x' population.
- Click the *Interactive Plots* icon (the double sine wave in the top right corner).

This plot should be familiar from the previous tutorial. It allows us to interactively change the input, and watch various output signals generated by the neurons.

- Right-click ‘x’ and select *value*. Right-click ‘x’ and select *spike raster*. Right-click ‘input’ and select *value*. Right-click ‘input’ and select *control*.

Change the layout to something you prefer by dragging items around. If you would like the layout to be remembered in case you close and re-open these plots, click the small triangle in the middle of the bottom of the window (this expands the view), then click the disk icon under *layout* (if it is not visible you may have to widen the window).

- Click the play button. Grab the control and move it up and down. You will see changes in the neural firing pattern, and the value graphs of the input and the population.

Note that the value graph of the ‘x’ population is the linearly decoded estimate of the input, as per the first principle of the NEF. Note also that the spike raster graph is displaying the encoding of the input signal into spikes. The spiking of only 10% of the neurons are shown by default. To increase this proportion:

- Right-click on the spike raster graph and select a larger percentage of neurons to show.

The population of neurons does a reasonably good (if slightly noisy) job of representing the input. However, neurons can not represent arbitrary values well. To demonstrate this do the following.

- Right-click the control and select *increase range*. Do this again. The range should now be ± 4 (to see the range, hover over the slider).
- Centre the controller on zero. The population should represent zero. Slowly move the controller up, and watch the value graph from the ‘x’ population.

Between 0 and 1, the graph will track the controller motions well. Notice that many of the neurons are firing very quickly at this point. As you move the controller past 1, the neural representation will no longer linearly follow your movement. All the neurons will become ‘saturated,’ that is, firing at their maximum possible rate. As you move the controller past 2 and 3, the decoded value will almost stop moving altogether.

- Move the controller back to zero. Notice that changes around zero cause relatively large changes in the neural activity compared to changes outside of the radius (which is 1).

These effects make it clear why the neurons do a much better job of representing information within the defined radius: changes in the neural activity outside the radius no longer accurately reflect the changes of the input.

Notice that this population, which is representing a scalar value, does not in any way store that value. Instead, the activity of the cells act as a momentary representation of the current value of the incoming signal. That is, the population acts together like a variable, which can take on many values over a certain range. The particular value it takes on is represented by its activity, which is constantly changing over time. This conception of neural representation is very different from that found in many traditional connectionist networks which assume that the activation of a neuron or a population of neurons represents the activation of a specific ‘concept’. Here, the same population of neurons, *differently activated*, can represent different ‘concepts’.⁸ I return to this issue in sections 9.4 and 10.1.1.

Representing a vector

A single scalar value is a simple neural representation, and hence at the bottom of the representational hierarchy. Combining two or more scalars into a representation, and moving up one level in the hierarchy, results in a vector representation. In this tutorial, I consider the case of two-dimensional vector representation, but the ideas naturally generalize to any dimension. Many parts of cortex are best characterized as using vector representations. Most famously, Apostolos Georgopoulos and his colleagues have demonstrated vector representation in motor cortex (Georgopoulos et al., 1984, 1986, 1993).

In their experiments, a monkey moves its arm in a given direction while the activity of a neuron is recorded in motor cortex. The response of a single neuron to forty different movements is shown in figure 2.15. As can be seen from this figure, the neuron is most active for movements in a particular direction. This direction is called the ‘preferred direction’ for the neuron. Georgopoulos’ work has shown that over the population of motor neurons, these preferred directions, captured by unit vectors pointing in that direction, are evenly distributed around the unit circle in the plane of movement(Georgopoulos et al., 1993).

To construct a model of this kind of representation, we can do exactly the same steps as for the scalar representation, but with a two-dimensional representation.

⁸This is true even if we carve the populations of neurons up differently. That is, there is clearly a range of values (perhaps not the whole range in this example) over which exactly the same neurons are active, but different values are represented.

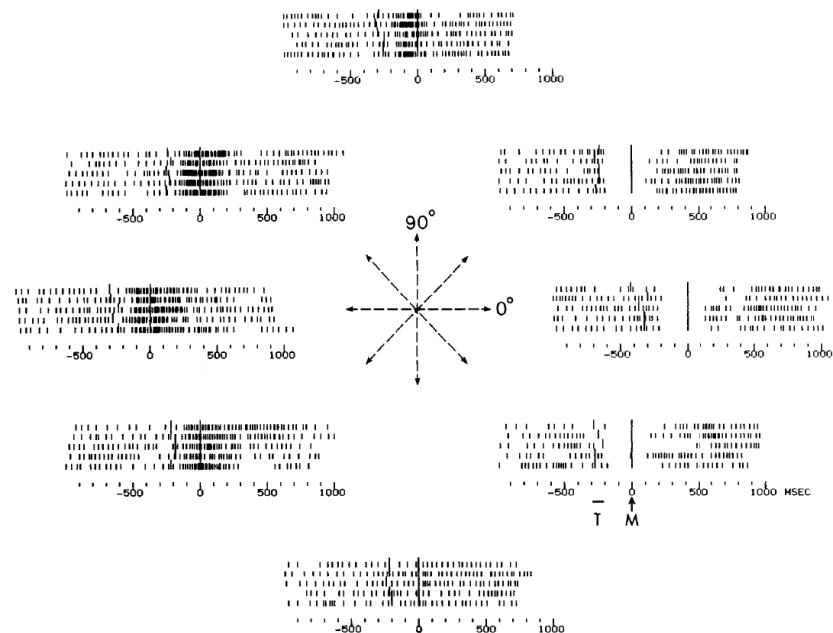


Figure 2.15: The response of a single neuron in motor cortex to different directions of movement. M indicates the time of movement onset. T indicates the time that the target appears. This data shows five trials in each direction for this neuron. The neuron is most active when the target is between 135 and 180 degrees. The direction of the peak of this response is called the neuron's ‘preferred direction’. In the NEF the preferred direction is represented as a unit (i.e., length one) vector pointing in that direction. (Reproduced from Georgopoulos et al. (1986) with permission.)

- In an empty Nengo world, drag a new *Network* into the workspace. Set the *Name* of the network to ‘Vector Representation’ and click *OK*.
- Create a new *Ensemble* within the network. Set *Name* to ‘x’, *Number of nodes* to 100, *Dimensions* to 2, *Node factory* to ‘LIF Neuron’, and *Radius* to 1.
- Click the *Set* button. In the panel that appears, you can leave the defaults (*tauRC* is 0.02, *tauRef* is 0.002, *Max rate* low is 100 high is 200, *Intercept* is low -1.0 and high is 1.0).

The only difference for a vector representation understanding of these parameters is that the *Max rate* and *Intercept* are defined along the preferred direction vector. That is, it is as if each neuron tuning curve is lined up with its particular preferred direction, and then the intercept and maximum rates are applied.

We can think of scalar representation as having only two possible ‘preferred directions’, positive and negative. The preferred direction vector generalizes that notion to higher-dimensional spaces.

- Click *OK* to complete the configuration of the neuron generator and return to the ensemble configuration window. Click *Advanced*. This expands the window. Ensure that the *Encoding Distribution* slider is all the way to the left (over 0.0 - *Evenly Distributed*).

The previous step gives a motor cortex-like distribution of preferred direction vectors.

- If the *Noise* parameter is blank set it to 0.1.

The noise parameter sets the expected amount of noise that Nengo uses to calculate decoder values. Do *not* set this parameter to zero as this would make the ensemble extremely sensitive to noise.

- Click *OK*, and the neurons will be created.

Given these settings, preferred direction vectors will be randomly chosen from an even distribution around the unit n -dimensional hypersphere (i.e. the unit circle in two dimensions). If you plot the constant rate responses, the neuron responses will be plotted along the preferred direction vector of the cell. Consequently, the plot is generated as if all neurons had the same preferred direction vector.

You can now create an input function, and run the network.

- Drag a new *Input* into the Vector Representation network.
- Set *Name* to 'input', make sure *Output Dimensions* is 2 and click *Set Functions*.
- Select *Constant Function* from the drop down menu, click *Set* and set *Value* to 0 for both functions. Click *OK* for each open window.
- Add a *Termination* to the 'x' ensemble. Set *Name* to 'input', *Weights Input Dim* to 2, and *tauPSC* to 0.02. Click *Set Weights* and set the weights to an identity matrix (the top-left and bottom-right values should be set to 1, while the remaining to values should be set to 0).

Notice that you have a matrix of weights to set. This is because the input functions can be projected to either of the dimensions represented by the neural population. For simplicity, we have told the first function to go only to the first dimension and the second function only to the second dimension.

- Click *OK* twice.
- Click and drag the 'origin' on the input function you created to the 'input' on the 'x' population.
- Click on the *Interactive Plots* icon.

We can begin by looking at the analogous displays to the scalar representation.

- Right-click 'x' and select *value*. Right-click 'x' and select *spike raster*. Right-click 'input' and select *value*. Right-click 'input' and select *control*.
- Press play to start the simulation. Move the controls to see the effects on the spike raster (right-click the raster to show a higher percentage of neurons).

You can attempt to find a neuron's preferred direction vector, but it will be difficult because you have to visualize where in the 2D space you are because the *value* plot is over time.

- Right-click the 'input' *value* plot and select *hide*.
- Right-click 'input' and select *XY plot*.

Now you can attempt to determine a neuron's preferred direction vector. This should be easier because you can see the position of the input vector in 2D space. There is a trail to the plot to indicate where it has recently been. The easiest way to estimate a neuron's preferred direction vector is to essentially replicate the Georgopoulos experiments.

- Move the input so both dimensions are approximately zero. Then move one input to its maximum and minimum values.

If a neuron does not respond (or responds minimally), that is not its preferred direction. A neuron(s) whose preferred direction is close to the direction you are changing will respond vigorously to the changes you make. Keep in mind that different neurons have different gains, meaning they may 'ramp' up and down at different rates even with the same preferred direction.

- Right-click the 'x' population and select *preferred directions* to show the neuron activity plotted along their preferred directions in the 2D space.

This plot multiplies the spike activity of the neuron with its preferred direction vector. So, the longer lines are the preferred directions of the most active neurons.

- Put one input at an extreme value, and slowly move the other input between extremes.

It should be clear that something like the average activity of the population of neurons moves with the input. If you take the mean of the input (a straight line through the middle of the blob of activity), it will give you an estimate of the input value. That estimate is something like the linear decoding for a vector space as defined in the first NEF principle (although it is not optimal, as in the principle).

- Right-click the 'x' population and select XY plot. This shows the actual decoded value in 2D space.

Another view of the activity of the neurons can be given by looking at the neurons plotted in a pseudo-cortical sheet.

- Right-click the 'x' population and select *voltage grid*.

This graph shows the subthreshold voltage to the neurons in the population in gray. Yellow boxes indicate that a spike is fired.

- Right-click on the voltage grid and select *improve layout* to organize the neurons so that ones with similar preferred direction vectors will be near each other, as in motor cortex.
- Move the sliders and observe the change in firing pattern.

Using the same kind of exploration of inputs as before, it is reasonably evident in this view which parts of the grid have which preferred direction vectors. This network reproduces the classic view of how motor cortex was thought to encode a movement.

Recent work has challenged the idea that there is such a clean mapping between neural activity and target location in general (Churchland et al., 2010). However, the same approach to vector representation can be used to capture these subtleties Dewolf and Eliamith. As a result, this network provides a useful introduction to vector representation in cortex.

It is important to keep in mind that some aspects of this tutorial on vector representation are specific to this motor cortical example. For instance, using an even distribution of preferred direction vectors, using low-dimensional vectors, and thinking of preferred directions as related to actual movement direction are appropriate because we are considering motor cortex. However, many aspects are more general, including the identification of (abstract) preferred direction vectors and the use of a linear decoding to get an estimate of the population representation.

It is worth highlighting that the (2D) ‘represented vectors’ in this characterization of neural function are *not* the same as the ‘activity vectors’ commonly discussed in artificial neural networks. Activity vectors are typically a set of neuron firing rates. If we have three neurons in our population, and they are active at 50, 100, and 20 Hz respectively, the 3D activity vector would be [50, 100, 20]. This vector defines a point in a space where each axis is associated with a specific neuron. In the example above, the activity vector would be in a 100D space because there are 100 neurons in the population. In contrast, the 2D space represented above has axes determined by an externally measured variable.⁹

Recall from section 2.3.1 that the 2D space in this example is the ‘state space’, whereas the 100D space is the ‘neuron space’. Consequently, we can think of the 2D state space as a standard Cartesian space, where two values (x and y coordinates) uniquely specify a single object as compactly as possible. In contrast, the 100D vector specifies the same underlying 2D object, but it takes many more

⁹These axes do not need to be externally measurable, as we shall see. However, they are in many of the simplest cases of neural representation.

resources (i.e. values) to do so. If there was no uncertainty in any of these 100 values this would be a simple waste of resources. However, in the much more realistic situation where there is uncertainty (due to noise of receptors, or noise in the channels sending the signals, etc.) this redundancy can make specifying that underlying object much more reliable. And, interestingly, it can make the system much more flexible in how well it represents different parts of that space. For example, we could use 10 of those neurons to represent the first dimension, or we could use 50 neurons to do so. The second option would give a much more accurate representation of that dimension than the first. Being able to redistribute these resources to respond to task demands is one of the foundations of learning (see section 6.4).

More extensive tutorials on neural representation are available on the CNRG website at <http://compneuro.uwaterloo.ca/cnrglab/?q=node/2>.

Chapter 3

Biological cognition – semantics

In reading what follows, it is important to keep in mind that the ideas captured here represent the beginning, not the end, of a research program. For this reason, it is perhaps worth stating what the following architecture – the Semantic Pointer Architecture (SPA) – is *not*. First, it is not testable by a single, or small set, of experiments. Being the foundation of a research program, the SPA gains some amount of credibility when it gives rise to successful models. Failures of such models lead to a reconsideration of those specific models and, when systematic, a reconsideration of their foundations.

Second, the SPA is not a completed theory of mental function: we have not yet actually built a fully cognitive brain (you will not be surprised to learn). In what follows I describe models of perception, action, and cognition. And, I describe these in a way that combining them is reasonably straightforward (an example of their combination is found in chapter 7). Nevertheless, there are many more behaviors involving these aspects cognition that I do not discuss.

Third, even theoretically speaking, the coverage of the SPA is uneven. Some aspects of cognition are more directly addressed than others – the SPA is undeniably a work in progress. Perhaps most obviously, the SPA has little to say about the development of cognitive systems either evolutionarily or over the course of a single life span. It is focused instead on characterizing already mature cognitive systems (i.e., “typical adult human cognizers”). This focus is shared with most, though not all, current cognitive architectures.

These qualifications aside, there are still compelling reasons to pursue the SPA. First, while it is *not* a completed, unified theory of mental function, it *is* an attempt to move towards such a theory. Such attempts can be useful in both their successes and failures: either way, we learn about constructing a theory of

this kind. As discussed earlier, some have suggested that such a theory does not exist (section 1.2). But the behavioral sciences are far too young to think we have done anything other than scratch the surface of possible cognitive theories. And, as I argue in detail in chapters 8 and 9, I believe the SPA moves beyond currently available alternatives.

A second, related reason to pursue the SPA is its close connection to biological considerations. A main goal of this work is to show how we can begin to take biological detail seriously, even when considering sophisticated, cognitive behavior. Even if it is obvious that we should use as much available empirical data as possible to constrain our theories in general, actually doing so requires well-specified methods that contact available data as directly as possible. In what follows, I describe specific applications of the SPA which draw on, relate to, and predict empirical data in new and interesting ways (see e.g., sections 3.5, 4.6, 5.7, 6.2, etc.).

A third reason to pursue the SPA is its generality. I have attempted to demonstrate the generality of the approach by choosing a broad set of relevant examples which demonstrate, but do not exhaust, the principles at work. Of course, one book is not enough to adequately address even a small portion of cognitive behavior. In short, the intent is to provide a method and an architecture that opens the way for a much wider variety of work than can possibly be captured in a single book, or for that matter, done in a single lab.

I leave a more detailed discussion of the consequences of adopting this architecture for chapters 9 and 10. Nevertheless, keeping in mind the main motivations behind the SPA should help situate the following introduction to it.

In the next four chapters, I describe and demonstrate central aspects of the Semantic Pointer Architecture (SPA): semantics, syntax, control, and memory and learning. In the chapter after that, I provide an integration of these aspects of cognition in the form of a general theoretical characterization of the Semantic Pointer Architecture, and an integrated model that addresses many features of biological cognition at once. I then explicitly compare the SPA to past suggestions for cognitive architectures, and discuss important practical and conceptual differences.

3.1 The semantic pointer hypothesis

Underlying the Semantic Pointer Architecture (SPA) is the semantic pointer hypothesis. Its purpose is to bridge the gap between the neural engineering framework (NEF) – a theory that tells us how a wide variety of functions may be imple-

mented in neural structures – and the domain of cognition – which is in need of ideas about how such neural structures give rise to complex behavior.

Here is a simple statement of the semantic pointer hypothesis:

Higher-level cognitive functions in biological systems are made possible by semantic pointers. Semantic pointers are neural representations that carry partial semantic content and are composable into the representational structures necessary to support complex cognition.

There are two aspects of this hypothesis that must be expanded in detail. First, I must indicate how semantic information, even if partial, will be captured by the representations that we choose to identify as semantic pointers. Second, I must give a characterization of how to construct “complex representational structures” given those same representations. Given the characterization of neural representation in the previous chapter, it is natural to begin with the claim that semantic pointers are vectors in a high-dimensional space. In the remainder of this chapter I address the semantics of these vector representations. In the next chapter, I discuss the construction of representational structures out of these vectors that support syntactic manipulation.

There have been a wide variety of proposals regarding the role that semantics and syntax should play in our theories of cognition. Traditionally, a classical cognitive system is characterized as largely a symbol processing system that relies on syntax to respect the semantics of the symbols and combinations of symbols found in the system (see, e.g., Fodor, 1975). So language-like syntax was thought to do most of the cognitive work, and semantics came along for the ride. In more connectionist approaches, semantics has been the focus, where the vector space defined by the activity of the nodes in the network was often thought of as a “semantic space” that related the meaning of different firing patterns (see, e.g., Churchland, 1979). In the SPA, syntax is inspired by some symbolicist observations (e.g., that there are syntactically structured representations in the head), and semantics is inspired by some connectionist observations (e.g., that vector spaces can be used to capture important features of semantics), and both are implemented in the same biologically plausible substrate.

So, considering semantics for the time being, it is a well-worn claim that a high-dimensional vector space is a natural way to represent semantic relationships between representations in a cognitive system. It is difficult to visualize high-dimensional spaces, but we can get a sense of what this claim amounts to by thinking of concepts in a 3-dimensional state space (a relatively low-dimensional space), as shown in figure 3.1. I refer to such a visualization as a “conceptual golf

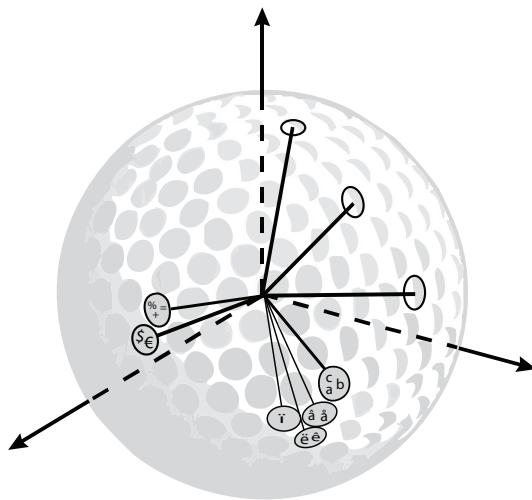


Figure 3.1: A “conceptual golf ball” depicting a 3D state space. Each dimple in the ball represents a concept, visualized here as a set of specific example symbols inside a circle. Vectors from the center of the sphere to the dimples on its surface are used to identify these concepts. In the figure concepts for letters are clustered together while other kinds of symbols (e.g. \$, %) are clustered separately; within the cluster of dimples storing letters, the letters with special accents are also grouped more closely together. The location of a concept (or specific example) on the surface of the conceptual golf ball thus carries information about its relationship to other concepts (and/or specific example).

ball”. The surface of the ball represents the conceptual space, and the dimples in the surface represent concepts. Specific examples of a concept are picked out by points that lie within a given dimple. Concepts (and examples) that are semantically similar lie in relatively close proximity compared to concepts that are semantically dissimilar.

Unfortunately, things will get quickly crowded in 3 dimensions, and so neighboring concepts will be easily confused. This is especially true if there is uncertainty, or noise, in the specification of a given example. Since the presence of noise should be expected, especially in a real physical system like the brain, the number of distinct concepts that can be represented effectively may be quite small in low-dimensional spaces. The important contribution of high-dimensionality is that the amount of surface area available to put concepts in increases incredibly quickly with dimensionality, as shown in figure 3.2. As a result, many concepts,

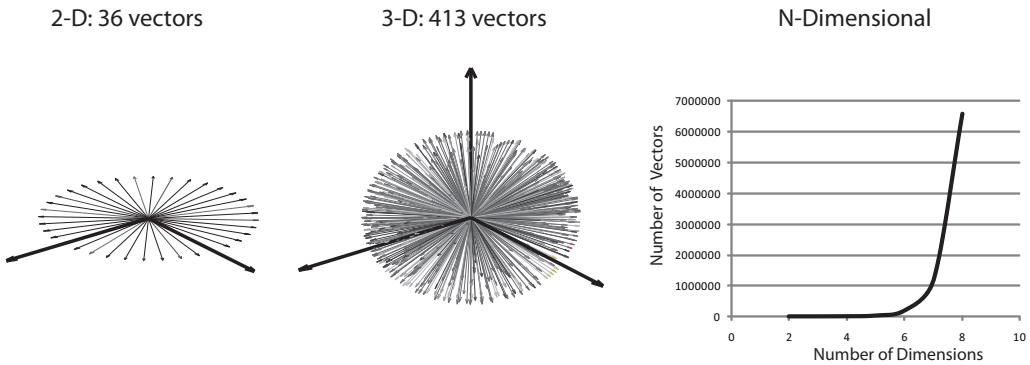


Figure 3.2: The scaling of the available conceptual space on the surface of a hypersphere with dimensionality. The figure shows the number of unit vectors that can be placed into a disk and ball under the constraint that there must be a minimum angle of ten degrees between any two vectors. The rightmost plot extends the result from intuitive two and three dimensional shapes to higher dimensional spheres.

even under noise, can be represented within a high-dimensional space.

An obvious drawback with such spaces is that it is not immediately clear how complex representational *structures* can be placed in such a space (since the space has no obvious structure to it): I return to this issue in the next chapter. However, even without worrying about representational structure *per se*, the semantics of even simple human concepts seems to be extraordinarily rich, extending beyond a simple mapping to a single, even very high-dimensional, vector space (Barsalou, 2009, 1999). The richness of human conceptual behavior suggests that it is unlikely that all of the aspects of a concept are actively represented at the same time. In earlier work, I have made a distinction between “conceptual” and “occurrent” representation (Eliasmith, 2000). Conceptual representations are representations that address the traditional questions about concepts that have preoccupied cognitive scientists and their predecessors for centuries. Occurrent representations are those which the present activity of a brain instantiates (i.e. representations as described by the NEF). Obviously the two are closely related. It is not far off to think of occurrent representations as being the currently active “parts” of complete conceptual representations.

With this distinction in hand, and acknowledging the richness of conceptual behavior, we are lead to the realization that it is a mistake to assume that there are occurrent representations in the brain that carry the full semantic content of

any given concept. There is simply too much information related to any particular conceptual representation to be able to actively represent and manipulate it all at the same time. This is why the Semantic Pointer Architecture employs the notion of a “pointer”.

A pointer, in computer science, is a set of numbers that indicate the address of a piece of information stored somewhere in memory. What is interesting about pointers, is that manipulations of pointers themselves can be performed which result in an indirect use of the information identified by the pointer, despite the fact that that information itself is never explicitly addressed. Most such manipulations are quite simple. For instance, if I am creating a list of data that needs to be grouped, I can store each piece of data with a pointer to the next data item in the list. Such a “linked list” provides a flexible method for traversing, removing, and adding items to a collection of data. Similarly, if I need to pass a data structure to a function which may use it, I can simply pass the pointer, which is typically much smaller than the data structure itself. As a result, much less information needs to be moved around within the system, while still making the relevant data available for subsequent use.

One notable feature of pointers in computer science, is that a pointer itself and the information contained at its address are arbitrarily related. As a result, having the pointer itself often indicates nothing about what sort of information will be found when the address to which it points is accessed. Since decisions about how to use information often depend on what the content of that information is, pointers are often “dereferenced” during program execution. Dereferencing occurs when the data at the address specified by the pointer is accessed. In a computer, this is a relatively cheap operation because memory is highly structured, and the pointer is easy to interpret as an address.

Given these features, pointers are reminiscent of symbols. Symbols, after all, are supposed to gain their computational utility from the arbitrary relationship they hold with their contents (Fodor, 1998). And symbols are often thought to act like labels for more sophisticated data structures (such as schemas, scripts, etc.), just as pointers act as labels for whatever happens to be at their address.

The semantic pointer hypothesis suggests that neural representations, especially those implicated in cognitive processing, share central features with this traditional notion of a pointer. In short, the hypothesis suggests that the brain manipulates compact, address-like representations to take advantage of the significant efficiency and flexibility afforded by such representations. Relatedly, such neural representations may be able to act like symbols in the brain.

However, the arbitrary relationship between a pointer and its contents seems

problematic for biological systems, because relationships between neural representations are most often learned. In contrast, the relationships between symbols in a digital computer are determined by human design decisions. It is precisely the failure of such decisions to adequately capture semantics that raises what has become known as the “symbol grounding problem” (Harnad, 1990). This, in short, is the problem of defining how symbols get their semantics – a difficult problem indeed, and one over which much ink has been spilled. As a result, it would be too hasty to simply import the notion of a pointer directly from computer science into our understanding of biological cognition.

Consequently, the hypothesis I am suggesting here is an extension of the standard notion of a pointer. In particular, the “semantic” in “semantic pointer” refers to the fact that the representations that play the role of a pointer contain semantic information themselves. That is, the relationship between a semantic pointer and the memory to which it points is not arbitrary. In section 3.5, I provide a detailed example in the visual system of how semantic pointers relate to the memories to which they point. For now, it is useful to think of the semantic information that is contained in a semantic pointer as a compressed version of the information contained in a more conceptual memory.

3.2 What is a semantic pointer?

In the preceding discussion of the semantic pointer hypothesis, I characterize semantic pointers in three different ways. First, I characterize them functionally: semantic pointers are compressed representations that point to fuller semantic content, and they can be composed into complex representational structures to support cognition. Second, I characterize them computationally: semantic pointers are vectors in a high-dimensional state space. And third, I characterize them mechanistically: semantic pointers are occurrent activity in a biological neural network.

To be clear, I take these characterizations to be different characterizations of the same thing. In the previous chapter, I described how vectors in a high-dimensional space can be instantiated by neural activities in a biologically plausible network. In a way, the central purpose of the NEF’s representational principle is to demonstrate how the second and third characterizations of semantic pointers are related (that is, the inter-level relation between a vector space and neural activities). The functional characterization goes beyond the NEF to specify what role semantic pointers play in the brain. In short, it suggests how those vectors

(implemented in neurons) are used to underwrite biological cognition. As a result, it partially specifies inter-level relations that relate simple units (e.g., light intensity) to complex units (e.g., object identity) across levels of description. A clear example of this can be found in section 3.5.

Each of these characterizations has a different explanatory role to play, but all are descriptions of the same “thing”. This is a common feature of scientific concepts. For instance, we can talk about an electrical current in terms of ion movements if we wish to explain the implementation of the current. We can talk about the current as being a specific number of amps (i.e., characterize it as a one-dimensional vector) if we want to talk about its role in Kirchoff’s current law (i.e., that the sum of currents flowing into and out of a point are equal). And, we can talk about a current causing a light to illuminate if we want to explore how it is functioning in a given circuit.

I should point out that this tripartite distinction is not the same as that proposed by Marr (1982). His was a distinction to describe processes at different levels of abstraction. Thus, all three levels were variously detailed descriptions of the functional (i.e., Marr’s “computational”) specification. While it may be possible, and even useful, to talk of the SPA in this way, semantic pointers themselves are intended to be single, unified objects which can be talked about in several related ways: functionally, computationally, and mechanistically. Throughout the book, I will move between these ways of talking about semantic pointers. But, in every case, I take it as evident how that way of talking could be “translated” into any of the other ways of describing semantic pointers.

With the previous rough characterization of semantic pointers, and this terminological clarification in hand, we are in a position to consider recent research on semantics and examine detailed examples of how motor and perceptual semantic pointers function in a neural architecture.

3.3 Semantics: An overview

There has been an historical tendency to start cognitive architectures with syntactic considerations and build on them by introducing more and more sophisticated semantics. However, psychologists and linguists have more recently argued that much syntactic processing can be best explained by sophisticated semantic processing.¹ In contrast to traditional approaches, this puts semantics in the driver’s

¹Research in the field of “cognitive linguistics” largely adopts this stance (e.g., Lakoff, 1987; Langacker, 1986; Fillmore, 1975), as do a number of psychologists (e.g., Johnson-Laird, 1983;

seat. As a result, much recent empirical work on semantics has focused on understanding what *kind* of semantic processing is needed to perform various cognitive tasks. In short, the question of interest has become: for which behaviors do we need “deep” semantic processing, and which can be effectively accounted for by “shallow” semantic processing?

The distinction between deep and shallow processing can be traced back to Allan Paivio’s (1986; 1971) Dual-Coding Theory. This theory suggests that perceptual and verbal information are processed in distinct channels. In Paivio’s theory, linguistic processing is done using a symbolic code, and perceptual processing is done using an analog code, which retains the perceptual features of a stimulus. Paivio (1986) provides a lengthy account of the many sources of empirical evidence in support of this theory, which has been influential in much of cognitive psychology, including work on working memory, reading, and human computer interface design.

More recently, this theory has been slightly modified to include the observation that both channels are not always necessary for explaining human performance on certain tasks (Simmons et al., 2008; Glaser, 1992). Specifically, simple lexical decision tasks do not seem to engage the perceptual pathway. Two recent experiments help demonstrate this conclusion. First, Solomon and Barsalou (2004) behaviorally demonstrated that careful pairings of target words and properties can result in significant differences in response times to determining if a property belongs to a target word. For instance, when subjects were asked to determine if the second word in a pair was a property of the first word, false pairings that were lexically associated took longer to process. For example, a pair like “cherry-card” resulted in 100ms quicker responses than a pair like “banana-monkey”. Second, Kan et al. (2003) observed that fMRI activation in perceptual systems was only present in the difficult cases for such tasks. Together, this work suggests that deep processing is not needed when a simple word association strategy is sufficient to complete the task.

Nevertheless, much of the semantic processing we perform on a daily basis seems to be of the “deep” type. Typical deep semantic processing occurs when we understand language in a way that would allow us to paraphrase its meaning, or answer probing questions about its content. It has been shown, for instance, that when professional athletes, such as hockey players, read stories about their sport, the portions of their brain that are involved in generating the motor actions associated with that sport are often active (Barsalou, 2009). This suggests

Barsalou, 1999).

that deep semantic processing may engage a kind of “simulation” of the circumstances described by the linguistic information. Similarly, when people are asked to think and reason about objects (such as a watermelon), they do not merely activate words that are associated with watermelons, but seem to implicitly activate representations that are typical of watermelon backgrounds, bring up emotional associations with watermelons, and activate tactile, auditory, and visual representations of watermelons (Barsalou, 2009).

Consider the Simmons et al. (2008) experiment which was aimed at demonstrating both the timing and relative functions of deep and shallow processing. In this experiment participants were each scanned in an fMRI machine twice. In one session, the experimenters were interested in determining the parts of the brain used during shallow semantic tasks and during deep semantic tasks. As a result, participants were asked two questions: first, “For the following word, what other words come to mind immediately?”; and second, “For the following word, imagine a situation that contains what the word means and then describe it?” The experimenters found that in response to the first question, language areas (such as Broca’s area) are most active. In contrast, in response to the second question participants engaged brain areas that are active during mental imagery, episodic memory, and situational context tasks (Kosslyn et al., 2000; Buckner and Wheeler, 2001; Barr, 2004). In other words, from this session it was evident that simple lexical association activated language areas, whereas complex meaning processing activated perceptual areas, as would be expected from the Dual-Coding Theory.

During the other scanning session, participants were asked to list, in their heads, answers to the question “what characteristics are typically true of X?”, where X was randomly chosen from the same set of target words as in the first session. When the two different scanning sessions were compared, the experimenters were able to deduce the timing of the activation of these two different areas. They found that the first half of the “typically true of X” task was dominated by activation in language areas, whereas the second half of the task was dominated by activation in perceptual areas. Consistent with the earlier behavioural experiments, this work shows that shallow processing is much more rapid, so it is not surprising that highly statistically related properties are listed first. Deep processing takes longer, but provides for a richer characterization of the meaning of the concept.

This experiment, and many others emphasizing the importance and nature of deep semantic processing, have been carried out in Larry Barsalou’s lab at Emory University. For the last two decades, he has suggested that his notion of “perceptual symbols” best characterizes the representational substrate of human cogni-

tion. He has suggested that the semantics of such symbols are captured by what he calls “simulations”. Indeed, the notion of a “simulation” has often been linked to ideas of deep semantic processing (e.g., Allport, 1985; Damasio, 1989; Pulvermüller, 1999; Martin, 2007). Consequently, they would no doubt agree with Barsalou’s claim that deep semantic processing occurs when “the brain simulates the perceptual, motor, and mental states active during actual interactions with the word’s referents” (Simmons et al., 2008, p. 107). Indeed, his data and arguments are compelling.

However, the important missing component of his theory is how such symbols and simulations can be implemented and manipulated by the brain. In a discussion of his work in 1999, one recurring critique was that his notion of “perceptual symbols” is highly under-defined. For instance, Dennett and Viger pointedly note “If ever a theory cried out for a computational model, it is here” (1999, p. 613). More to the point, they conclude their discussion in the following manner:

We want to stress, finally, that we think Barsalou offers some very promising sketchy ideas about how the new embodied cognition approach might begin to address the “classical” problems of propositions and concepts. In particular, he found some novel ways of exposing the tension between a neural structure’s carrying specific information about the environment and its playing the sorts of functional roles that symbols play in a representational system. Resolving that tension in a working model, however, remains a job for another day.

Indeed, in the conclusion to a recent review of his own and others’ work on the issue of semantic processing, Barsalou states (Barsalou, 2009, p. 1287):

Perhaps the most pressing issue surrounding this area of work is the lack of well-specified computational accounts. Our understanding of simulators, simulations, situated conceptualizations and pattern completion inference would be much deeper if computational accounts specified the underlying mechanisms. Increasingly, grounding such accounts in neural mechanisms is obviously important.

This, of course, is the purpose of the Semantic Pointer Architecture: to provide a neurally grounded account of the computational processes underwriting cognitive function.

Given the important distinction between deep and shallow processing, and the evidence that these are processed in different ways in the brain, a central question

for the SPA is: how can we incorporate both deep and shallow processing? The central hypothesis of the SPA outlines the answer I will pursue in the next three sections: that semantic pointers carry partial semantic information. The crucial steps to take now are to: 1) describe exactly how the partial semantic information carried by semantic pointers is generated (and how they capture shallow semantics); and 2) describe how semantic pointers can be used to access the deep semantics to which they are related (i.e., how to dereference the pointers).

3.4 Shallow semantics

In essence, the shallow semantics captured by a semantic pointer can be thought of as a kind of “compressed” representation of complex relations that underly deep semantics. Compression comes in two forms: “lossless” like the well known .zip methods used to compress computer files that need to be perfectly reconstructed; and “lossy”, such as the well-known .jpg method for compressing images, and which loses some of the information in the original image that was compressed. I take semantic pointers to be lossy compressions of the information they are generated from. To demonstrate the utility of lossy compression, and its relevance to cognition, let us consider a recently proposed class of lexical semantic representations. These representations have been developed by researchers who build algorithms to do automatic text processing. These same representations have been shown to capture many of the word-similarity effects that have been extensively studied by psychologists (Deerwester et al., 1990a; Landauer and Dumais, 1997).

These representations are constructed by having a computer process a very large corpus of example texts. During this processing, the computer constructs what is called a term-document frequency matrix (see figure 3.3). The columns of this matrix index specific documents in the corpus. The rows in the matrix index words that appear in those documents. When the computer reads a particular document, it counts the number of times any word occurs in the document, and adds that value to the appropriate cell of the matrix. This way, if we look down the columns of the matrix we can determine how many times each word appears in a given document. If we look across the rows of the matrix, we can determine how many times each word appears in each document.

Practically speaking, there are a number of important subtleties to consider when constructing these kinds of representations to do actual textual processing. For instance, such matrices tend to get very large, since many standard corpora have over 20,000 documents and 100,000 unique words (Fishbein, 2008). Conse-

	Moby Dick	Hamlet	Alice in Wonderland	Sense and Sensibility
boat	330	0	0	0
enemy	5	0	0	1
mad	36	17	14	0
manners	1	1	1	34
today	1	0	1	9
tomorrow	1	0	1	15

Figure 3.3: Term-document frequency matrix. Each row of the matrix shows the number of occurrences of a specific word, written in the first column, in the four selected texts specified by the other columns. An extremely crude classification system based on this matrix might categorize Moby Dick, Hamlet, and Alice in Wonderland together based on their themes of madness, while distinguishing Moby Dick based on its abnormally frequent reference to boats. The row vectors of the words “today” and “tomorrow” are also noteworthy for being quite similar, suggesting some level of semantic similarity between the words. In most applications, such a matrix is compressed before it is used.

quently, methods for reducing the size of the matrix are often employed. These methods are chosen to ensure that the most important statistical relationships captured by the original matrix are emphasized as much as possible. This, then is a kind of lossy compression applied to the original raw matrix. One of the best known compression methods is singular-value decomposition, which is used, for example, in the well-known latent semantic analysis (LSA) approach (Deerwester et al., 1990b). For explanatory purposes, however, I will simply consider the raw matrix shown in figure 3.3.

The reason such a matrix can capture semantic information is that we expect semantically similar words to occur in the same documents. This is because most of the considered documents in these computational experiments are short, and on one or a few basic themes – like the books and stories encountered in early childhood. So, if we compare the row-vectors of semantically similar words, we expect them to have similar vectors, because those words will appear in many of the same contexts. Semantically unrelated words will have vectors that are not very similar. Notice that like semantic pointers, these representations of words are first and foremost compressed high-dimensional vectors.

While simple, this method is surprisingly effective at categorizing documents.

Using a wide variety of text corpora, well above 90% of new documents can be correctly classified with such representations (Yang and Liu, 1999). More importantly, this same kind of representation has been used by researchers to capture psychological properties of language. For example, Paafset al. (2004) demonstrate how prototypes² can be extracted using such a method. The resulting representations capture standard typicality effects.³ As well, this kind of representation has been used to write the Test of English as a Foreign Language (TOEFL). The computer program employing these representations scored 64.4%, which compares favorably to foreign applicants to American universities, who scored 64.5% on average (Landauer and Dumais, 1997).

However, these representations of words clearly only capture shallow semantics. After all, the word representations generated in this manner bear no relation to the actual objects that those same words pick out: instead, they model the statistics of text. Nevertheless, this work shows that semantic representations which capture basic statistics of word use can effectively support certain kinds of linguistic processing observed in human subjects. As mentioned earlier, these representations are compressed high-dimensional vectors, just like semantic pointers. That is, they capture precisely the kind of shallow semantics that semantic pointers are proposed to capture in the SPA. In short, the SPA suggests that shallow semantic processing can be performed without using the pointer as a pointer: i.e., by relying solely on the semantic content captured by the pointer representation itself without dereferencing it.

However, consideration of the kinds of psychological experiments discussed in the last section suggests that there is an important and distinct role for deep semantic processing. This role is clearly not captured by the kinds of simple lexical associations used to generate these shallow representations for automatic text categorization. And, these shallow semantics are, in general, not sufficient to address the symbol grounding problem identified in section 3.1. There is, after all, no way to get back to a “richer” representation of a word from these term-document representations. To address both deep semantics and symbol grounding, in the next section I turn to a different, though still statistical, method of generating

²Prototypes of categories are often used to explain the nature of concepts (Smith, 1989). It has been a matter of some debate how such prototypes can be generated.

³Typicality effects are used to explain why subjects rate some concept instances as being more typical than others. These effects are often explained by the number of typical features that such instances have (the more typical features an instance has, the more typical it will be). Typical instances are both categorized more quickly and produced more readily by subjects. The prototype theory of concepts has been successful at capturing many of these effects.

semantic pointers, one connected more directly to biological mechanisms.

3.5 Deep semantics for perception

It should be clear from the preceding discussion that there are two questions of interest when considering the Semantic Pointer Architecture. First, how are the semantics of the pointers generated? Second, how can the pointers be used to engage the deep semantic processing system when appropriate? The discussion in the previous section addresses only the first question. This is useful for demonstrating how shallow semantics can be captured by a semantic pointer, in the undeniably cognitive task of language processing.

In this section, I pursue another method for generating shallow semantics that demonstrates how the resulting representation remains linked to deep semantics in a neural architecture. However, the task I consider is more perceptual. In section 4.7, I describe how these (and other) different methods of generating and employing semantic pointers can be integrated.

Let us begin with object recognition in vision. Many of the most impressive results in machine vision employ *statistical modeling* methods. It is important to note that the word “model” in statistics – and in the next few paragraphs – is not used in the same way as it is throughout most of this book, and in most of cognitive science. In statistics, the term “model” refers to an equation that captures relationships: there is no expectation that the elements of the equation pick out objects in the world. In contrast, “model” in non-statistical usages typically refers to abstractions whose parts are expected to map onto objects in the world. The neural models I describe throughout take their abstract parts (neurons, brain areas, etc.) to map onto real parts of the brain. These latter models are sometimes called “mechanistic” models, to distinguish them from statistical models.

In general, statistical modeling methods are centrally concerned with characterizing the measured state of the world, and identifying important patterns in those often noisy, measured states. In short, these methods have been developed to describe complex relationships given real-world data. This should sound familiar: the lexical representations described in the previous section are a kind of statistical model, attempting to describe the relationships between words in real-world text data (in which there are many spelling errors, non-words, etc.). Considered generally, describing complex real-world relationships with uncertain information is exactly the problem faced by biological systems.

For objection recognition, we can begin to formalize this problem by suppos-

ing there is some visual data (e.g., an image), y , which is generated by the external visual world and drives neural activity. If we suppose that the purpose of perceptual systems is to construct and use a statistical model of this data, the system must figure out some function $p(y)$ that describes how likely each state y is, so it can use that information to disambiguate future data. For instance, if I am in North America and a medium-sized brown animal is coming in my direction, I can assign probabilities to the various kinds of animal it might be (e.g. groundhog, dog, and rabbit are high, capybara and wallaby are low). Assigning those probabilities is an example of using the model, $p(y)$, that I have constructed based on past data.

Since the real world is extremely complex, the ideal statistical model will also be enormously complex (as it is the probability of all possible data at all times). As a result, the brain probably approximates this distribution by constructing what is called a *parameterized* model. Such a model identifies a small number of parameters that capture the overall shape of the ideal model. For example, if the statistics of the data y lie in the famous Bell curve (or Gaussian) distribution, we can model the data with an equation like:

$$p(y) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(y-\bar{y})^2/2\sigma^2}$$

Then, to “capture” all past data using our model, we only need to remember two parameters, \bar{y} (the mean) and σ (the standard deviation), and the equation describing their relationship. This is much more efficient than remembering each value of $p(y)$ for each value of y explicitly.

To build such a model, the system needs to estimate the parameters (the mean and standard deviation in this case). Of course, to do any such estimating, the system needs data. As a result, a kind of bootstrapping process is necessary to construct this kind of model: we need data to estimate the parameters; then use our best estimate of the parameters to interpret any new data. Despite this seeming circularity, extremely powerful and general algorithms have been designed for estimating exactly these kinds of models (Dempster et al., 1977). Such methods have been extensively employed in building connectionist-type models, and have been suggested to map to biological neural networks.⁴

Note, however, that the methods for model inference do not specify the structure of the model itself (i.e. the relationships between the parameters). In the

⁴A good place to start for state-of-the-art applications of these methods is Geoff Hinton’s web page at <http://www.cs.toronto.edu/~hinton/>. For discussion of biological mappings of these methods see Friston (2003).

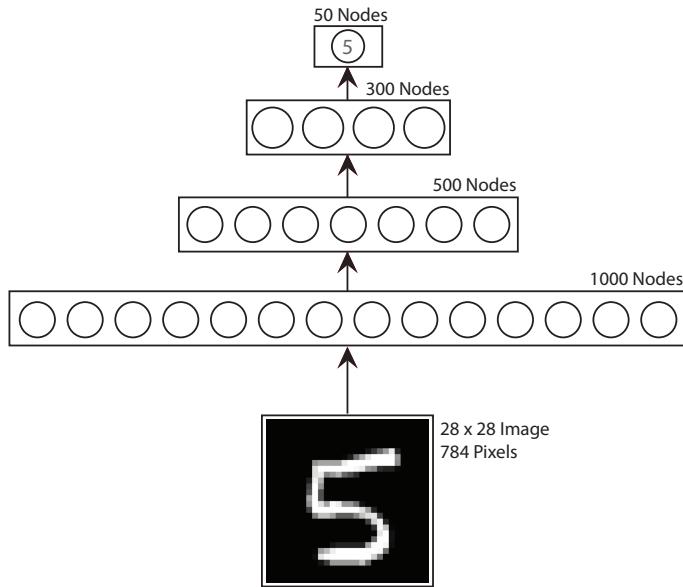


Figure 3.4: A hierarchical statistical model. An original image consisting of 784 pixels is compressed into a 50-dimensional compressed representation through a hierarchical series of statistical summaries. Note that the number of nodes at each level defines the dimensionality of the state space.

artificial neural network application of these methods, this structure is often “biologically inspired”. One notable feature of brain structure that has proven a very useful starting point to constrain such models, is its hierarchical nature. The best known example of this structure in neuroscience is the visual hierarchy. For object recognition, this hierarchy begins with the retina, and proceeds through thalamus to visual areas V1 (primary visual cortex), V2 (secondary visual cortex), V4, and IT (inferotemporal cortex) (Felleman and Essen, 1991).

In a hierarchical statistical model, each higher level in the hierarchy attempts to build a statistical model of the level below it. Taken together, the levels define a model of the original input data (see figure 3.4). This kind of hierarchical structure naturally allows the progressive generation of more complex features at higher levels, and progressively captures higher-order relationships in the data. Furthermore, these kinds of model lead to relations between hierarchical levels that are reminiscent of the variety of neural connectivity observed in cortex: feed-forward, feedback, and recurrent (interlayer) connections are all essential.

The power of these methods for generating effective statistical models is im-

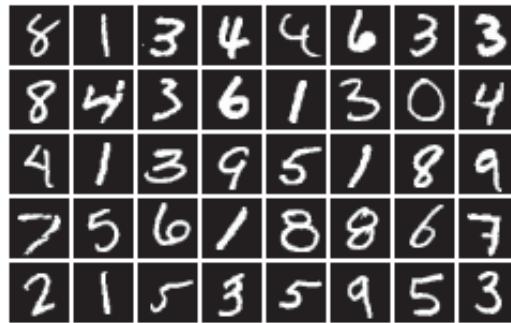


Figure 3.5: Input images from the MNIST database of handwritten digits.

pressive (Beal, 1998). They have been applied to solve a number of standard pattern recognition problems, improving on other state-of-the-art methods (Hinton and Salakhutdinov, 2006). Furthermore, they have been shown to generate sensitivities to the input data that look like the tuning curves seen in visual cortex (Lee et al., 2007), when constructing models of natural images. In fact, many of the most actively researched models of vision are naturally interpreted as constructing exactly these kinds of hierarchical statistical models.

To get a clearer picture of what this approach to perceptual modeling offers, and how it can be used to generate semantic pointers, let us turn to an example of such a model that was built in my lab by Charlie Tang. The purpose of this system is to construct representations which support recognition of handwritten digits presented as visual input. The input is taken from the commonly used MNIST database at <http://yann.lecun.com/exdb/mnist/>. Examples of the input are shown in figure 3.5. The model is structured as shown in figure 3.4, and is shown 60,000 examples out of this data set, and told how those examples should be categorized. Based on this experience, the model tunes its parameters to be able to deal with another 10,000 unseen, though similar, visual inputs in the data set.

To maintain biological relevance, the first layer of the model is trained on natural images, in order to construct an input representation that looks like that found in primary visual cortex. As shown in figure 3.6, the tuning curves capture many of the properties of V1 tuning, including a variety of spatial frequencies (i.e. narrowness of the banding), positions, orientations of the bands, and the edge-detector-like shape.⁵ More direct matches to biological data are shown at the bottom of figure 3.6. The methods used to generate these tuning curves from

⁵Training such networks on natural images has often been shown to result in V1-like tuning (Olshausen and Field, 1996).

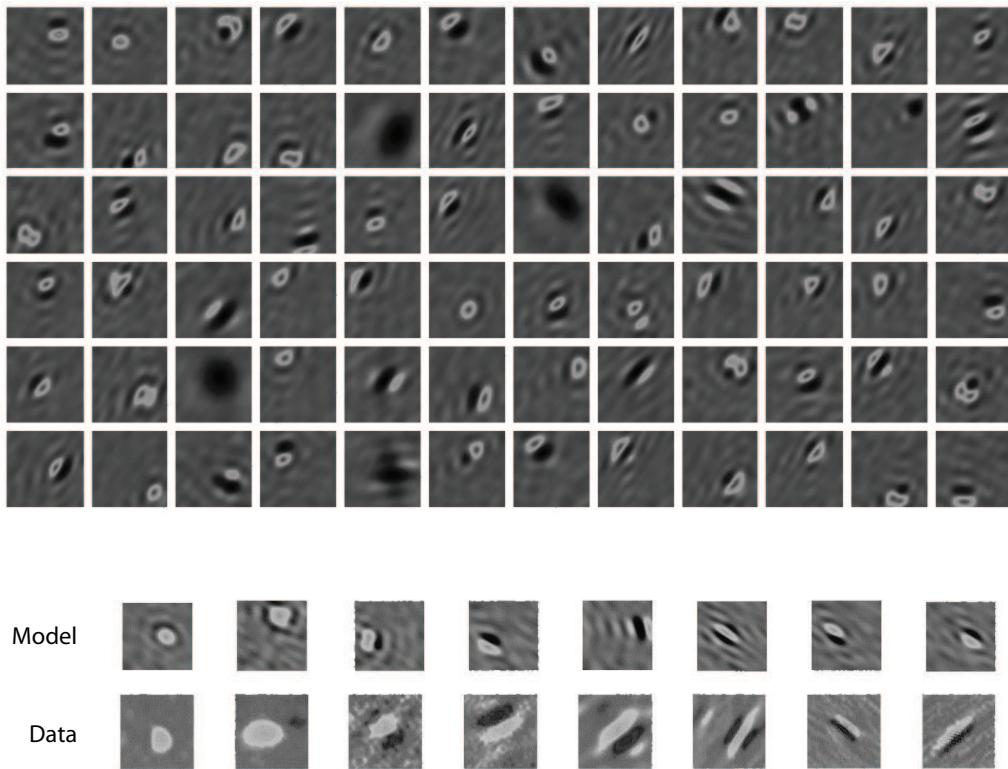


Figure 3.6: A sample of tuning curves of neurons in the model. These learned representations have a variety of orientations, spatial frequencies, and positions, like those found in V1. A comparison of data and model tuning curves from V1. Data adapted from Ringach (2002) with permission.

spiking single cells are identical to those used to generate the biological curves (Ringach, 2002).

These tuning curves are used to represent the input digits, and then the remaining layers of the network are trained to capture the statistics of that input. By the time we get to the highest level of the hierarchy, we have a much smaller (i.e., compressed) representation summarizing what has been presented to the retina. This compressed representation is a semantic pointer.

Notably, the highest layer of this network has 50 nodes, which means that the state space is 50-dimensional. It is this 50D space that contains the semantic pointers whose contents tell us about the presented digits. Clearly, this representation does not contain all of the information available in early visual areas. Instead,

it is a summary that usefully supports this object recognition task. This network, like most in machine vision, can score less than 2% error on the 10,000 test digits which have not been used to train the network (i.e., they classify about 200 wrong). In fact, these models outperform humans on this and other similar tasks (Chaaban and Scheessele, 2007).

So, these compressed representations (i.e., semantic pointers), like the lexical ones discussed previously, can capture important information that can be used to classify the input. In both cases, it is shallow comparisons, that is, those directly between the semantic pointers, that result in the classification. So, in both cases, the semantic pointers themselves carry shallow semantic information. However, there are two important differences between these semantic pointers and the lexical ones. First, these were generated based on raw images. This, of course, is a more biologically relevant input stream than text: we do not have organs for directly detecting text. Consequently, the 50D pointers are grounded in a natural visual input. If we have a way of treating these pointers like symbols, then we have found a natural solution to the symbol grounding problem.

Second, and more importantly, these 50D pointers can be used to drive deep semantics. That is, we can, in a sense, run the model “backwards” to decode, or dereference, the 50D representation. In other words, we can clamp the semantic pointer representation at the top level of the network and then generate an input image at the lowest level.⁶ Several examples of this process are shown in figure 3.7.

This figure demonstrates that a lot of the detail of an input is in fact captured by the semantic pointer representation. Subtleties of the way particular letters are drawn, such as whether an “8” is slanted or not, can be reconstructed from the pointer that such a figure generates. It is these subtleties that capture the deep semantics of this representation. However, it is obviously not always the case that we have a precisely drawn “8” in mind when we talk about the number “8”. That is, we might want to have access to the deep visual semantics of a pointer, when it is generated by an auditory input. In such a case, we can still use a visual semantic pointer to represent a generic “8”. A natural choice is the mean of the pointers associated with a category (see figure 3.7), which can be thought of as a prototype

⁶This does not suggest that the brain somehow recreates retinal images (there are no neurons that project from cortex to retina). Instead, figure 3.7 shows the retinal images that are consistent with the unpacked cortical representations. The deep semantic information at these non-retinal levels is accessible to the rest of cortex. In the brain, the unpacking would stop sooner, but could still be carried out to as low a level as necessary for the task at hand. This is one reason why seeing an image is not the same as imagining one, no matter how detailed the imagining.

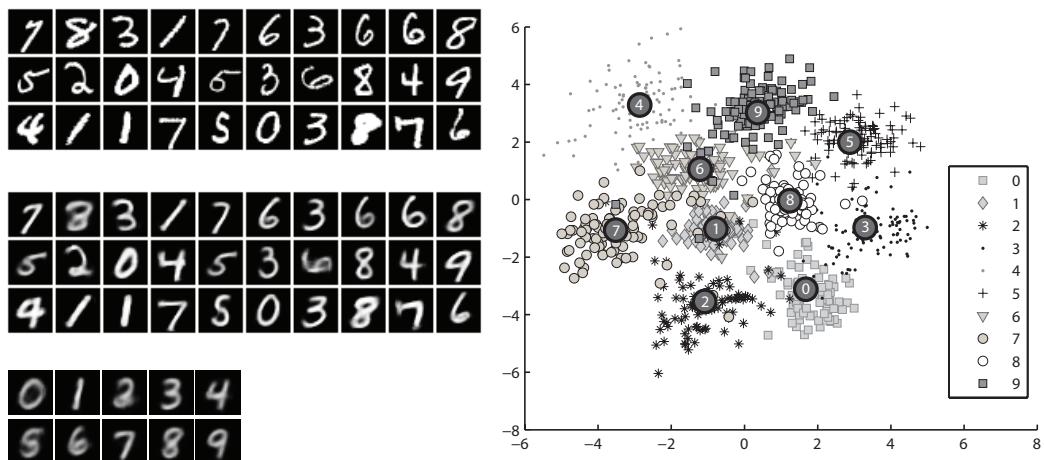


Figure 3.7: Dereferencing semantic pointers. (Top left) The original input of several randomly chosen images. These are compressed into a 50D representation at the highest level of the model. (Middle left) The resulting images from dereferencing the 50D semantic pointers. The 50D representations generated by the network based on the input are clamped at the highest-level, and the rest of the network is used to generate a guess as to what input would produce that semantic pointer. (Right) The compressed 50D representation is plotted in two dimensions using dimensionality reduction techniques. Large dark circles indicate the mean value of the semantic pointers for each category. These can be thought of as the middle of the dimples on the conceptual golf ball (see figure 3.1). (Bottom left) The unpacked mean semantic pointer for each category. These can be thought of as a prototype of the category.

of the category.⁷ If other instances of the category need to be generated, small random movements around this prototype will result in a variety of examples.

Because semantic pointers can be dereferenced to provide detailed perceptual information, it is natural to think of them as “pointers”. However, the dereferencing procedure depends on having the full perceptual hierarchy available. This maps well onto the fact that during deep semantic processing, perceptual areas are active in human subjects, as discussed earlier. It is, of course, a top-down procedure that allows these deep, grounded semantics to be regenerated.

The existence of a dereferencing procedure also serves to distinguish this method for generating semantic pointers from the purely shallow method described in the previous section, in the context of language processing. This is a good example of a difference in degree becoming a difference in kind. In both cases, a statistical model of a domain is generated based on a large set of training examples. But only in the perceptual case is the detailed connection of the semantic pointer to those examples at all preserved. The representation of words in the linguistic context is simply a co-occurrence structure (which captures very little about syntactic structure or meaning). The representation of digits in the perceptual context captures the relationships between a wide variety of deformations of the input along many degrees of freedom relevant for constructing digits. Indeed, the structure of this space seems to be nearly as good as that employed by people, given similar performance in classification tasks (Chaabani and Scheessele, 2007).⁸ Consequently, only in the second case would we want to claim that we have deep semantics for the domain of interest. That is, only in the case where we can define a dereferencing procedure that accounts for most of the domain’s structure, would we claim to capture deep semantics.

So far, I have not said how the pointers themselves can be used in symbol-like structured representations (that is the purpose of the next chapter). But, if that story is plausible, then the overall shape of the SPA should be clear. Semantic pointers, generated by grounded perceptual processing, can be “stripped-off” of that processing and used to both carry shallow semantics and be treated like a symbol. If deep semantics are needed, the semantic pointer can be used to clamp the top layer of the perceptual network that gave rise to it, and the network can

⁷Note that the “2” prototype is a combination of twos with straight bottoms and twos with loopy bottoms. This may suggest simply taking the mean is a mistake, and that there are two subcategories here. Dealing with these interesting, but additional complexities is beyond the scope of the present discussion.

⁸I suspect the human specification is much more robust to a wider variety of noise and variability in viewing conditions (Tang and Eliasmith, 2010).

re-generate the deep semantics.

In the terminology of the NEF, this story is one at the level of the state space. That is, these pointers and processing are defined in vector spaces that are represented by neurons. However, we have also applied the NEF principles to construct such a model in spiking neurons, to ensure that it will function on a biological substrate, and to allow more direct comparison of our model with neuroscientific data, as we have done in figure 3.6. There, we considered similarities in tuning curves between the model and neurons recorded in V1.

We can also examine the kinds of tuning curves evident in the highest level of our model, associated with processing in inferotemporal cortex (IT). As shown in figure 3.8, we get a wide variety of tuning curves in these cells. Some cells, such as that in figure 3.8a, are reminiscent of the well-known Jennifer Aniston or Halle Berry neurons reported by (Quiroga et al., 2005), because they are highly selective to instances of a single category. Other cells are more broadly selective for large parts of one category as in figure 3.8b. Other cells do not seem especially selective for any category, as in figure 3.8c. Finally, some cells are simply difficult for us to classify in any systematic way, such as that shown in 3.8d, where the cell seems tuned to arbitrary, specific cases from a variety of categories.

This variety of tuning reflects that observed in single cell responses in visual cortex, but a detailed comparison of these responses is outside of my present scope. Perhaps the most important consequence of these observations is that attempts to argue for particular kinds of ‘localist’ representation (Bowers, 2009) which do not take into account this variety – focusing instead on specific, hand-selected neurons – are doomed to a degree of arbitrariness. Understanding how *populations* of neurons represent an entire space of perceptual input is much more relevant for constructing grounded neural representations which support cognitive function (Stewart et al., 2011). This, of course, is the purpose of introducing the notion of a semantic pointer as a functionally relevant unit of representation. Semantic pointers, after all, are represented by populations of cells and identified with respect to a semantic space.

While models like the one presented here provide specific examples of how semantic pointers may be generated, it is important to keep in mind the limitations of such models. For instance, there is no doubt much more of relevance regarding biological structure that should inform how we construct our statistical models that has not yet been taken into account. While these models mimic the basic hierarchical structure of perceptual systems, and can approximately be mapped onto observed anatomical connections, most such models do not have connections that skip layers of the hierarchy, as observed in cortex. In addition, many other

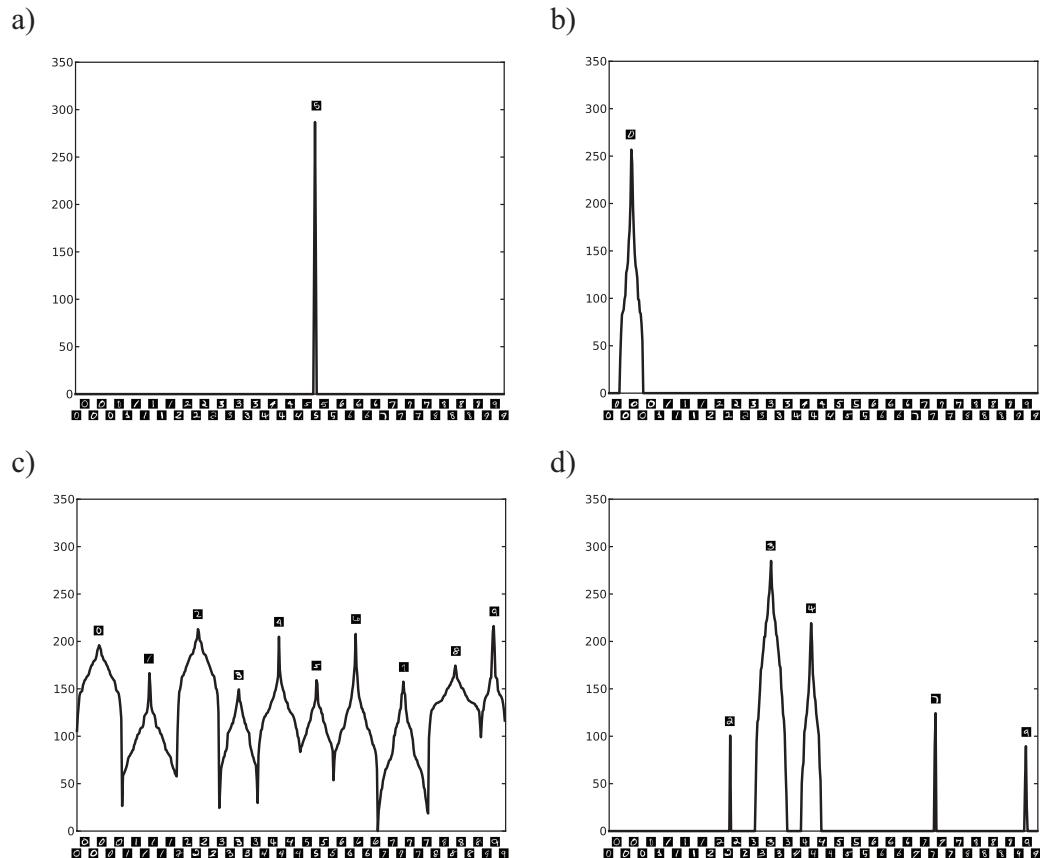


Figure 3.8: Tuning curves of neurons in IT in the model. a) A highly specific neuron, that is highly selective of an instance of a single category. b) A more categorical neuron that is responsive largely to a single category. c) A very broadly tuned neuron, that does not obviously distinguish categories. d) A complex neuron whose response specificity is difficult to intuitively categorize. All tuning curves were generated by placing the highest firing rate item in the middle and working progressively outwards, by category.

processes important for recognition, such as attention, tend to be excluded from current state-of-the-art models (c.f., Tang and Eliasmith, 2010).

Finally, there remain many questions regarding how all aspects of such models could be implemented by biological networks: Can the learning of these models be done by biologically plausible learning rules? How many dimensions should each layer of the network be taken to represent? How does the brain access information at any desired level of the perceptual network in a flexible manner? The model presented above does not address these issues in detail (although see 6.4 and 5.5 for related considerations). Nevertheless, for current purposes, there are three crucial features of this model that are relevant for understanding the SPA characterization of semantics. Specifically, these are: 1) the model constructs high-level representations of the perceptual input that capture statistical relationships that describe the domain’s structure; 2) that such representations are “compressed” (i.e., lower dimensional) representations of the input; and 3) such representations can be “dereferenced” to provide detailed perceptual information.

3.6 Deep semantics for action

Interestingly, we can find these same features in biological representations used to drive motor states. Of course, the task for motor systems seems much different than that for perceptual systems. The motor system does not need to classify presented stimuli at all, rather it needs to direct a nonlinear, high-dimensional system towards a desired state. So, in general we might notice that perceptual systems often need to map from a high-dimensional, ambiguous state (e.g., images generated by highly nonlinear environmental processes) to a much smaller set of states (e.g., object categories), whereas motor systems often need to map from a small set of states (e.g., desired pointing targets) to a high-dimensional, ambiguous state (e.g., any one of the possible configurations of muscle tensions over the body that results in pointing to a target). These tasks seem to be almost exact *opposites*.

Sometimes, opposition is a highly informative relationship. For instance, in this case there is a lot of shared by perception and motor control: both need to map low- to high-dimensional states; both need to deal with complexity and non-linearity in doing so; both need to deal with uncertainty and ambiguity in doing so; and both need to share information between the past and future in doing so. In mathematical terms, problems which share their structure in this kind of way are called “dual problems”. It is useful to identify duality between problems because if one kind of problem can be solved, then so can the other.

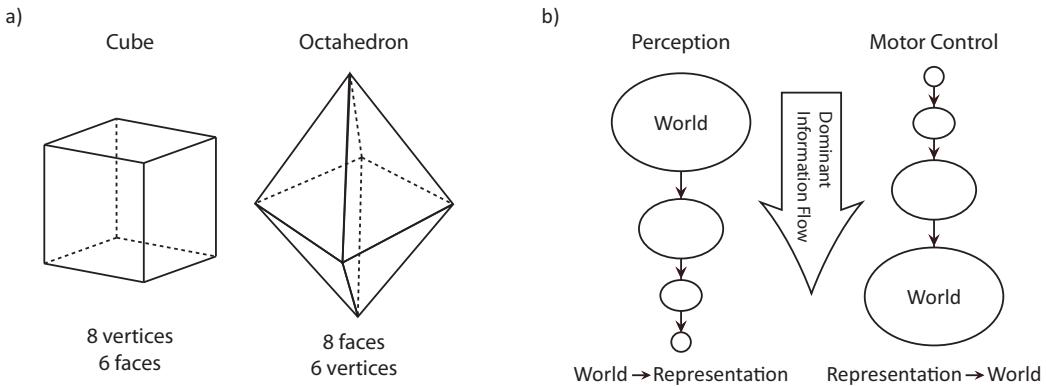


Figure 3.9: Dual problems. a) The cube and the octahedron provide a simple example. Both figures share a similar geometric description, but the number of vertices and faces of the two shapes are reversed. b) The perceptual and motor systems are much more complicated, but can also be treated as duals. The perceptual system encodes high-dimensional, non-linear relations using hierarchically compressed representations. The motor control system reverses this process (though with similar structure and representations) to determine high-dimensional nonlinear control signals from a low-dimensional signal.

As a simple example, consider the relationship that exists between a cube and an octahedron (see figure 3.9a). Notice that there are the same number of sides in a cube as vertices in an octahedron and vice versa. As well, both have twelve edges, connecting the relevant vertices/faces in a structurally analogous manner. These two solids are thus duals. If we pose a problem for one of these solids – e.g., What is the volume of the smallest sphere that intersects all vertices (faces)? – then a solution for one of the solids provides a solution for the other. This is true as long as we swap the relevant structural elements (i.e., faces and vertices) appropriately.

Why does this matter for understanding perception and motor control? Because this dual relationship has been suggested to exist between statistical models of perceptual processes and optimal control models of motor processes (Todorov, 2007, 2009). Figure 3.9b suggests a mapping between perceptual and motor systems that takes advantage of this duality. From an architectural point of view, this duality is very useful because it means that there is nothing different *in kind* about perceptual and motor systems. From the perspective the SPA in particular, this means that semantic pointers can play the same role in both perception and

action. Much of the remaining discussion in this section describes this role in the context of motor control.

To begin, like the perceptual system, the motor system is commonly taken to be hierarchical (see figure 3.10). Typically, we think of information as flowing down rather than up the motor hierarchy. For instance, suppose you would like to move your hand towards a given target object. Once the desired goal state has been generated, it is provided to the cortical motor system. The first level of this system may then determine which direction the arm must move in order to reach that goal state.⁹ Once the direction of movement has been determined, it can be used as a control signal for a controller that is lower in the hierarchy. Of course, specification of the direction of movement does not determine how torques need to be applied at the joints of the arm in order to realize that movement. This, then, would be the function of the next lower level in the control hierarchy. Once the specific forces needed to move the arm are determined, the specific tensions that need to be produced in the muscles in order to realize those forces must be generated by the activity of motor neurons. In short, as the motor command becomes more and more specific to the particular circumstance (e.g., including the particular part of the body that is being moved, the current orientation of the body, the medium through which the body is moving, etc.), lower level controllers are recruited to determine the appropriate control signal for the next lower level. Ultimately, this results in activity in motor neurons that cause muscles to contract and our bodies move.

Notably, just as it is inaccurate to think of information as flowing only “up” the perceptual hierarchy, so is it a mistake to think of this picture of motor control as being in one direction. As well, in both systems, there are many connections that skip hierarchical levels, so the flow of information is extremely complex. Nevertheless, the main problems to be tackled are very similar. So, just as we can begin our characterization of perception as thinking of higher levels in the perceptual hierarchy as constructing models of the levels below them, so we can think of higher levels in the motor hierarchy as having models of the lower levels. Higher levels can then use such models to determine an appropriate control signal to affect the behavior of that lower level.

There is good evidence for this kind of control structure in the brain (Wolpert and Kawato, 1998; Kawato, 1995; Oztop et al., 2006). That is, a control structure

⁹This description is highly simplified, and employs representations and movement decompositions that may not occur in the motor system. Identifying more realistic representations would require a much lengthier discussion, but would not add to the main point of this section. For more detailed descriptions of such representations, see (Dewolf, 2010), and Dewolf neur eng. paper???

in which higher levels of the hierarchy have explicit models of the behavior of lower levels. When we attempt to specify these models, a simplification in our characterization of perception becomes problematic: we assumed that the models were static. In motor control, there is no such thing as non-temporal processing. Time is unavoidable. Embracing this observation actually renders the connection between perceptual and motor hierarchies even deeper: higher levels in both hierarchies must model the statistics *and* the dynamics of the levels below them.

In the previous perceptual model, the statistics at higher-levels capture the regularities at lower levels. In the neural model, these were evident as neural tuning curves that were used to represent the perceptual space. In motor control, these same kinds of regularities are often called “synergies” (Bernstein, 1967; Lee, 1984). Synergies are useful organizations of sets of movements that are often elicited together. As before, we would expect the tuning curves of neural models to reflect these synergies for the representation of the space of motor actions. These representations need to be learned based on the statistics of the space they are representing, as in the perceptual model above.

What is not addressed by that simple perceptual model is dynamics.¹⁰ This simplification is not appropriate when considering motor control. After all, the particular dynamics of the system are likely to affect which synergies are most useful for controlling action. So, the representational and dynamical aspects of the system are tightly coupled. In fact, it is natural to describe this connection in a way captured by the third principle of the NEF: the dynamical models are defined over the representational state space. That is, the dynamical models of a given level of the hierarchy are defined using the synergies of that level.¹¹ The resulting dynamics then drive lower levels, and feedback from lower levels can inform these higher-level models. So, despite the added complexity of dynamics, the general structure of the motor system is hierarchical, just like the perceptual system.

Consequently, like the simpler perceptual model, there are two main features

¹⁰For effective control, dynamical models are typically broken into two components: a forward model and an inverse model. A forward model is one which predicts the next state of the system given the current state and the control signal. An inverse model performs the opposite task, providing a control signal that can move between two given states of the system. For simplicity, I discuss both using the term “dynamical model”, as this level of detail is beyond the current scope.

¹¹There is some issue here about drawing boundaries to determine the input and output of these models. Essentially, a control (error) signal can be thought of as generated at a given hierarchical level and then mapped to a lower (higher) level, or the mapping and generation may be considered concurrent, in which case the dynamical model maps synergies at different levels. It is possible that both happen in the nervous system, but I believe the former is easier to conceptualize.

of the motor system: 1) the (dynamical) model constructs representations of the motor output that capture the statistical relationships that sufficiently describe the domain’s structure; and 2) that such representations are “compressed” (i.e., lower dimensional) representations of the output. I suspect, but will not consider the details here, that the dynamical motor hierarchy is a better model for perceptual processing (including object recognition) than the kinds of static models considered earlier. Nevertheless, the main conceptual points relevant to the SPA remain: perception and action have dual descriptions in which semantic pointers can play the same role. Let us now consider, in more detail, how the SPA relates to motor control.

A graduate student in my lab, Travis Dewolf (2010), recently proposed a framework that integrates known neuroanatomy with a hierarchical control characterization of the function of the motor system. This framework is called the Neural Optimal Control Hierarchy (NOCH), and a simplified version is shown in figure 3.10. This particular description is tailored to arm control to simplify this discussion. Models based on NOCH are able to explain the effects of various motor system perturbations including Huntington’s disease, Parkinson’s disease, and cerebellar damage (DeWolf and Eliasmith, 2010). The task I consider here is reaching in a plane towards a target (see figure 3.11).

As shown in figure 3.11, a simple NOCH model is able to move from a low-dimensional target specification, to a higher-dimensional control signal appropriate for moving the arm, to an even higher-dimensional control signal needed to contract muscles appropriately to cause that movement. In this case, the synergies are not learned, but rather chosen to be appropriate for arm movements. Nevertheless, the dual relation to the perceptual hierarchy is clear: low-dimensional, high-level representations in premotor cortex (PM) are used to drive the high-dimensional, low-level spinal cord to affect movement.

While simple, this motor control example can be used to give a sense of how semantic pointers capture the distinction between deep and shallow semantic processing in the motor system. As in the perceptual case, a low-dimensional semantic pointer can be “dereferenced” by the remainder of the system into a specific, detailed movement. This is demonstrated in figure 3.12 where similar semantic pointer representations are shown to result in similar high-dimensional control signals and similar movements.

This is consistent with a contemporary understanding of how motor cortex functions. Indeed, the work of Georgopoulos might be seen as an attempt to map some levels of this hierarchy (see section 2.5, especially figure 2.15), and hence it has already been shown that aspects of movement structure are reflected in mo-

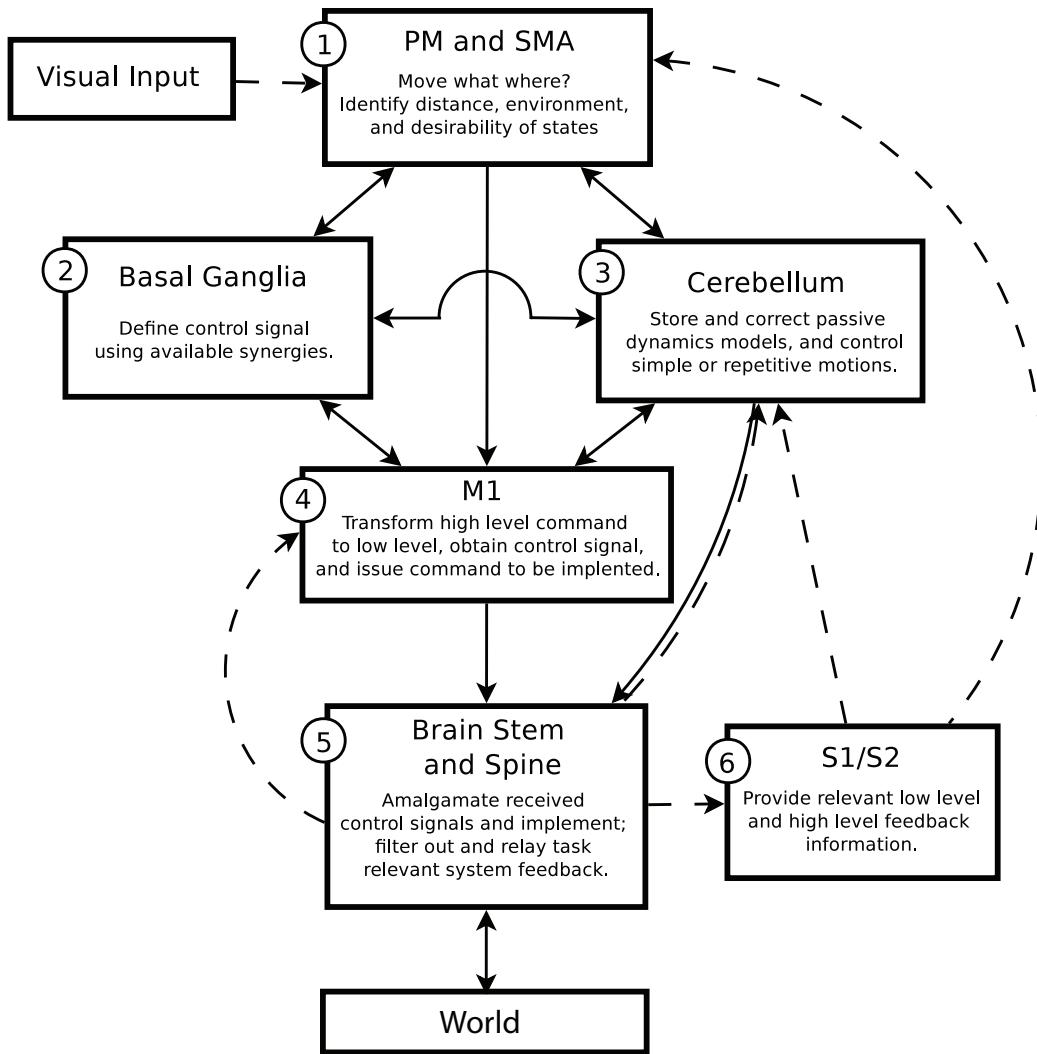


Figure 3.10: Neural optimal control hierarchy (NOCH). This framework maps neuroanatomy onto optimal control. The numbers indicate the approximate order of processing for a movement. Each box contains a brief description of the function of that area. Semantic pointers can be thought of as driving the top of this hierarchy, as in the perceptual system. Abbreviations: PM - premotor area; SMA - supplementary motor area; M1 - primary motor area; S1 - primary somatosensory area; S2 - secondary somatosensory area. This diagram is a simplified version of NOCH presented in Dewolf (2010).

Figure 3.11: Example reaches of an arm driven by a hierarchical controller. a) Eight example reaches in a 2D space of a nonlinear 3 degree-of-freedom arm. b) The high-level (2D), mid-level (3D) and low-level (6D) control signals generated by the control hierarchy for the first movement (shown in bold in a). ???Travis

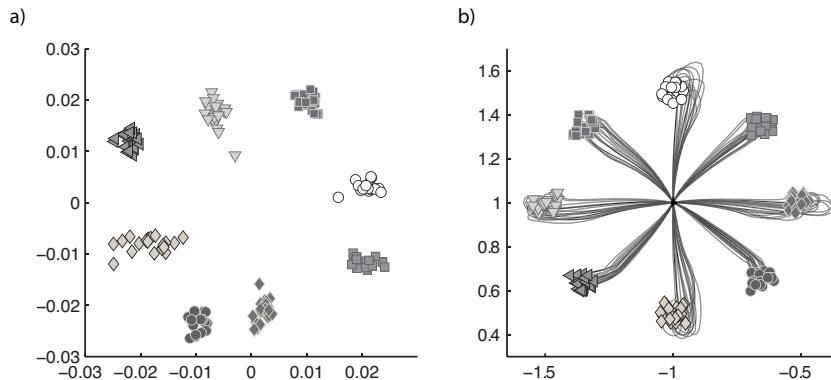


Figure 3.12: A semantic motor space. This figure shows how similar high-level semantic pointers can generate similar low-level movements. a) The clustering of 6D control signals used to drive the movements in b). These are arbitrarily rotated because of the dimensionality reduction performed to plot them in a 2D space. b) The movements of the arm driven by one of eight semantic pointers (corresponding to different marker shapes). The “dereferencing” of the pointers drive the high-dimensional system consistently in external space.

tor areas. More recent work in motor control has been suggesting that higher-dimensional representations are in fact the norm in motor cortex (Churchland et al., 2010). And, in our own recent work we have shown that a spiking neuron implementation of this kind of NOCH model effectively captures a wide variety of the nonlinear temporal dynamics observed in the responses of motor neurons (dewolf eliasmith neural eng paper???). As a result, the NOCH approach maps well to the biological basis of motor control.

Conceptually, the most important consequence of NOCH is captured by a comparison of figure 3.12 to the perceptual example shown in figure 3.7. This comparison highlights the many similarities of the characterizations of motor and perceptual semantics I have presented so far. In short, the high-dimensional control signals of the motor system are like the high-dimensional input images, and the low-dimensional semantic pointers for target positions are like the semantic

pointers for image category.

In cases that demand deep semantic processing in the motor system (e.g., that require estimating the precise configuration of joints), the high-level semantic pointers can be used to “run” the models in the motor system in order to internally generate more information about the nature of reaching movements at intermediate levels. Automatically inferring details about motor actions associated with some concepts during deep processing is a central aspect of conceptual semantics (Barsalou, 2009). In addition, there are well-known positive effects of “visualizing” motor performances before executing them (Landers and Feltz, 2007). Thus, as in the perceptual case, deep processing of motor semantics is supported by the same kind of dereferencing process of a high-level semantic pointer.

Recall that in the previous section, we identified two important features of semantic pointers: they capture higher-order relationships; and they are compressed representations. It should now be clear how these features are realized by the semantic pointers generated by the motor system. First, the semantic pointers capture higher-order relationships between states of the body because they can be “dereferenced” in order to coordinate those states for successful motor action. And second, the representations at the highest level of the motor hierarchy are lower dimensional, and in many ways less specific, than those at the lowest level. As a result, they are usefully considered as compressed representations.

This simple example suggests that semantic pointers can be found playing similar roles in both the perceptual and motor systems. There are, of course, important differences between the perceptual and motor models I have presented above. Most obviously, the perceptual model does not include dynamics, and the motor model does not learn the lower-level representations (i.e., synergies). There is ongoing research that attempts to address both of these challenges,¹² including work in my own lab. The central theoretical features of this account, however, should not change as these simplifications are rectified so long as the identified duality of the sensory and motor systems remain.

¹²For instance, Rod Grupen, Rolf Pfeifer, Dana Kulic, and other roboticists have been working on these issues from a motor control and reinforcement learning perspective. Geoff Hinton, Yann LeCun, Yoshua Bengio and others have been adopting a more statistical modelling, and perceptually oriented approach.

3.7 The semantics of perception and action

My discussion of the SPA in the context of perception and action has adopted the usual approach of considering these aspects of behaviour somewhat independently. However, it is hopefully clear from the above discussion that action and perception are not only tightly conceptually linked, but also mutually serve to capture deep semantics in the SPA. The parallel stories provided above for action and perception are intended to lay the groundwork for more cognitive consideration of semantics. I leave that task until the end of the next chapter (section 4.7), after considering how semantic pointers can be used in a symbol-like manner. For the moment, I would like to further consider the relationship between action and perception in the SPA.

Notice that both characterizations depend on the hierarchy being “run” in both directions. In the forward direction (up the hierarchy), the perceptual hierarchy allows classification of visual stimuli, in the reverse direction it allows the generation of deep semantics. Both directions help the system learn appropriate representations. For the motor hierarchy, the forward direction (down the hierarchy) allows for the control of a sophisticated body using simple commands and the generation of deep semantics. The reverse direction allows for online fine tuning that allow the commands to be effective. Again, both directions allow the system to learn appropriate representations (i.e., synergies). So it is clear that in both cases, the semantic pointers operating at the top of the hierarchy are always “pointing to a memory”, and can be used in the hierarchy to elicit details of that memory. Of course, this use of the term “memory” is, unlike in a computer, necessarily constructive. We can make movements we have never made before, and we can imagine handwriting we have never seen before. This observation supports the notion that perceptual and motor systems are well-characterized as constructing statistical models of their past.

In the above examples, I have discussed a number of different ways of constructing these models – of producing semantic pointers. It is worth reiterating that these suggestions are not an essential part of the SPA itself. It is also worth noting that I have been speaking of semantic pointers as if they only reside at the “top” of these hierarchies. I would like to be clear that this is a simplification to help introduce the notion. Clearly, some representations at other levels of the hierarchy are also compressed, lower-dimensional (than earlier in the hierarchy) representations. They too may be used by other parts of the system as pointers to more detailed semantic information. For explanatory purposes, and perhaps for reasons of efficiency, the most compressed representations are paradigmatic

semantic pointers, but they need not be the only kind. So, the means of defining the semantics of the pointers, and their precise location in the hierarchy are not essential to the SPA. What is central to the architecture is that the representations generated by these models act as low-dimensional summaries of high-dimensional systems engaged by the brain. This central feature highlights another defeasible assumption of the above examples: that the perceptual and motor models are generated independently.

As has been argued forcefully in a variety of disciplines,¹³ it is a mistake to think of biological systems as processing perceptual information and then processing motor information. Rather, both are processed concurrently, and inform one another “all the way up” the hierarchies. Consequently, it is more appropriate to think of the semantics of items at the top of both hierarchies as having concurrent perceptual and motor semantics. The integrative model in chapter 7 provides an example of this. In that case, the semantic pointer generated by the perceptual system is “dereferenced” by the motor system. Thus, the deep semantics of the pointer depends on both of the models that can be used to dereference it.

The SPA thus provides a natural, but precise and quantitative, means of specifying the kinds of interaction between perception and action that have seemed central to cognitive behavior. Such a characterization may remind some of the notion of an “affordance”, introduced by psychologist James Gibson (1977). Affordances are “action possibilities” that are determined by the relationships between an organism and its environment. Gibson suggested that affordances are automatically picked up by animals in their natural environments, and provide a better characterization of perception (as linked to action), than traditional views.

Several robotics researchers have embraced these ideas, and found them useful starting points for building interactive perception/action systems (Scheier and Pfeifer, 1995; Ballard, 1991). Similarly, the notion of an “affordance”, while somewhat vague, seems to relate to the information captured by a semantic pointer when embedded in the SPA. As just mentioned, these pointers can be used as direct links between action and perception that depend on the motor and perceptual experiences of the organism. Consequently, their semantics are directly tied to both the environment and body in which they are generated. There are, undoubtedly, many differences between semantic pointers and affordances as well (e.g., semantic pointers do not seem to be “directly perceived” in the sense champi-

¹³I can by no means give comprehensive coverage of the many researchers who have made these sorts of arguments, which can be found in robotics, philosophy, neuroscience, and psychology. Some useful starting points include Brooks (1991); Churchland et al. (1994); Port and van Gelder (1995); Regan and Noë (2001).

oned by Gibson, for example). Nevertheless, the similarities help relate semantic pointers to concepts already familiar in psychology.

As well, affordances highlight the important interplay between perception and action. Motor control, after all, is only as good as the information it gathers from the environment. As a result, it is somewhat awkward to attempt to describe a motor controller without discussing perceptual input (notice that NOCH, in figure 3.10, includes perceptual feedback). It is much more appropriate to conceive of the entire perception/action system as being a series of nested controllers, rather than a feed-in and feed-out hierarchy. As depicted in figure 3.13, nested controllers become evident by inter-leaving the hierarchies of the kind described above.¹⁴ What should be immediately evident from this structure, is that the process of perceiving and controlling becomes much more dynamic, interacting at many levels of what were previously conceived of as separate hierarchies. This, of course, makes the system much more complicated, which is why it was more convenient to describe it as two hierarchies. And, indeed, the real system is more complicated still, with connections that skip levels of the hierarchy, and multiple perceptual paths interacting with a single level of the controller hierarchy. The question of relevance is: does identifying this more sophisticated structure change our story about semantic pointers?

Given the suggested genesis and use of semantic pointers, I believe the answer is no. Recall that the way the highest level representations in the motor and perceptual hierarchies were generated, was by characterizing statistical models that identify the relationships within the domain of interest. Whether these models are influenced solely by perceptual or motor processes, or whether they are influenced by perceptual and motor processes, may change the nature of those relationships, but will not change the effective methods for characterizing those relationships. It will, then, still be the case that dereferencing a perceptual representation for deep semantic processing results in identifying finer perceptual details not available in the higher-level (semantic pointer) representation. As suggested earlier, these same pointers may include information about relevant motor activities. So, semantic pointers will still be compressed and capture higher-order relationships, it just may be that their contents are neither strictly perceptual nor strictly motor. So perhaps the perceptual/motor divide is one that is occasionally convenient for theorizing, even though it is not built into the semantics of our conceptual system.

¹⁴The observation that perceptual and motor cortex are both hierarchically structured and mutually interacting is hardly a new new one (Fuster, 2000), what is new, I believe, is the computational specification, the biological implementation of the computations, and the integration into a cognitive hierarchy.

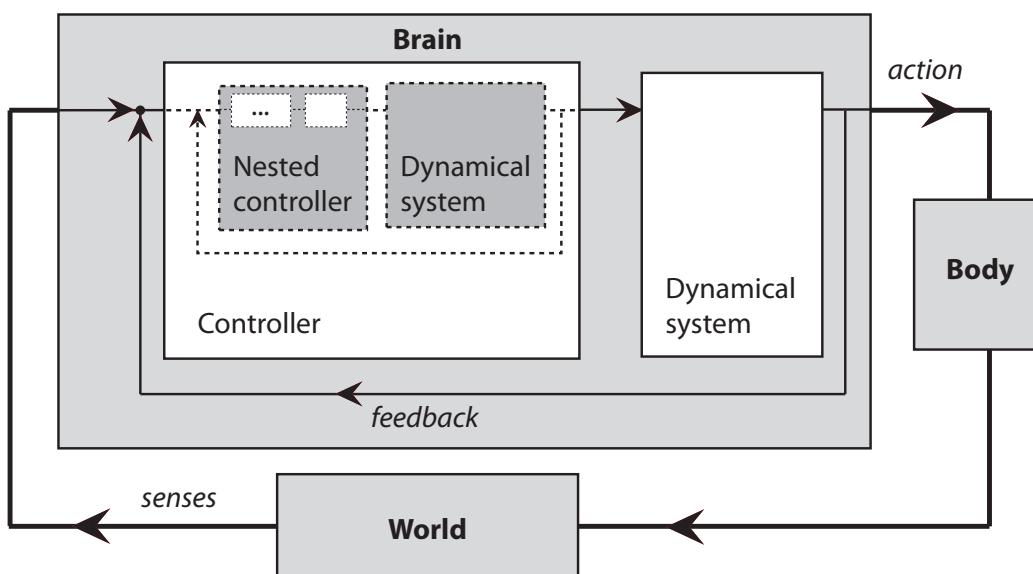


Figure 3.13: SPA as nested controllers. A controller is a component of a dynamic system that reacts to an input to produce a certain output which will influence other parts of the system. Feedback from the resulting change is then added to the input to the controller, allowing it to dynamically adjust its output to achieve a particular state. In this diagram, the brain acts as a controller for the body, triggering an action that has an impact on the input the brain receives through the senses, but the brain itself contains other dynamical systems and controllers which may be composed of further controllers and subsystems.

We should also be careful, however, not to overstate the closeness of the relationship between perception and action. Dissociation between impairment to visually guided motor control and object identification has been well-established (Goodale and Milner, 1992). Some patients, with damage to the dorsal visual pathways, can identify objects but not reach appropriately for them. Others, with damage to the ventral visual pathways, can reach appropriately, but not classify objects. This suggests two things: 1) that motor action and visual perception can come apart; and 2) that the object classification model presented earlier only models part of the visual system (the ventral stream), at best. Again, the relevant point here is not about the completeness of the models, but that the subtle, complex character of action and perception in biological systems is consistent with the assumptions of the SPA. Capturing the appropriate degree of integration and distinctness of action and perception is a task independent of generating semantic pointers, so long as there are high-level representations with the assumed properties of semantic pointers.

So, semantic pointers are compressed representations of motor/perceptual information found within a nested control structure. The story I have told so far is intended to characterize semantics, both deep and shallow, as related to semantic pointers. Consequently, I would argue that this account is theoretically appealing because it unifies deep and shallow semantics along with sensory and motor semantics in a manner that can be realized in a biological cognitive system. What remains to be described in order to make semantic pointers cognitively relevant, is how they can actually be used to encode complex syntactic structures. That is, how are these perceptual/motor vectors used to generate the kinds of language-like representations that underlie high-level cognition? Answering this question is the purpose of the next chapter.

3.8 Nengo: Neural computations

Characterizing neural representation is a useful first step to understanding biological cognition. However, in order to generate interesting behavior, neural representations must be transformed in various ways – i.e., they must be used in neural computations. In many cases such transformations, especially when they are non-linear, result in novel neural representations.

In this tutorial, we examine how the second NEF principle (see section 2.3.2) can be exploited in Nengo to compute transformations. This discussion builds on the previous tutorial on neural representation (section 2.5), so it may be helpful

to review that material. The following discussion is broken into two sections, the first on linear transformations and the second on nonlinear transformations. The tutorial focuses on scalar transformations, but the methods generalize readily to any level of the representational hierarchy, as I briefly discuss.

Linear transformations

As formulated, principle 2 makes it clear that transformation is an extension of representation. In general, transformations rely on the same encoding, but exploit a different decoding than is used when representing a variable. I will introduce such transformations shortly. First, however, we can consider a transformation that is a basic property of single neurons – addition. Addition transforms two inputs into a single output which is their sum.

- In Nengo, create a new network. Set the *Name* of the network to ‘Scalar Addition’.
- Create a new ensemble within the ‘Scalar Addition’ network. Name the new ensemble ‘A’ and set *Number of Nodes* to 100, otherwise use the default settings.

To ensure the ensemble uses the default settings, choose ‘default’ from the drop-down menu at the top and then change any settings as appropriate. In general, Nengo reuses the last set of settings used to generate an ensemble.

- Create a second ensemble within the ‘Scalar Addition’ network. Name this ensemble ‘B’ and use the same parameters as you did for ensemble A.

These two neural ensembles will represent the two variables being summed. As with the tutorial on neural representation, we now need to connect external inputs to these ensembles.

- Create a new function input, name it ‘input A’, set *Output Dimensions* to 1, and click *Set Functions* to ensure that the function will be a *Constant Function*.
- Click *OK* until all dialog windows are closed.
- Add a termination to ensemble A.

- Name the termination ‘input’ and ensure that number of input dimensions (*Input Dim*) is set to 1. The default value of *tauPSC* (0.02) is acceptable.
- Click *Set Weights* and confirm that the value in the 1 to 1 coupling matrix that appears is 1.0.
- Click *OK* to confirm the options set in each window until the new termination is created.
- Make a connection between the origin of ‘input A’ to the new termination on ensemble A.
- Follow the same procedure to make a second constant function input and project it to ensemble B.

This process should be familiar from the previous tutorial. We have represented two scalar values in neural ensembles and we could view the result in the *Interactive Plots* window as before. However, we wish to add the values in the two ensembles and to do this we will need a third ensemble to represent the sum.

- Create another ensemble. Call it ‘Sum’ and give it 100 nodes and 1 dimension. Set *Radius* to 2.0. Click *OK*.

Note that we have given the new ensemble a radius of 2.0. This means that it will be able to accurately represent values from -2.0 to 2.0 which is the maximum range of summing the input values. The ‘Sum’ ensemble now needs to receive projections from the two input ensembles so it can perform the actual addition.

- Create a new termination on the ‘Sum’ ensemble. Name it ‘input 1’, set *Input Dim* to 1.
- Open the coupling matrix by clicking *Set Weights* and set the value of the connection to 1.0.
- Finish creating the termination by clicking *OK* as needed.
- Create a second termination on the ‘Sum’ ensemble following the procedure outlined above, but name it ‘input 2’.
- Form a projection between ensemble ‘A’ and the ‘Sum’ ensemble by connecting the origin of ‘A’ to one of the terminations on ‘Sum’.

- Likewise form a projection between ensemble ‘B’ and the ‘Sum’ ensemble by connecting its origin to the remaining termination.

You have just completed the addition network; the addition is performed simply by having two terminations for two different inputs. Recall from the discussion in section 2.1 that when a neuron receives a signal from another neuron, postsynaptic currents are triggered in the receiving neuron and these currents are summed in the soma. When signals are received from multiple sources they are added in exactly the same fashion. This makes addition the default transformation of neuron inputs.

- To test the addition network, open the *Interactive Plots* viewer, display the control sliders for the inputs and the value of the ‘Sum’ ensemble, and run the simulation.
- Play with the values of the input variables until you are satisfied that the ‘Sum’ ensemble is representing the sum correctly. If the value graphs are difficult to see, right-click on the graph and select ‘auto-zoom’.

Another exploration to perform with this network is to set one of the inputs to zero, and change the value of the other input. You will see that the ‘Sum’ ensemble simply re-represents that non-zero input value. This is perhaps the simplest kind of transformation and is sometimes called a ‘communication channel’ because it simply communicates the value in the first ensemble to the second (i.e., it performs the identity transformation). It is interesting to note that the firing patterns can be very different in these two ensembles, even though they are representing the same value of their respective variables. This can be best seen by viewing either the spike rasters or the voltage grid from both ensembles simultaneously.

Now that you have a working addition network, it is simple to compute any linear function of scalars. Linear functions of scalars are of the form $z = c_1x_1 + c_2x_2 + \dots$ for any constants c_n and variables x_n . So, the next step towards implementing any linear function is to create a network that can multiply a represented variable by a constant value (the constant is often called a ‘gain’ or ‘scaling factor’). Scaling by a constant value should not be confused with multiplying together two variables, which is a non-linear operation that will be discussed below in section 3.8.

We will create this network by editing the addition network.

- Click the *Inspector* icon (the magnifying glass in the top right corner of the window).

- Click on the input termination on the ‘Sum’ population that is connected to ‘A’.
- In the inspector, click the gray triangle pointing to *A transform* (at the very bottom).

This now shows the coupling weight you set earlier when creating this termination. It should be equal to 1.0.

- Double-click the 1.0 value. In the window that appears, double-click the 1.0 value again. Change this value to 2.0.
- Click *Save Changes* on the open dialog box. You should see the updated value in the *Inspector*.

Setting the weight of a termination determines what constant multiple of the input value is added to the decoded value of an ensemble. The preceding steps have set up the weights such that the value of ‘A’ input is doubled.

- Test the network in the *Interactive Plots* window.

You should be able to confirm that the value of ‘Sum’ is twice the value of ‘A’, when you leave ‘B’ set to zero. However, if you leave ‘A’ at a high value and move ‘B’ you will begin to saturate the ‘Sum’ ensemble because its radius is only 2.0. The saturation (i.e. under-estimation) effect is minimal near the radius value (2.0), and becomes more evident the farther over the radius the ensemble is driven. To prevent saturation, change the radius of the ‘Sum’ population by changing the ‘radii’ property in the *Inspector* when the population is selected.

You can easily change the linear transformation between these ensembles by selecting any scalar value for the *A transform* of the two inputs to the ‘Sum’ population as described above. However, you must also set an appropriate radius for the ‘Sum’ ensemble to avoid saturation.

We can now generalize this network to any linear transformation, all of which can be written in the form $\mathbf{z} = \mathbf{C}_1\mathbf{x}_1 + \mathbf{C}_2\mathbf{x}_2 + \dots$, where \mathbf{C}_n are constant matrices and \mathbf{x}_n are vector variables. Computing linear transformations using vector representations instead of scalars does not introduce any new concepts, but the mapping between populations becomes slightly more complicated. We will consider only a single transformation in this network (i.e. $\mathbf{z} = \mathbf{Cx}$ where \mathbf{C} is the coupling matrix).

- In a blank Nengo workspace, create a network called ‘Arbitrary Linear Transformation’.

- Create a new ensemble named ‘x’ with the default template, and give it 200 nodes and 2 dimensions. This will be the input.
- Create another ensemble named ‘z’ with the default template, and give it 200 nodes and 3 dimensions. This will be the output.

Note that we have two vector ensembles of different dimensionality. To connect these together, we must specify the weights between each pair of variables in the two ensembles.

- Add a new decoded termination to ensemble ‘z’ called ‘input’.
- Set *Input Dim* to 2 then click *Set Weights*.

You should now see a ‘2 to 3 Coupling Matrix’ window. Each cell in the matrix defines a weight on a variable in ensemble ‘x’ and they combine to direct the result to the relevant variable in ensemble ‘z’. This works exactly like standard matrix multiplication. So, since there are 3 dimensions in the output, we have three rows. Each row weights the 2 input components and is then summed to give one value in the output vector.

- Enter the matrix

$$\begin{bmatrix} 0.0 & 1.0 \\ 1.0 & 0.0 \\ 0.5 & 0.5 \end{bmatrix}$$

into the coupling matrix and press *OK*.

This matrix swaps the first and second dimension of ‘x’ in the output ‘z’ and puts the average of those dimensions in the third dimension of ‘z’. So, for example, $\mathbf{x} = [1, 2]$ becomes $\mathbf{z} = [2, 1, 1.5]$.

- Click *OK* to create the decoded termination.
- Create a projection from the origin of ensemble ‘x’ to the termination of ensemble ‘z’.
- Create an ‘input’ termination on ensemble ‘x’. Set *Input Dim* to 2 and click *Set Weights* to set the coupling matrix to an identity matrix (1.0 along the diagonal).

Creating a coupling matrix with a diagonal matrix of ones in this manner causes the input dimensions to be mapped directly to corresponding output dimensions.

- Create a new function input. Give it 2 dimensions and set the functions for each of the 2 dimensions as constant functions.
- Project the new input to the termination of ensemble ‘x’.

You’ve now completed a network to implement arbitrary linear transformations from a 2-dimensional vector to a 3-dimensional vector, represented with neurons.

- Click the Interactive Plots icon, pull up input sliders, value graphs, and anything else of interest and test the simulation while it is running.

A simple way to check that it is performing the desired transformation is to move the first input slider to 0.5 and the second one to -0.5. The ‘z’ value should have the black and blue lines reversed (first two dimensions), and the red line should be at zero (the third dimension, which is the average of the first two).

Nonlinear transformations

In the previous section, you learned how to build a linear brain. The real world is filled with nonlinearities, however, and so dealing with it often requires nonlinear computation. In this tutorial, we compute nonlinear functions by, surprisingly, finding linear decodings. In fact, nonlinear decoding can be thought of as nothing more than an ‘alternate’ decoding of a represented value, which we also linearly decode. In this tutorial we only consider decoding using linear combinations of *neural responses* but, as discussed later in section 4.2.2, combinations of *dendritic responses* may allow for even more efficient computation of nonlinear transformations.

- Create a new network called ‘Nonlinear Function’.
- Inside that network, create an ensemble with the ‘default’ template and name it ‘X’ (it should have 100 neurons representing 1 dimension with a radius of 1.0).
- Create a constant function input for this ensemble and connect it to the ensemble (remember to create a decoded termination with a connection weight of 1.0 on the ‘X’ ensemble and a ‘tauPSC’ of 0.02).

As in the preceding section of this tutorial, we are starting with a scalar ensemble for the sake of simplicity. Now we will square the value represented in ‘X’ and represent the squared value in another ensemble.

- Create an ensemble with the ‘default’ template named ‘result’.
- Create a termination on the ‘result’ ensemble (weight of 1.0, tauPSC 0.02).

Note that we are not using the termination to calculate the transformation. This is because the termination can only scale the decoded input by a constant factor, leading to linear transformations such as those in the first part of this tutorial. Non-linear transformations are made with decoded *origins*.

- Drag an *Origin* from the template bar onto the ‘X’ ensemble.
- In the dialog box that appears, name the origin ‘square’, set *Output Dimensions* to 1, and click *Set Functions*.
- Select *User-defined Function* from the drop-down list of functions and click *Set*.

The ‘User-defined Function’ dialog box that is now displayed allows you to enter arbitrary functions for the ensemble to approximate. The ‘Expression’ line can contain the variables represented by the ensemble, mathematical operators, and any functions specified by the ‘Registered Functions’ drop-down list.

- In the *Expression* field, type ‘ $x0*x0$ ’.

‘ $x0$ ’ refers to the scalar value represented by the ‘X’ ensemble. In an ensemble representing more than one dimension, the second variable is ‘ $x1$ ’, the third is ‘ $x2$ ’, and so forth. Returning to the preceding expression, it should be clear that it is multiplying the value in the ‘X’ ensemble by itself, thus computing the squared value.

- Click OK to close dialogs until the decoded origin is created.
- Drag from the ‘square’ origin to the ‘input’ termination to create a projection between the ‘X’ and ‘result’ ensembles.

The projection that has just been made completes the set up for computing a non-linear transformation. The ‘X’ and ‘result’ ensembles have been connected together and Nengo automatically sets connection weights appropriately to match the specified function (mathematical details can be found in section A.2 of the appendix).

- Open the *Interactive Plots* window.

- Right-click ‘result’ and display its value.
- Right-click your input and display its control.
- Right-click ‘X’ and select $X \rightarrow \text{value}$.
- Right-click ‘X’ and select $\text{square} \rightarrow \text{value}$.

Note that, unlike the ‘result’ ensemble, the ‘X’ ensemble doesn’t have just one value that can be displayed. Instead, it has the representational decoding of ‘X’, as well any transformational decoding we have specified, in this case ‘square’.

- Run the simulation. To demonstrate it is computing the square, move the slider as smoothly as possible between -1 and 1. You should see a line in the ‘X’ value and a parabola in the ‘square’ and ‘result’ values.

Remember that these decoded values are not actually available to the neurons, but are ways for us to visualize their activities. However, if we only examine the relationship between the input and the spike responses in the output, we will see that the firing rates of the output neurons are in an approximate ‘squaring’ relation (or its inverse for ‘off’ neurons) to the input values. This can be seen by examining the spike raster from the ‘result’ population while moving the slider smoothly between -1 and 1.

This tutorial has so far covered linear transformations of scalar variables, linear transformations with vectors, and nonlinear transformations of scalars. The final step is to extend nonlinear transformations to vector representations. As in the scalar case, a nonlinear transformation of a vector is implemented simply with an alternate linear decoding of a neural ensemble (as illustrated for the scalar case in figure 2.9 and derived mathematically in appendix A.2). Crucially, this means that we can understand nonlinear functions of multiple scalars (e.g., $x \times y$) as a linear decoding of a higher-dimensional population. Let us examine this in practice:

- In the ‘Nonlinear Functions’ network, create a new ‘default’ ensemble named ‘Y’.
- Add a decoded termination to the ‘Y’ ensemble (weight of 1, ‘tauPSC’ of 0.02).
- Add a new constant function input named ‘input Y’ and project it to the termination on ensemble ‘Y’.

- Now create a ‘default’ ensemble named ‘2D vector’ and give it 2 dimensions.
- Create a termination named ‘x’ on the ‘2D vector’ ensemble. Set the dimension to 1, the weight matrix to [1, 0], and ‘tauPSC’ to 0.02.
- Create a termination named ‘y’ on the ‘2D vector’ ensemble. Set the dimension to 1, the weight matrix to [0, 1], and ‘tauPSC’ to 0.02.
- Connect the ‘x’ origins of the ‘X’ and ‘Y’ ensembles to the ‘x’ and ‘y’ terminations on ‘2D vector’ respectively.

The value from the ‘X’ ensemble will be stored in the first component of the 2D vector and the value from the ‘Y’ ensemble will be stored in its second component. At this point, there is no output for the network. We are going to represent the product of the two components of the ‘2D vector’ ensemble in the ‘result’ ensemble. To do this we cannot simply use the default ‘x’ decoded origin on the ‘2D vector’ ensemble. We need to create an alternate decoding of the contents of ‘2D vector’.

- Drag a new decoded origin onto the ‘2D vector’ ensemble. Name this origin ‘product’, give it 1 output dimension, and click *Set Functions*.
- In the *Function 0* drop-down menu, select *User-defined Function* and click *Set*.
- Set *Expression* to ‘ $x0*x1$ ’. *Input Dimensions* should be set to 2.

As mentioned earlier, $x0$ refers to the first component of the vector and $x1$ to the second. Given this expression, Nengo generates decoders that yield an approximation of the two components multiplied together.

- Click ‘OK’ to complete all the open dialogs.
- Disconnect the ‘X’ and ‘result’ ensembles by dragging the ‘result’ termination away from the ensemble. To eliminate the hanging connection entirely, right click it and select ‘remove’.
- Create a projection from the ‘product’ origin on ‘2D vector’ to the termination on ‘result’.

This completes the network. The ‘2D vector’ population receives scalar inputs from two populations; stores these inputs separately as two components of a vector, and has its activity decoded as a nonlinear transformation in its projection to the ‘result’ ensemble.

- Click the *Interactive Plots* icon, and show the ‘X’, ‘Y’, and ‘result’ values. Show the ‘input’ and ‘input y’ sliders.

Move the sliders to convince yourself that the ‘result’ is the product of the inputs. For instance, recall that 0 times anything is 0, the product of two negative numbers is positive, the product of opposite signs is negative, and so on. You may have noticed that when computing 1 times 1, the answer is slightly too low. This is because with the two inputs set to maximum, the 2D population becomes saturated. This saturation occurs because the population represents a unit circle, which does not include the point [1, 1]. The farthest diagonal point it includes is $[\sqrt{2}/2, \sqrt{2}/2]$. To remove this saturation you need to increase the radius of the 2D population.

We can also examine the effects of changing the number of neurons on the quality of the computation.

- In the *Network Viewer*, select the ‘2D vector’ ensemble, open the *Inspector*, double-click ‘i neurons’, then change the number of neurons from 100 to 70.
- Test the network in *Interactive Plots* again.

Note that multiplication continues to work well with only 70 neurons in the vector population, but does degrade as fewer neurons are used. I appeal to this fact in later chapters when estimating the number of neurons required to compute more complex nonlinear operations.

Chapter 4

Biological cognition – syntax

4.1 Structured representations

To this point, we have seen how we can construct semantic pointers: compressed high-dimensional vectors that carry information about the statistical structure of some domain. Constructing this general kind of representation has been accomplished for many years by connectionist neural networks. And, they have exploited such representations to great effect. However, the main complaint against connectionists remains: there is little evidence that these representations are helpful for explaining complex – truly cognitive – behavior. It has been forcefully argued that structured representations, like those found in natural language, are essential for explanations of a wide variety of cognitive behavior (Anderson, 2007). As I discuss in more detail in chapter 8, researchers have often moved from this fairly uncontroversial observation to much stronger claims regarding essential properties of these structured representations (e.g., that they are compositional). For the time being, I will remain largely agnostic about these stronger claims. Instead, I take the ultimate arbiter of whether our theories about such representations is a good one to be the behavioral data. The purpose of this chapter is to suggest how semantic pointers can be used in structured representations that are able to explain that data – with no additional assumptions about the essential features of structured representations used for cognition. The main behavioral target for this chapter will be the Raven’s Progressive Matrices, a test of general fluid intelligence that demands inferring structural patterns from structured representations (see section 4.6). First, however, let me consider structured representations in general.

A typical structured representation is given by natural language utterances like “The dog chases the boy.” The structure underlying this representation is the grammar of English, and the component parts are individual words. In an artificial language, like those typically used in computers, such a phrase may be represented with the structured representation *chases(dog, boy)*. Such a representation is structured because the position in which each term appears determines its grammatical role. The “verb” is first, the “agent” is second and the “theme” is third. If the order of the elements changes, so does the meaning of the structured representation.

The fact that entire concepts, denoted by words, can be moved into different roles has historically posed a serious challenge for approaches that think of the representation of concepts as being patterns in a vector space. In addition, because the kind of relation that the concept enters into changes depending on which role it is in, it is not immediately obvious how simple associative relationships can be used to capture such structures. The first of these challenges targets distributed connectionist models, and the second targets both localist and distributed connectionist models. In essence, both challenges boil down to the problem of determining how to take the preferred kind of representation of concepts, and construct a new representation that combines multiple concepts playing different kinds of structural, or syntactic, roles. That is, how can concept representations be bound to roles within a structure? As Barsalou (1999, p. 643) has noticed:

It is almost universally accepted now that representation schemes lacking conceptual relations, binding, and recursion are inadequate.

Consequently, it has been seen as a major challenge for neurally inspired architectures and theories of cognition to address the issue of how structure can be represented. For instance, Barsalou’s own perceptual symbol system theory, while addressing semantics well, has been criticized because the theory (Edelman and Breen, 1999, p. 614)

leaves the other critical component of any symbol system theory – the compositional ability to bind the constituents together – underspecified.

Unfortunately, Barsalou’s discussion of “frames”, while partly addressing the issue, does not describe how neural representations can be bound into such structures. So, despite recognizing the need for building such structures with binding

in neural accounts, no precise computational suggestions have been broadly accepted – and no biologically realistic accounts (i.e. that use noisy spiking neural networks) have been offered.

4.2 Neural binding

4.2.1 Without neurons

Connectionists, being concerned with cognitive modelling, have been working on ways of binding multiple representations since the 1990s. Perhaps the best known early solution is that offered by Paul Smolensky (1990). He suggested that a kind of vector multiplication called a “tensor product” could perform the necessary binding. Despite the exotic sounding name, tensor products are a straightforward solution to the problem of trying to multiply two vectors, and then later get back one of the vectors given the other (i.e. trying to “divide” them). It is obvious how to do this if we work with scalars: $6 \times 2 = 12$ so, $12 \times \frac{1}{2} = 6$ and $12 \times \frac{1}{6} = 2$. Notice that to compute the unknown element given the other two (i.e., the answer and one of the multiples), we needed to “invert” the known element and multiply. The question is, what happens if the mathematical objects we have are vectors? What is $(6, 3) \times (2, 5, 4)$?

This is where tensor products come in. They define what we mean by “ \times ” in the context of vectors, so we can have operations that mimic what we are used to with scalars. To distinguish this operation from regular multiplication, the “ \otimes ” sign is typically used. If the vectors we wanted to multiply were the same length, we might not have to bother with tensor products – we could just multiply the elements. Unfortunately, we can’t just multiply if there are more elements in one vector than the other. So, the next obvious solution is to multiply all the elements, which gives the tensor product:

$$\begin{pmatrix} 6 \\ 3 \end{pmatrix} \otimes (2 \ 5 \ 4) = \begin{pmatrix} 6 \times 2 & 6 \times 5 & 6 \times 4 \\ 3 \times 2 & 3 \times 5 & 3 \times 4 \end{pmatrix} = \begin{pmatrix} 12 & 30 & 24 \\ 6 & 15 & 12 \end{pmatrix}.$$

This way, if we are given the final matrix and one of the vectors, it is clear how we can recover the other vector.

The relevance of tensor products to binding was evident to Smolensky: if we represent items with vectors, and structures with collections (sums) of bound vectors, then we can use tensor products to do the binding, and still be able to “unbind” any of the constituent parts. Consider the previous example of *chases* (dog,

boy). To represent this structure, we need vectors representing each of the concepts and vectors representing each of the roles, which for this representation includes (I write vectors in bold): **dog**, **chase**, **boy**, **verb**, **agent**, **theme**. In the context of the SPA, each of these would be a semantic pointer.

To bind one role to one concept, we can use the tensor product:

$$\mathbf{b} = \mathbf{dog} \otimes \mathbf{agent}.$$

Now, the resulting vector **b** can be used to recover either of the original vectors given the other. So, if we want to find out what role the **dog** is playing, we can unbind it by inverting it and multiplying, just as in the scalar case:

$$\mathbf{b} \otimes \mathbf{dog}^{-1} = \mathbf{agent}.$$

A technical issue regarding what we mean by “inverting” a vector arises here. Suffice it to say for present purposes that there is a good definition of the necessary inversion called the pseudo-inverse, and I will return to this issue later in a related context.

Given a means of binding and unbinding vectors, we also need a means of collecting these bindings together to construct structured representations. Smolensky adopted the standard connectionist approach of simply summing to conjoin vectors. This results in a straightforward method for representing sentences. For example, the structured vector representation of the proposition “The dog chases the boy” is:

$$\mathbf{P} = \mathbf{verb} \otimes \mathbf{chase} + \mathbf{agent} \otimes \mathbf{dog} + \mathbf{theme} \otimes \mathbf{boy}.$$

In the case where all of the role vectors are linearly independent (no one vector is a weighted sum of the others), decoding within this structure is the same as before – just take the tensor product of **P** with the inverse of the role. These tensor product representations are quite powerful for representing structure with vectors as elements of the structure. They can be combined recursively to define embedded structures, and they can be used to define standard LISP-like operations that form the basis of many cognitive models, as Smolensky shows.

However, tensor products were never broadly adopted. Perhaps this is because many were swayed by Fodor and McLaughlin’s contention that this was “merely” a demonstration that you could implement a standard symbolic system (Fodor and McLaughlin, 1990). As such, it would be unappealing to symbolists because it just seemed to make models more complicated, and it would be unappealing to connectionists because it seemed to lose the neural roots of the constituent representations. For instance, Barsalou (1999, p. 643) comments that

Early connectionist formulations implemented classic predicate calculus functions by superimposing vectors for symbolic elements in predicate calculus expressions (e.g., Pollack 1990; Smolensky 1990; van Gelder 1990). The psychological validity of these particular approaches, however, has never been compelling, striking many as arbitrary technical attempts to introduce predicate calculus functions into connectionist nets.

I suspect that both of these played a role. But there is also an important technical limitation to tensor products, one which Smolensky himself acknowledges (1990, p. 212):

An analysis is needed of the consequences of throwing away binding units and other means of controlling the potentially prohibitive growth in their number.

He is referring here to the fact that the result of binding two vectors of lengths n and m respectively, gives a new structure with nm elements. If we have recursive representations, the size of our representations will grow quickly out of hand. That is, they will not *scale* well. This also means that, if we are constructing a network model, we need to have some way of handling objects of different sizes within one network. We may not even know how “big” a representation will be before we have to process it. Again, it is not clear how to scale a network like this.

I believe it is the scaling issue more than any other that resulted in sparse adoption of Smolensky’s suggestion in the modeling community. However, modelers began to attack the scaling problem head on, and now there are several methods for binding vectors that avoid the issue. For binary representations, there are Penti Kanerva’s Binary Spatter Codes or BSCs (Kanerva, 1994). For continuous representations, there are Tony Plate’s Holographic Reduced Representations or HRRs (Plate, 1994). And, also for continuous representations, there are Ross Gayler’s Multiply-Add-Permute or MAP representations (Gayler, 1998). In fact, all of these are extremely similar, with BSCs being equivalent to a binary version of HRRs and both HRRs and BSCs being a different means of compressing Smolensky’s tensor product (see figure 4.1).

Noticing these similarities, Gayler suggested that the term “Vector Symbolic Architectures” be used to describe this class of closely related approaches to encoding structure using distributed representations (Gayler, 2003). Like Smolensky’s tensor product representations, each VSA identifies a binding operation and

$$\begin{aligned}
 \textcircled{1} \quad & \begin{pmatrix} A \\ B \\ C \\ D \end{pmatrix} \otimes \begin{pmatrix} E \\ F \\ G \\ H \end{pmatrix} = \begin{pmatrix} AE & AF & AG & AH \\ BE & BF & BG & BH \\ CE & CF & CG & CH \\ DE & DF & DG & DH \end{pmatrix} \\
 \textcircled{2} \quad & \begin{pmatrix} A \\ B \\ C \\ D \end{pmatrix} \times \begin{pmatrix} E \\ F \\ G \\ H \end{pmatrix} = \begin{pmatrix} AE \\ BF \\ CG \\ DH \end{pmatrix} \\
 \textcircled{3} \quad & \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \oplus \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} \\
 \textcircled{4} \quad & \begin{pmatrix} A \\ B \\ C \\ D \end{pmatrix} \circledast \begin{pmatrix} E \\ F \\ G \\ H \end{pmatrix} = \begin{pmatrix} AE + BH + CG + DF \\ AF + BE + CH + DG \\ AG + BF + CE + DH \\ AH + BG + CF + DE \end{pmatrix}
 \end{aligned}$$

Figure 4.1: Various vector binding operations. (1) Tensor products are computed as a multiplication of each element of the first vector with each element of the second vector. (2) Piecewise multiplication, used in Gayler’s Multiply-Add-Permute representation, multiplies each element of the first vector with one corresponding element in the second vector. (3) Binary Spatter Codes (BSCs) combine binary vectors using an exclusive or (XOR) function. (4) Circular convolution is the binding function used in Holographic Reduced Representations (HRRs). An overview of circular convolution and a derivation of its application to binding and unbinding can be found in section B.1.

a conjoining operation. Unlike tensor products, the newer VSAs do not change the dimensionality of representations when encoding structure.

There is a price for not changing dimensionality during binding operations: degradation of the structured representations. If we constantly encode more and more structure into a vector space that does not change in size, we must eventually lose information about the original structure. Representations that slowly lose information in this manner have been called “reduced” representations, to indicate that there is less information in the resulting representation than in the original structures they encode (Hinton, 1990). Thus the resulting structured representations does not explicitly include all of the bound components: if they did, there would be no information reduction. This information reduction has important consequences for any architecture, like the SPA, that employs them.

For one, this means that such an architecture cannot be a classical architecture. In short, the reason is that the representations in such an architecture are not perfectly compositional. This is a point I will return to in chapter 10. A more technical point is that, as a result of losing information during binding, the unbinding operation returns an *approximation* of the originally bound elements. This means that the results of unbinding must be “cleaned-up” to a vector that is familiar to the system. Consequently, use of a VSA imposes a functional demand for a clean-up memory, i.e., a memory that maps noisy versions of allowable representations onto those allowable representations. In the previous example, all of the vectors in the structure (i.e., **boy**, **dog**, etc.) would be in the clean-up memory, so results of unbinding could be “recognized” as one of the allowable representations. I return to this important issue in section 4.5.

In any case, what is clear even without detailed consideration of clean-up, is that there are limits on the amount of embedded structure that a reduced representation can support before noise makes the structure difficult to decode, resulting in errors. As we will see, the depth of structure that can be encoded depends on the dimensionality of the vectors being used, as well as the total number of symbols that can be distinguished. Ultimately, however, I think it is this capacity for error that makes the SPA psychologically plausible. This is because the architecture does not define an idealization of cognitive behavior like classical architectures tend to, but rather specifies a functional, implementable system that is guaranteed to eventually fail. I take it that capturing the failures as well as the successes of cognitive systems is truly the goal of a cognitive theory. After all, many experiments in cognitive psychology are set up to increase the cognitive load to determine what kinds of failure arise.

Now to specifics. In this book I adopt Plate’s HRR representations because

they work in continuous spaces, as neurons do, and because he has established a number of useful theoretical results regarding their capacity limitations (Plate, 2003). Still, most of the examples I provide do not depend on the VSA chosen. In fact much work remains to be done on the implications of this choice. In particular, it would be interesting to determine in more detail the expected behavioral differences between HRRs and MAPs. But, such considerations are beyond the scope of the present discussion. It is worth noting, however, that the structure of the SPA itself does not depend on the particular choice of a VSA. I take any method for binding that depends on a piecewise (i.e., local) nonlinearity in a vector space to be able to support the main ideas behind the SPA. Nevertheless, it is necessary to choose an appropriate VSA in order to provide specific examples.

In the remainder of this chapter, I turn to the consideration of implementing the architectural components necessary to support HRR processing in a biologically realistic setting. First, I consider binding and unbinding. I then turn to demonstrating how these operations can be used to manipulate (as well as encode) structured representations, and I show how a spiking neural network can learn such manipulations. I then return to the issue of implementing a clean-up memory in neurons, and discuss some capacity results that suggest the SPA will scale appropriately. Finally, I describe a detailed SPA model that performs structural inference, mimicking human performance in the Raven’s Progressive Matrices test of general fluid intelligence.

4.2.2 With neurons

The relevance of VSAs to what the brain does is far from obvious. As Bob Hadley (2009) has recently noted: “It is unknown whether or how VSAs could be physically realized in the human brain” (p. 527). However, to perform binding with neurons, we can combine the above characterization of vector binding with my earlier characterizations of vector representation (section 2.5) and transformation (section 3.8). Notably, each of the VSA methods for binding rely on an element-wise nonlinearity (see figure 4.1), so we need to perform a nonlinear transformation (the same one we did in the last tutorial; section 3.8). In the case of HRRs in particular, we need to perform a linear transformation before the element-wise nonlinearity.

Specifically, the binding operation for HRRs is called “circular convolution” and has the operator symbol “ \circledast ”. The details of circular convolution can be found in appendix B.1. What is important for our purposes is that the binding of any two

vectors **A** and **B** can be computed by

$$\mathbf{C} = \mathbf{A} \circledast \mathbf{B} = \mathbf{F}^{-1}(\mathbf{F}\mathbf{A}.\mathbf{F}\mathbf{B})$$

where “.” is used to indicate element-wise multiplication of the two vectors. The matrix **F** is a linear transformation that is the same for all vectors of a given dimension. The resulting vector **C** has the same number of dimensions as **A** and **B**. The network that computes this binding is simply a standard, two-layer feedforward network (see figure 4.2).

Because we have already seen examples of both linear transformation and the multiplication of two numbers, it is straightforward to build this network in Nengo. The tutorial at the end of this chapter gives detailed instructions on building a binding network to encode structured representations (section 4.8). Figure 4.3 shows the results of binding various ten-dimensional vectors using the circular convolution method.

To unbind vectors, the same operation can be used. However, as explained earlier we need to compute the inverse of one of the bound vectors to get the other. Conveniently, for HRRs a good approximation to the inverse can be found by a simple linear transformation (see appendix B.1). Calling this transformation **S**, we can write the unbinding of any two vectors as

$$\mathbf{A} \approx \mathbf{C} \circledast \mathbf{B}^{-1} = \mathbf{F}^{-1}(\mathbf{F}\mathbf{C}.\mathbf{F}\mathbf{S}\mathbf{B}).$$

Recall that the result is only approximate and so must be cleaned-up, as I discuss in more detail shortly (section 4.5).

From a network construction point-of-view, we can redeploy exactly the same network as in the binding case, and simply pass the second vector through the transformation **S** before presenting it to the network (this amounts to changing the weights between the B and Bind layers in figure 4.2). Figure 4.4 demonstrates, using the output of the network in figure 4.3, that vectors can be effectively unbound in this manner.

It is perhaps unsurprising that we can implement binding and unbinding in neural networks. However, this does not allay concerns, like those expressed by Barsalou, that this is merely a technical trick of some kind that has little psychological plausibility. I believe the psychological plausibility of these representations needs to be addressed by looking at larger scale models, the first of which we will see later in this chapter (section 6.6). There I show how this operation is able to capture psychological performance on a fluid intelligence test. Additional examples are presented in later chapters.

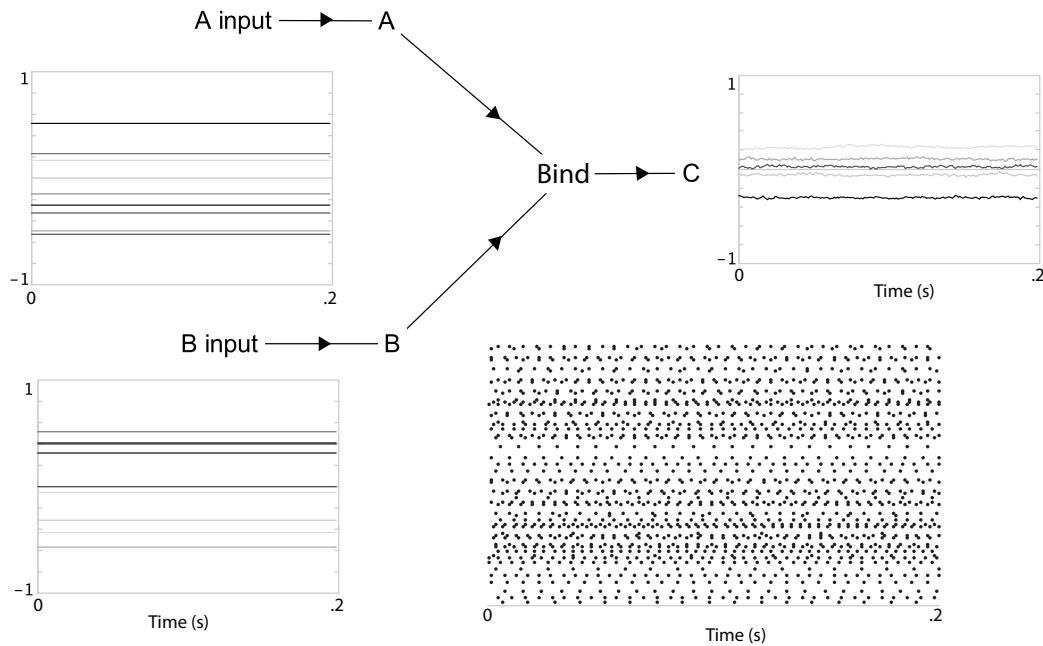


Figure 4.2: Network architecture of a binding network. The network is a simple two-layer, feedforward network, as there are connection weights only into the Bind and C layers. In this particular network, A, B, and C have 300 neurons and Bind has 2400 neurons. This network is binding two 10-dimensional vectors (shown in the graphs below the input nodes), projected into the A and B neurons. These neurons project to the Bind layer, which forms a representation that allows the computation the necessary nonlinearities. The spiking activity is shown for the Bind neurons. This activity drives the C layer, which extracts the binding from that representation. The decoding of the bound vector is shown in the rightmost graph, as another 10-dimensional vector. The results for 200ms of simulation time are shown.

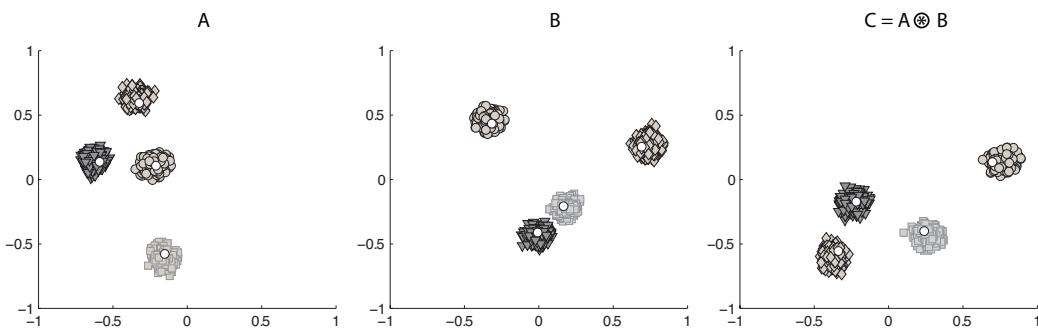


Figure 4.3: Binding vectors with spiking neurons. A ten-dimensional vector **A** is bound with a ten-dimensional vector **B** to produce a ten-dimensional vector **C** (see figure 4.2). This figure shows four separate instances of binding. In each instance, instantaneous decoded samples are drawn every 10ms from a neural representation of vectors **A**, **B**, and **C** over a period of one second. This forms a collection of points that are plotted with a unique marker (e.g., the light gray circles in the first plot were bound with the light gray circles in the second plot resulting in the group of light gray circles in the third plot). These samples form a “cloud” because of the neural variability over time due to spiking and other sources of noise. To visualize the ten-dimensional vectors used in this process, vectors are reduced to two-dimensional points using principle component analysis to preserve as much of the variability of the higher-dimensional data as possible. Similar vectors thus map to points that are close together in these plots. Mean values of the groups of points are given by white circles.

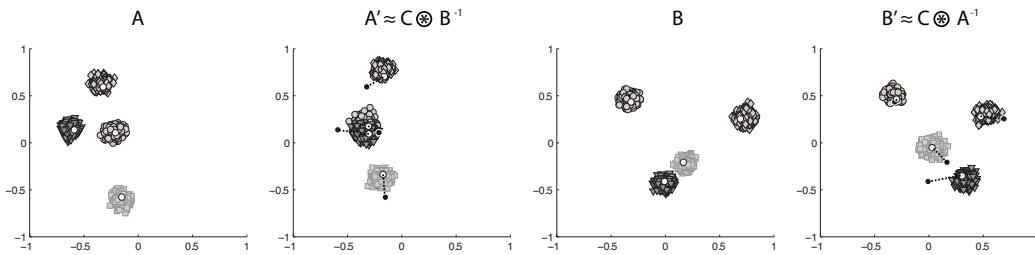


Figure 4.4: Unbinding vectors with neurons. The reconstructed vectors **A** and **B** were obtained by using the transformations $\mathbf{A} \approx \mathbf{C} \otimes \mathbf{B}^{-1}$ and $\mathbf{B} \approx \mathbf{C} \otimes \mathbf{A}^{-1}$ respectively. This figure is using the same initial vectors and graphing methods as figure 4.3 and the original **A** and **B** vectors have been reproduced alongside the reconstructed vectors for ease of comparison. In the graphs of the reconstructed vectors **A'** and **B'**, black circles indicate the original average values of the vectors and the dotted lines indicate the decoding error.

We might also be concerned about the neural plausibility of this approach. Three possible sources of concern regarding the neural plausibility of this implementation are: 1) the specific nonlinearity employed; 2) the connectivity of the network; and 3) determining how well the network scales to large lexicons. Let me consider each in turn.

The nonlinearity that is needed for the binding operation is computing the product of two scalar input signals. We have demonstrated in the tutorials how this can be done accurately using only linear dendrites, and with a small number of neurons (section 3.8). The assumption of linear dendrites is, in fact, a limiting assumption, not a beneficial one. We make it here because it is the “textbook” assumption about dendritic processing. However, much recent work strongly suggests that pyramidal cells (the most common cortical cells) have nonlinear dendritic interactions.

Consequently, an alternative model of cortical pyramidal cells has been proposed based on physiological studies reporting that dendritic spikes evoked by focal synaptic stimulation often remain confined to a single dendritic branch (Schiller et al., 2000), and that within-branch stimulation from two electrodes can produce sublinear, linear or superlinear responses, depending on the placement and timing of the stimulation (Polsky et al., 2004). This alternative model suggests that thin basal dendrites of pyramidal cells, where the majority of excitatory synapses are located, act as processing subunits, where the activity each of branch is determined by its own sigmoidal nonlinearity (Poirazi et al., 2003). A second stage of

processing *then* occurs at the soma, similar to the standard linear model, where the input is a weighted summation of the subunit activities that determines the neuron's activity. This two-stage process has been observed in a study of subthreshold synaptic integration in basal dendrites of neocortical cells, where stimulation within a single terminal dendritic branch evoked a sigmoidal subunit nonlinearity, but the activity summed linearly when different terminal branches were stimulated (Schiller et al., 2000). In effect, this provides evidence that a single neuron may have the processing power of a two-layer linear network (although the connectivity is more restricted).

If our binding networks employed such nonlinearities, we could eliminate the middle layer currently used to compute the nonlinearity. Consequently, the binding networks would use significantly fewer neurons than in the examples I consider here. So, the examples I present are worst case scenarios for the number of neurons that are needed to generate structured representations. In section 5.5 I consider a model that uses nonlinear neurons.

The second concern regarding neural plausibility is connectivity. That is, we know that in general cortical circuits are fairly locally and sparsely connected (Song et al., 2005; Hellwig, 2000; Lund et al., 1993). Do these binding networks respect this constraint? To begin, we need to be more accurate about what we mean by locally connected. In their anatomical work that involves microinjections of a tracer across several areas of monkey cortex, the Levitt group has found that small injections of $100\mu\text{m}$ (i.e., 0.1mm) spread to areas of approximately 3mm by 3mm or slightly larger (Lund et al., 1993). There is an interesting structure to this projection pattern, in which there are patches of about $300\text{-}400\mu\text{m}$ in diameter separated by spaces of a similar size. These patches correspond well to the sizes of dendritic fields of the pyramidal cells in these areas. Given that there are 170,000 neurons per mm^2 of cortex,¹ this corresponds to about 85,000 neurons within the needed distance for connectivity to a given cell. Consequently, a single neuron could easily be connected to any other neurons in a network of about 85,000 neurons. So “local” connectivity seems to be local within $400\mu\text{m}$ or about 85,000 neurons, with sparser connections between such patches. This pattern of connectivity has been found throughout visual, motor, somatosensory and prefrontal cortex, and across monkeys, cats, and rodents.

The plausibility of connectivity is closely tied to the third issue of scaling. To

¹This estimate is based on the data in Pakkenberg and Gundersen (1997) which describes differences in density across cortical areas, as well as the effects of age and gender. This value was computed from Table 2 for females, using a weighted average across cortical areas.

determine if a binding network can be plausibly fit within this cortical area, it is important to know what the dimensionality of the bound vectors is going to be. It is also important to know how the size of the binding network changes with dimensionality. I discuss in detail the dimensionality of the bound vectors in section 4.5, because the necessary dimensionality is determined by how well we can clean up vectors after binding. There I show that we need vectors of about 500 dimensions in order to capture large structures with an adult-sized vocabulary. It is also clear that the binding network itself will scale linearly with the number of dimensions because the only nonlinearity needed is the element-wise product. Thus, one additional dimension means computing one additional product. Recall from section 3.8 that 70 neurons per multiplication results in good quality estimates of the product, and two multiplications are needed per dimension. Taken together, these considerations suggest that about 70,000 neurons are needed to bind two vectors. Coupled with the previous local connectivity constraint, this suggests that binding networks of the appropriate size to capture human-like structured representations can be constructed in this manner.

Recalling that there are about 170,000 neurons per mm^2 of cortex suggests that we need about 0.7 mm^2 of cortex to perform the necessary binding. The projection data suggests that local projections cover at least 9 mm^2 of cortex, so the binding layer can fit comfortably within this area. If we want to include the two input populations, the binding layer and the output population with similar assumptions, we would need at most 2 mm^2 of cortex. Recall that the unbinding network requires the same resources. Consequently, these networks are consistent with the kind of connectivity observed in cortex. As well, the architecture scales linearly so this conclusion is not highly sensitive to the assumptions made regarding the dimensionality of the representations. It is worth noting that if we allow dendritic nonlinearities, the binding layer is not needed, so the entire network would require less cortex still.

Together, these considerations suggest that the vector binding underlying the SPA can be performed on the scale necessary to support human cognition in a neurally plausible architecture. I return to considerations of the neural plausibility of the SPA in chapter 9.4.

4.3 Manipulating structured representations

To this point, my discussion on syntax in the SPA has focused on how structures can be encoded and decoded using binding. However, to make such represen-

tations cognitively useful they must be *manipulated* to support reasoning. An interesting feature of VSA representations is that they can often be manipulated without explicitly decoding the elements. This has sometimes been called “holistic” transformation, since a semantic pointer representing a sentence can be manipulated in a way that affects the entire sentence at once without individually manipulating the entities it encodes.

Several researchers have demonstrated transformations of structured representations using distributed representations, although experiments have been limited to fairly simple transformations and the transformations tend to be hand-coded.² Here, I begin by considering simple, hand-coded transformations as well, but demonstrate in the next section how a broad class of transformations can be learned.

Conveniently, performing transformations using VSAs relies on binding as well. Consequently, the same network architecture shown in figure 4.2 can be used to perform syntactic manipulation. To see how these manipulations work, let us return to the *chases(dog, boy)* example, where this sentence is encoded as

$$\mathbf{P} = \mathbf{verb} \circledast \mathbf{chase} + \mathbf{agent} \circledast \mathbf{dog} + \mathbf{theme} \circledast \mathbf{boy}.$$

Recall that given this encoding, we can decode elements by binding it with the appropriate inverses. For instance, if we multiply \mathbf{P} by the approximate inverse of **chase** (i.e., **chase'**) we get:

$$\begin{aligned} \mathbf{chase}' \circledast \mathbf{P} &= \mathbf{chase}' \circledast (\mathbf{verb} \circledast \mathbf{chase} + \mathbf{agent} \circledast \mathbf{dog} + \mathbf{theme} \circledast \mathbf{boy}) \\ &= \mathbf{chase}' \circledast \mathbf{verb} \circledast \mathbf{chase} + \mathbf{chase}' \circledast \mathbf{agent} \circledast \mathbf{dog} + \mathbf{chase}' \circledast \mathbf{theme} \circledast \mathbf{boy} \\ &= \mathbf{verb} + \mathbf{noise} + \mathbf{noise} \\ &\approx \mathbf{verb} \end{aligned}$$

So we can conclude that the “chase” plays the role of a verb in \mathbf{P} .

A few comments are in order. First, this is an algebraic demonstration of the effects of binding **chase'** to \mathbf{P} , and does not require the decoding of any of the other sentence elements (i.e., we don’t have to “remove” the agent and theme). Second, this demonstration takes advantage of the fact that \circledast can be treated like

²Niklasson and van Gelder (1994) demonstrated how to transform a logical implication into a disjunction, as did Plate (1994); Pollack (1990) transformed reduced representations of the form *loved(x,y)* to the form *loved(y,x)*; Legendre et al. (1994) showed how to transform representations for active sentences into representations for passive sentences (and vice-versa).

regular multiplication as far as allowable operations are concerned (i.e., it is communicative, distributive and associative). Third, I have used the ' sign to indicate the pseudo-inverse and distinguish it from a true inverse. This pseudo-inverse is computed by a simple linear operation (a shift), as mentioned earlier (section 4.2.2). Fourth, **chase'** \otimes **verb** \otimes **chase** is replaced by **verb** because a vector times its inverse is equal to one. Or, more accurately, a vector times its pseudo-inverse is approximately equal to one, and any difference is absorbed by the **noise** term. Finally, the last two terms are treated as noise because circular convolution has the important property that it maps the elements to an approximately orthogonal result. That is, the result $\mathbf{w} = \mathbf{x} \otimes \mathbf{y}$ is unlike either \mathbf{x} or \mathbf{y} . By “unlike” I mean that the dot product (a standard measure of similarity) between \mathbf{w} and \mathbf{x} or \mathbf{y} is close to zero. Consequently, when we add in these new but unfamiliar items from the last two terms, they will slightly blur the correct answer, but not make it unrecognizable. In fact, the reason we really think of these as noise is because they will not be similar to any known lexical item (because they are orthogonal to known items), and hence during the clean-up operation will be ignored. Once again, it is clear that clean-up, while not an explicit part of most VSAs, plays a central role in determining what can be effectively represented and manipulated. This is why it is a central part of the SPA.

We could think of this simple decoding as a kind of manipulation of the structure, but we can also perform more extensive manipulations, such as switching roles and changing the relation, in a similar manner. Consider the hand-constructed vector **T**:

$$\mathbf{T} = \mathbf{agent}' \otimes \mathbf{theme} + \mathbf{chase}' \otimes \mathbf{hug} + \mathbf{theme}' \otimes \mathbf{agent}$$

Convolving our original representation of the sentence **P** with **T** produces:

$$\mathbf{T} \otimes \mathbf{P} = \mathbf{verb} \otimes \mathbf{hug} + \mathbf{agent} \otimes \mathbf{boy} + \mathbf{theme} \otimes \mathbf{dog} + \mathbf{noise}.$$

That is, we now have a representation of **hug(boy, dog)**. This occurred because the first term in **T** converts whatever was the **agent** into the **theme** by removing the **agent** vector (since **agent'** \otimes **agent** \approx 1) and binding the result with **theme**. The last term works in a similar manner. The second term replaces **chase** with **hug**, again in a similar manner. The **noise** term is a collection of the noise generated by each of the terms.

For a demonstration of the usefulness of such manipulations, we can consider one of the standard tasks for testing cognitive models: question answering. Figures 4.5 and 4.6 show a network of 10,000 neurons answering four different

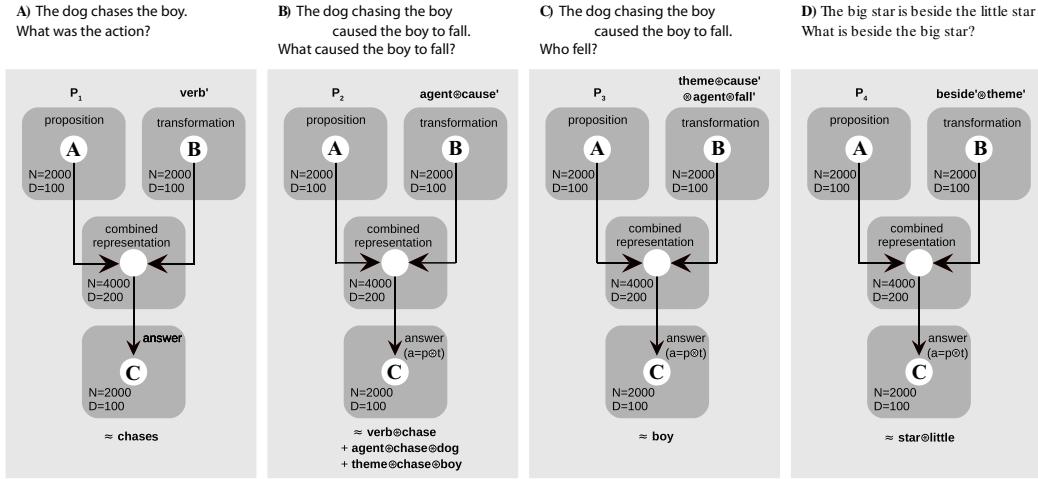


Figure 4.5: A question-answering network applied to four different questions. For each case, the sentence and question are provided as separate inputs (**A** and **B**). Notably, the network is in no way modified during this task, so this one network is capable of answering any question of a similar form using these lexical items. Behavior of the network over time is shown in figure 4.6.

questions about different sentences. For each sentence and question, a different input is provided to the same network for 0.25 seconds. In figure 4.6, the results of clean up are shown over time, so the similarity between the output of the network and possible responses is plotted. The system is considered to have given a correct response if its output is more similar to the correct response than any other possible response.

To understand these results, let us consider each question in more detail. The first case is nearly identical to the first transformation we considered above. Instead of determining the role of **chase**, the network determines what the action of the sentence is.

Turning to the second question, we see that the sentence has more complex, embedded structure. To capture such structures, we need a way to map natural language sentences onto semantic pointers. For the second case in figure 4.5, the roles “agent” and “theme” are multiply instantiated. Consequently, roles need to be tagged with the verb they are roles for, so the items they are bound to do not get confused. For instance, if the roles were not tagged, it would be difficult to distinguish whether **dog** was the agent of **chase** or of **fall**. This kind of role

tagging can be accomplished through binding as follows:

$$\mathbf{P}_2 = \mathbf{verb} @ \mathbf{cause} + \mathbf{agent} @ \mathbf{cause} @ (\mathbf{verb} @ \mathbf{chase} + \mathbf{agent} @ \mathbf{chase} @ \mathbf{dog} + \\ \mathbf{theme} @ \mathbf{chase} @ \mathbf{boy}) + \mathbf{theme} @ \mathbf{cause} @ (\mathbf{verb} @ \mathbf{fall} + \mathbf{agent} @ \mathbf{fall} @ \mathbf{boy})$$

This additional tagging of elements by their verb does not affect previous results (i.e., we could have tagged all roles with the verb). I did not discuss it earlier for the sake of simplifying the exposition. Here, as in other more sophisticated linguistic structures, it is essential to remove this simplification and distinguish different roles in order to preserve the appropriate structure. Using this encoding, we can now answer the question “What caused the boy to fall?”, by convolving \mathbf{P}_2 with $\mathbf{agent} @ \mathbf{cause}'$ and the correct answer is provided.

The third case presents a more challenging kind of question that queries the contents of subphrases that make up the sentence, e.g. “Who fell?”. This question is asked in the third case by determining the $\mathbf{agent} @ \mathbf{fall}$ of the $\mathbf{theme} @ \mathbf{cause}$ all at once. That is,

$$\begin{aligned} & \mathbf{P}_2 @ (\mathbf{theme} @ \mathbf{cause} @ \mathbf{agent} @ \mathbf{fall})' \\ & \approx \mathbf{boy} \end{aligned}$$

The same answer would result from applying the two transformations in series.

Finally, we can consider the fourth case from figure 4.5. Here the system answers the question “What is beside the big star?” for the sentence “The big star is beside the little star.” This example was chosen to demonstrate that there is no inherent “problem of two” for this approach (see section 1.3). This is not surprising since representation of the same object twice does not result in representations that cannot be distinguished. This case is structurally identical to the first case, with the difference that there are modifiers (“big” and “little”) bound to the nouns.

To conclude this section, it is worth noting two important challenges that are raised by considering the manipulation of language-like representations. First, there is the issue of mapping language onto a VSA representation like that used by the SPA. There are important decisions regarding how we should represent complex structure that are insufficiently addressed here. I have employed a representation that has proven useful for several models, but does not have a solid linguistic justification. The second closely related challenge is how natural language questions should be mapped to transformations of those structures. There are multiple possible ways to affect such transformations. For instance, in this chapter I have been considering transformations that are bound to the representations in one step. However, it is quite likely that in many circumstances, the way

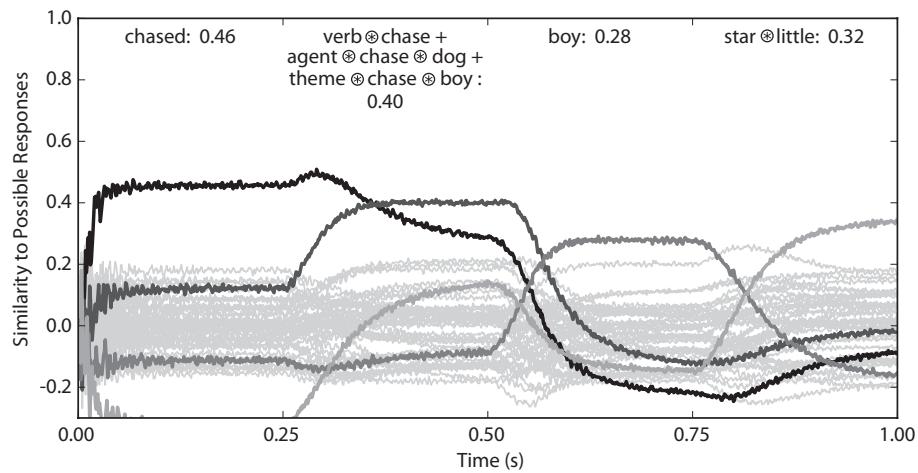


Figure 4.6: The network answering four questions over time. Each question is presented for 0.25 seconds by changing the input to the neural groups representing vectors **A** and **B** in figure 4.5. The resulting spiking behavior of neural group **C** is interpreted as a semantic pointer and compared to the possible responses. This plot shows the similarity of the vector **C** to various possible responses (i.e., the top line in the plot represents the output of a clean-up memory). Dark gray through medium gray lines indicate the similarity to the four correct responses. The lightest gray lines are the similarities of the output with 40 randomly chosen other semantic pointers in the lexicon, representing other possible responses. Since the output is closer to the correct response than any other answer in each case, this network successfully answers all of the questions.

questions are answered is partly determined by how representations are moved through the system. If the question demands deep semantic analysis, for instance, then the employed pointers need to be dereferenced. A simple syntactic manipulation will not suffice. Consequently, the next chapter, on control structures in the SPA, is relevant to how such question-answering is, in fact, performed in a biological system. In short, I want to be clear that the treatment provided here is self-consciously superficial. I do not want to suggest that all reasoning can be accomplished by syntactic manipulation of the kind presented here, even though such manipulation is likely important to much language-based cognition.

4.4 Learning structural manipulations

Unlike most connectionist approaches, the NEF does not rely on a learning process to construct model networks.³ This is demonstrated, for example, by the binding networks derived above. However, learning clearly plays a large role in cognition, and is important for explaining central features of cognition, such as syntactic generalization and fluid intelligence (which I consider in more detail in sections 6.6 and 4.6 respectively).

In my previous example of structural manipulation, as in most past work on syntactic transformations, the transformation vectors \mathbf{T} are hand-picked based on previous knowledge of the vectors, and the ways in which they are combined. This is generally undesirable from a cognitive modelling perspective, since it requires significant design decisions on the part of the modeller, and these tend to be driven by particular examples. It is even more undesirable from a psychological perspective because one of the distinctive features of cognition is the ability to generate new rules given past experience. This, of course, is called *induction*.

Inductive reasoning is the kind of reasoning that proceeds from several specific examples to a general rule. Unlike deductive reasoning, which forms the foundation of classical logic, and guarantees the truth of any correctly derived conclusion, inductive reasoning can be correctly used to generate rules that may

³There is much to recommend not having learning be a sole network design method. It makes it possible to avoid some of the unprincipled aspects of learning, such as how to distinguish example and test elements, how to order the examples, when to turn learning on and off, what learning rates to choose, etc. It is also easier to test specific ideas about what a brain area is doing, to design large-scale and many layered networks, and to take advantage of established empirical facts about the area under study. Of course, if you want to answer certain developmental questions, or questions related to certain kinds of adaptation of the system, careful consideration of learning is unavoidable. I address learning more systematically in chapter 6.

well be false. Nevertheless, induction is perhaps the most common kind of reasoning used both in science, and in our everyday interactions with the world.

In the language of the SPA, performing induction is equivalent to deriving a transformation that accounts for the mapping between several example structure transformations. For example, if I tell you that “the dog chases the boy” maps to “the boy chases the dog”, and that “John loves Mary” maps to “Mary loves John” and then provide you with the following structure: “the bird eats the worm”; you would probably tell me that the expected mapping would give “the worm eats the bird” (semantic problems notwithstanding). In solving this problem, you used induction over the syntactic structure of the first two examples to determine a general structural manipulation that you then apply to a new input. We would like our architecture to do the same.

Intuitively, to solve the above example we are doing something like generating a potential transformation given information only about the pre- and post-transformation structures. We may be able to do this with one example, but the second example can help confirm our initial guess, by ensuring we would derive the same transformation provided that information. In short, we are making sure that if we were given just the pre-transformation structure, we could generate the post-transformation structure. One way to characterize this process is to say we are minimizing the error between our application of the transformation and the post-transformation structure.

Neumann (2001) presented an early investigation into the possibility of learning structural transformations using a VSA. She demonstrated that with a set of example transformations available, a simple error minimization would allow extraction of a transformation vector. In short, she showed that if we had examples of the transformation, we could infer what the transformation vector is by incrementally eliminating the difference between the examples and the transformations we calculated with our estimate of \mathbf{T} (see appendix B.2).

A simple extension to her original rule that allows the transformation to be calculated on-line is the following (Eliasmith, 2004):

$$\mathbf{T}_{i+1} = \mathbf{T}_i - w_i [\mathbf{T}_i - (\mathbf{A}'_i \circledast \mathbf{B}_i)]$$

where i indexes the example, \mathbf{T} is the transformation vector we are trying to learn, w is a weight that determines how important we take the current example to be compared to past examples, and \mathbf{A} and \mathbf{B} are the pre- and post-transformation structured vectors. This rule essentially estimates the transformation vector that would take us from \mathbf{A} to \mathbf{B} , and uses this to update our current guess of the transformation. Specifically, the rule computes the convolution between the inverse of

the pre-transformation vector \mathbf{A} and the post-transformation vector \mathbf{B} . That is, it assumes that $\mathbf{B} = \mathbf{T} \circledast \mathbf{A}$, and hence $\mathbf{T} = \mathbf{A}' \circledast \mathbf{B}$. The result of that computation is subtracted from the current estimate of the transformation vector \mathbf{T}_i to give an error. If that error is zero, the transformation has been learned. If it is not zero, the rule uses that difference to update the current transformation to something different (specifically, to a running average of inferred transformations).

Although simple, this rule is surprisingly powerful, as I show in section 4.6. Before doing this, however, I need to return to the issue of clean-up memory, which is also used in that model.

4.5 Clean-up memory and scaling

The result of applying most transformations to semantic pointers encoding structure will be noisy. This is true for any compressed representation employed in VSAs. In the case of HRRs, the noisy results are in a high-dimensional continuous vector space, and they must be “cleaned up” to the nearest allowable representation. Most typically, we can think of “allowable” representations as those vectors which are associated with words in a lexicon. However, they might also be sentences, subphrases, or other structured representations that are pertinent to the current inferential context. Ultimately, what makes a representation allowable is its inclusion in a clean-up memory.

As is clear from the examples in section 4.3, the more complex the manipulation, and the more elements in a structure, the more noise there will be in the results of a manipulation. Consequently, to demonstrate that the SPA is feasible in its details, it is crucial to show that a scalable, effective clean-up memory can be constructed out of biologically plausible neurons. Two researchers in my lab, Terry Stewart and Charlie Tang, have worked out the details of such a memory (Stewart et al., 2010b). Here I describe some of the highlights of that work. I begin by describing the clean-up memory itself and then turn to issues of scaling.

The particular implementation I discuss here is relevant for non-SPA architectures as well. This is because mapping a noisy or partial vector representation to an allowable representation plays a central role in several other neurally inspired cognitive models (Sun, 2006; Anderson and Lebiere, 1998; Pollack, 1988). Indeed, there have been several suggestions as to how such mappings can be done, including Hopfield networks, multilayer perceptrons, or any other prototype-based classifier: in short, any type of auto-associator can fulfill this role.

In the SPA, an ideal clean-up memory would take a given structured semantic

pointer \mathbf{A} and be able to correctly map any vector extracted from it to a valid lexical item. For present purposes, we can suppose that \mathbf{A} is the sum of some number of “role-filler” pairs. For example, there are three such pairs in the previously discussed encoding of “The dog chases the boy.”

Algorithmically, decoding any such structure would consist in:

1. Convolving input \mathbf{A} with a probe vector \mathbf{p} ;
2. Measuring the similarity of the result with all allowable items; and
3. Picking the item with the highest similarity and returning that as the cleaned up result.

Perhaps the most difficult computation in this algorithm is measuring the similarity with all allowable items. This is because the many-to-one mapping between noisy input and clean output can be quite complicated.

So, unfortunately, many simple auto-associators, including linear associators and multilayer perceptrons, do not perform well (Stewart et al., 2010b). These are simple because they are feedforward: the noisy vector is on the input, and after one pass through the network, the cleaned up version is on the output. Better associators are often more complicated. The Hopfield network, for instance, is a dynamical system that constantly feeds its output back to the input. Over time, the system settles to an output vector that can be considered the cleaned up version of an input vector. Unfortunately, such recurrent networks often require several iterations before the results are available, slowing down overall system performance. Ideally, we would like a clean-up memory that is both fast, and accurate.

To build such a memory, we can exploit two of the intrinsic properties of neurons described in section 2.1, and demonstrated in the tutorial on neural representation (section 2.5). The first is that the current in a neuron is the dot-product of an input vector with the neuron’s preferred direction in the vector space. The dot product is a standard way of measuring the similarity of two vectors. So, a similarity measure is a natural neural computation. The second property is that neurons have a non-linear response, so they do not respond to currents below some threshold. This means they can be used to compute nonlinear functions of their input. Combining a similarity measure and a nonlinear computation turns out to be a good way to build a clean-up memory.

Specifically, for each item in the clean-up memory, we can set a small number of neurons to have a preferred direction vector that is the same as that item. Then, if that item is presented to the memory, those neurons will be highly active. And,

for inputs near that item, these neurons will also be somewhat active. How near items must be to activate the neurons will depend on the firing thresholds of the neurons. Setting these to be slightly positive in the direction of the preferred direction vector will make the neuron insensitive to inputs that are only slightly similar. In effect, the inherent properties of the neurons are being used to clean up the input.

Figure 4.7 shows how this neural clean-up memory scales. The parameters affecting how good the clean-up is are: k , the number of role-filler items in the input vector; M , the number of valid lexical items to compare to; and D , the dimensionality of the vector space (i.e., the dimensionality of the semantic pointer). In the simulations shown here, performance for values up to $k = 8$ are shown because George Miller’s (1956) classic standard limits of seven plus or minus two on working memory fall in about the same range. The graphs show M up to a value of 100,000 as this is well over the size of an adult lexicon (Crystal, 2003, suggests 60,000 terms). The plot tracks the dimensionality of the vectors that allow for 99% clean-up accuracy over these ranges of k and M .

Notably, this clean-up memory is also very fast. The network is purely feed-forward, and so clean-up occurs on the time-scale of the neurotransmitter used in the network. For most excitatory connections in cortex, this is on the order of about 5ms (i.e., for AMPA receptors).

So, what conclusions can we draw about the SPA with such a model? While it would be difficult to argue that we are in a position to definitively set a “maximum” number of dimensions for semantic pointers in the SPA, these simulations provide good evidence that the architecture is scalable to within the right neighborhood. Because this model is constructed with 10 neurons per lexical item, we need approximately 600,000 neurons to implement the clean-up memory model described here for an adult-sized lexicon. This works out to about 3.5mm² of cortex, which fits well within the connectivity patterns discussed in section 4.2.2. In addition, the connectivity matrix needed to implement this memory is of the same dimensionality as in the binding network, and so it respects the anatomical constraints discussed there as well.

When coupled with the previous discussion regarding the neural plausibility of the binding network (section 4.2.2), there is a strong case to be made that with about 500 dimensions, structure representation and processing of human-like complexity can be accomplished by a SPA. This is, we have good reason to suppose we have appropriate tools for constructing a model able to perform sophisticated structure processing in a biologically plausible manner. We will see an example of such a model in the next section.

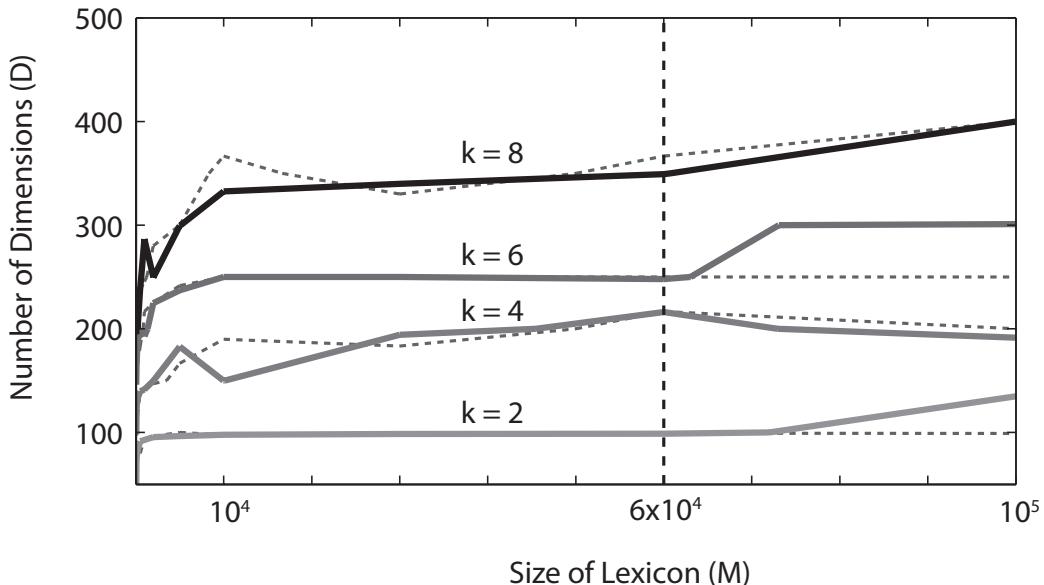


Figure 4.7: Scaling properties of a neural clean-up memory. An input vector is formed by binding k pairs of semantic pointers together and then adding the bound pairs together. The clean-up memory must recover one of the individual semantic pointers that formed the input vector by convolving the input with a probe vector. This figure plots the minimum number of dimensions required to recover a semantic pointer from the input vector 99% of the time. Data was collected using average results from 200 simulations for each combination of k , M , and D values. The vertical dashed line indicates the approximate size of an adult lexicon (Crystal, 2003). The horizontal dashed lines show the performance of a non-neural clean-up directly implements the three-step algorithm. The neural model performs well compared to this purely computational implementation of clean-up.

Before doing so, however, it is worth considering a few issues related to clean-up memory in the SPA. In general, the SPA is not very specific with respect to the anatomical location or number of clean-up memories we should expect to find in the brain (see section 10.2). The calculations above suggest they may be ubiquitous throughout cortex, but perhaps many structural manipulations can be performed before clean-up is needed, which would be more efficient. As well, construction of clean-up memories may be a slow process, or a rapid one, or, I suspect most likely, both. That is, it makes sense to have long-term clean-up memories that account for semantic memory, as well as clean-up memories constructed on-the-fly that are able to track the current inferential context. It is well possible, for instance, that areas of the brain known for rapid learning, such as the hippocampus, may play a central role in helping us to navigate through complex structure processing. The considerations in chapter 6 on memory and learning speak to general problems related to learning, so I will not consider these further here.

However, I would like to address a concern that has been expressed about any attempt to choose a fixed number of dimensions for semantic pointers. The concern is that we have no good reasons to expect that all representations will be of the same dimension. Perhaps some simple representations may not be of sufficiently high dimension, or that perhaps perceptual and motor systems work with vectors that are of very different dimensions. Or, perhaps, if we have a “maximum” number of dimensions, the system simply stop working if we go over that number.

There are two kinds of response to these concerns, theoretical and practical. On the theoretical side, we can note that any vectors with lower than the maximum number of dimensions can be perfectly represented in the higher-dimensional space. In mathematical terms, note that vector spaces with smaller numbers of dimensions are subspaces of vector spaces with more dimensions. Hence, there is nothing problematic about embedding a lower-dimensional space in a higher-dimensional one.

In the opposite case, if there are more dimensions in a representation than in the vector space, we can simply choose a random mapping from the representation to the vector space (even truncation often works well). There are more sophisticated ways of “fitting” higher-dimensional spaces into lower dimensional ones (such as performing singular-value decomposition), but it has been shown that for high-dimensional spaces, simple random mappings into a lower-dimensional space preserve the structure of the higher-dimensional space very well (Bingham and Mannila, 2001). In short, all or most of the structure of “other dimensional”

spaces can be preserved in a high-dimensional space like we have chosen for the SPA. Vector spaces are theoretically quite robust to changes in dimension.

On the more practical side, we can run explicit simulations in which we damage the simulation (thereby not representing the vector space well, or losing dimensions), or add significant amounts of noise, which distorts the vector space and can effectively eliminate smaller dimensions (as measured by the singular values). Performing these manipulations on the binding network described earlier (specifically, the third example in figure 4.5) demonstrates significant robustness to changes in the vector space. For instance, after randomly removing neurons from the binding population in the network, accurate performance occurs even with an average of 1,221 out of 4000 neurons removed (see figure 6.20).

As well, the discussion surrounding figure 6.20 shows that the network is accurate when up to 42% Gaussian noise is used to randomly vary the connection weights. Random variation of the weights can be thought to reflect a combination of imprecision in weight maintenance in the synapse, as well as random jitter in incoming spikes. Consequently, such simulations speak both to concerns about choosing a specific number of dimensions, and to concerns about the robustness of the system to expected neural variability.

Overall, then, the components of the SPA allow for graceful degradation of performance. This should not be too surprising. After all, previous considerations of neural representation make it clear that there is a smooth link between the number of neurons and the quality of the representation (see section 2.5). As well, we have just seen that there is a smooth relationship between the quality of a clean-up memory, the number of dimensions, and the complexity of the representation for a fixed number of neurons. Not surprisingly, there are similarly smooth relationships between dimensionality and complexity in ideal VSAs (Plate, 2003). Since the SPA combines VSAs with neural representation, it is also not surprising that it inherits the kinds of graceful degradation characteristic of these approaches.

4.6 Example: Fluid intelligence

In cognitive psychology, “fluid intelligence” is distinguished from “crystallized intelligence” to mark the difference between cognitive behavior that depends on currently available information that must be manipulated in some sophisticated manner, and cognitive behavior that depends on potentially distant past experiences. Fluid intelligence is exemplified by solving difficult planning problems, whereas crystallized intelligence is exemplified by recalling important facts about

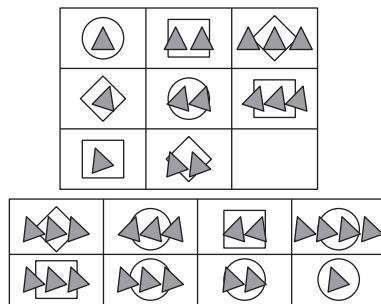


Figure 4.8: Example Raven's Progressive Matrix (RPM). A subject must examine the first two rows (or columns) of the matrix to determine a pattern. The task is pick one of the eight possible answers shown along the bottom to complete that pattern for the third row (or column). Matrices used in the RPM cannot be published, so this is an illustrative example only.

the world. Most tests that have been designed to measure general intelligence, focus on fluid intelligence. One of the most widely used and respected tools for this purpose is the Raven's Progressive Matrices (RPM; Raven, 1962).

In the RPM subjects are presented with a 3×3 matrix where each cell in the matrix contains various geometrical figures, with the exception of the final cell, which is blank (figure 4.8). The task is to determine which of eight possible answers most appropriately belongs in the blank cell. In order to solve this task, subjects must examine the contents of all of the other cells in the matrix to determine what kind of pattern is there, and then use that pattern to pick the best answer. This, of course, is an excellent example of induction, which I mentioned earlier in section 4.4.

While the RPM is an extremely widely used clinical test, and many experiments have been run using the test, our understanding of the processes underlying human performance on this task is minimal. To the best of my knowledge, there have been no cognitive models of the RPM that include the inductive process of rule generation. The best-known model of RPM includes all of the possible rules that the system may need to solve the task (Carpenter et al., 1990). In solving the task, the model chooses from the available rules, and applies them to the given matrix. However, this treats the RPM like a problem that employs crystallized intelligence, which contradicts its acceptance as a test of fluid intelligence (Prabhakaran et al., 1997; Gray et al., 2003; Perfetti et al., 2009).

Recently, however, a graduate student in my lab named Daniel Rasmussen

proposed a model of the RPM that is implemented in a spiking neural network, using the methods described here (Rasmussen and Eliasmith, 2011). To understand how his model works, we can consider the example Raven’s-style matrix shown in figure 4.8. To correctly solve this problem, a subject needs to extract information from the filled-in cells that identifies three rules: 1) the number of triangles increases by one across the row; 2) the orientation of the triangles stays the same across the row; and 3) each cell also contains a shape in the background, which, across the row includes each of a circle, a square, and a diamond. This combination of rules identifies the correct response (i.e., three triangles tilted to the left over top of a circle), which can be determined by applying those rules to the last row. While not all subjects will explicitly formulate these rules, they must somehow extract equivalent information (or get lucky) to get this matrix correct.

In Rasmussen’s graduate work, he argues that there are three kinds of rules needed to solve the entire set of problems on the RPM (Rasmussen, 2010). The first kind are induction rules, which generalize across neighboring cells of the network (e.g., increase the number of triangles by one). The second kind are set completion rules, which account for set completion across an entire row (e.g., each of a circle, a square, and a diamond are represented within a row). The third kind are visually based XOR rules, which account for the presence and absence of particular visual features across a row. In addition, the choice and coordination of the application of these rules is accounted for by an executive system. In recent work he has shown that the full model gets 67% correct on average, whereas college undergraduates get about 61% correct on average. However, for simplicity, here I will only focus on the induction rules (for discussion of other these aspects of the model, and full implementational details see Rasmussen (2010)).

In section 4.4, I described how we can perform induction to infer the transformation of abstract structure. To make this characterization more concrete, consider the simplified Raven’s-style matrix shown in figure 4.9. Each cell in this matrix is represented by a semantic pointer which consists of role-value pairs. For example, the top right cell could be represented as:⁴

$$\text{cell} = \text{shape} \circledast \text{circle} + \text{number} \circledast \text{three}.$$

Once all of the cells are represented, the previously described induction rule is

⁴Surprisingly few decisions need to be made about which role-filler pairs to include in the representations of cells. As long as the information necessary to identify the required transformation is available, the induction rule is generally able to extract it. Thus, including additional information like **color** \circledast **black** + **orientation** \circledast **horizontal** + **shading** \circledast **solid**, etc. in this instance does not affect performance.

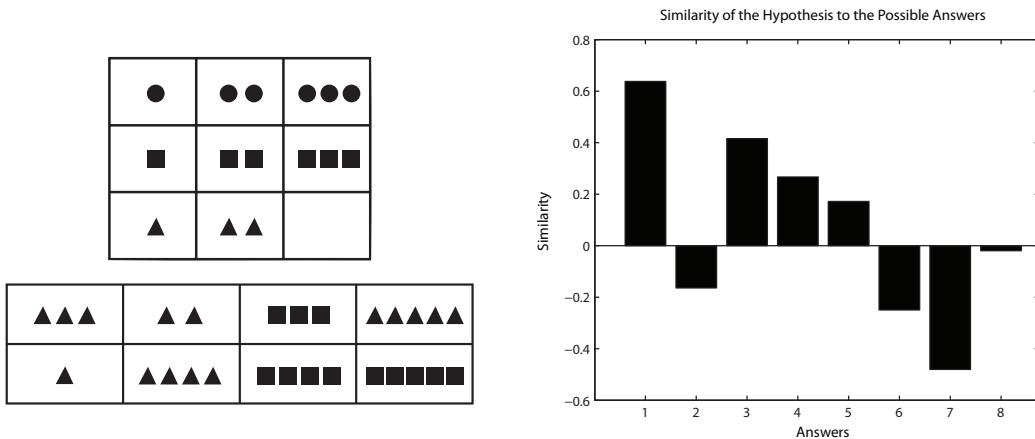


Figure 4.9: A simple induction RPM example. On the left is a simple RPM matrix in which the number of items increases by one across each row. The model is presented with consecutive pairs of cells, and infers a transformation. That transformation is applied to the last cell and the similarity to each answer is computed. This similarity is shown on the right, indicating that the model correctly identifies the first answer (i.e., three triangles) as the correct one.

provided the pairwise examples in the order they tend to be read by subjects (i.e., across rows, and down columns; Carpenter et al. (1990)). As the examples are presented, the inferred transformation is built up out of the average of each of the pairwise transformations. That inferred transformation is then be applied to the second last cell in the matrix. This results in a vector which is compared to each of the possible answers, and the most similar answer is chosen as the correct one by the model.

Comparing the model across many runs to average human performance on all the matrices that include an induction rule (13 out of 36 examples), humans score about 76% on average (Forbes, 1964), and the model scores 71% on average (chance is 13%). So the model is performing induction at a similar level to human subjects.

More generally, the model can account for several other interesting experimental observations on the RPM. For instance, subjects improve with practice if given the RPM multiple times (Bors, 2003), and also show learning within the span of a single test (Verguts and De Boeck, 2002). As well, a given subject's performance is not deterministic; given the same test multiple times, subjects will get previously correct answers wrong and vice versa (Bors, 2003). In addition, there

are both qualitative and quantitative differences in individual ability; there is variability in “processing power” (variously attributed to working memory, attention, learning ability, or executive functions), but there are also consistent differences in high-level problem-solving strategies between low-scoring and high-scoring individuals (Vigneau et al., 2006). This is not an exhaustive list, but it represents some of the features that best characterize human performance, and all of these features are captured by the full RPM model (Rasmussen, 2010).

That being said, the theoretically most important feature of the model for my purposes is its ability to successfully perform induction and extract rules. However, simply choosing the best of the eight possible answers might not convince us that it has in fact found a structural regularity. Much more convincing, is determining how the learned transformation applies in circumstances never encountered by the model during training. Figure 4.10 shows two examples of this kind of application. In the first instance, the model is asked to apply the learned transform to four triangles. As is evident from figure 4.10a, the model’s preferred answer corresponds to five triangles. This suggests the inferred transformation encodes the correct “increase by one” aspect of the rule, at least in the context of triangles. Figure 4.10b shows that, in fact, the “increase by one” is generic across objects. In this case, the rule is applied to four squares, and the preferred result is five squares. In short, the model has learned the appropriate counting rule, and generalized across objects.

This is a simple, but clear, example of what is sometimes called “syntactic generalization”. Syntactic generalization is content insensitive generalization; i.e., driven by the syntactic structure of the examples. It is important to note that precisely this kind of generalization has been often been claimed to distinguish cognitive from non-cognitive systems (Fodor and Pylyshyn, 1988a; Jackendoff, 2002; Hummel and Holyoak, 2003). In this case, it is clear that the generalization is content insensitive because “increase by one” is applied for squares, triangles, or what have you. I provide more examples of syntactic generalization in section 6.6.

It might not be too surprising that in order to perform cognitively interesting induction, one has to perform syntactic generalization. What is more interesting, is that once we have an appropriate representational system (semantic pointers and binding), the method for performing such syntactic induction is quite simple (i.e. averaging over examples), and can be built into a biologically realistic network. In addition, the same basic architecture is able to capture the other two kinds of reasoning usually identified in cognitive science: abduction and deduction. In previous work, I showed that the approach underlying the SPA can be used to capture

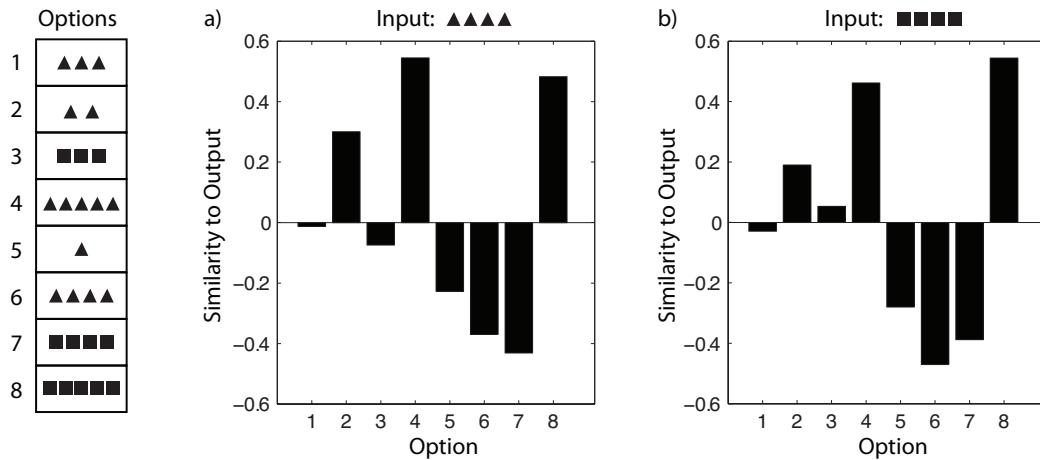


Figure 4.10: Broader application of the rule induced from figure 4.9. a) Applied to four triangles, the model chooses five triangles as the next item in the sequence. b) Applied to four squares the model chooses five squares as the next item in the sequence.

central features of human deductive reasoning (Eliasmith, 2004); I describe this work in more detail in section 6.6. Thagard and Litt (2008) also showed that a version of this model can be used to capture abductive reasoning (i.e., inference to the best explanation).

This unification of kinds of reasoning in a simple neural mechanism is appealing because it brings together a range of what might otherwise seem very different cognitive strategies. Furthermore, having a consistent architecture in which to explore these mechanisms in a biologically constrained manner provides a useful way to begin constructing models of biologically based reasoning. As I noted earlier, central issues regarding the control, contextual effects, competing strategies, etc. are not accounted for by this mechanism. Nevertheless, such a mechanism provides fertile ground for building more sophisticated models, especially in the broader context of the SPA.

I take it that it is no accident that past models of the RPM have not had mechanisms for inferring rules. The mechanisms are not there because there is no obvious way to infer the appropriate cognitive rules given the example rules in a symbolic representation. There is little sense to be made of “averaging” a series of symbolic rules. And, there is no simple replacement for “averaging” with another operation. With the SPA, however, averaging past examples of rules becomes both

very natural, and very powerful. It is powerful because the SPA representations allow whatever changes across examples to become “noise”, and whatever stays consistent to be reinforced. Crucially, this is true for both content and syntactic structure.

One final note, which strikingly distinguishes this SPA implementation from standard connectionist approaches, is that there are no connection weight changes in this model. This is true despite the fact that it learns based on past experience, and is able to successfully generalize. I return to this observation in my discussion on learning in chapter 6.

4.7 Deep semantics for cognition

In the last chapter, I distinguished deep and shallow semantics for both perception and action, but I did not address the issue of deep semantics for more cognitive representations. This is because I could not address sophisticated, structured representations without discussing syntax. As I commented in the previous chapter, I did not describe a means of integrating the variety of possible sources of semantics, which no doubt include the different modalities, motor systems, lexical relationships, and so on. Now, however, we have seen a method for building structures out of sets of semantic pointers.

Consequently, it is a natural extension of the previous discussion on semantics to suggest that the complex kind of semantic structures found in human cognition can be constructed by combining semantic pointers from different semantic sources. So, for example, the perceptual features of a “robin” might be represented as something like:

$$\text{robinPercept} = \text{visual} \circledast \text{robinVis} + \text{auditory} \circledast \text{robinAud} + \text{tactile} \circledast \text{robinTouch} + \dots$$

The pointers **robinVis**, **robinAud**, etc. could be those at the top of the relevant hierarchy (like the semantic pointers used for visual number classification in the previous chapter). We could then take those perceptual features to be integrated into a more lexical representation of the concept “robin” that might look something like:

$$\text{robin} = \text{perceptual} \circledast \text{robinPercept} + \text{isA} \circledast \text{bird} + \text{indicatorOf} \circledast \text{spring} + \dots$$

Importantly, the resulting representations **robin** and **robinPercept** are also semantic pointers. After all, circular convolution is a compression operation, so

the results of binding two vectors is a compressed representation of the original content (as for any semantic pointer). In this case, what is being represented is not perceptual features, but structure. And, just as with perceptual semantic pointers, structured semantic pointers must also be dereferenced by effectively “running the model backwards”. In the case of structure, “running the model backwards” means unbinding, and cleaning up the results of that operation. In the case of perceptual semantic pointers, this meant decompressing the representation by seeding the model that had been constructed based on perceptual experience. The only difference, it may seem, is that the perceptual model is hierarchical and clean-up memory is not. But this, we will see shortly, may not be a difference after all.

In any case, we now have, theoretically at least, a means of bringing together shallow and deep semantics by binding them into a single semantic pointer. But, we can go further. One additional, interesting feature of the preceding characterization of the SPA is that transformations of semantic pointers are themselves semantic pointers. For example, when the model solving the Raven’s matrices induced a transformation to explain the changes in the structures it had been presented with, the result was a transformation vector in the same vector space as the structures it transformed. In some ways, it should not be too surprising that transformations can be encoded as semantic pointers given my earlier consideration of the motor control system. After all, in that context semantic pointers were used to drive motor control – a clear case of transformations.

Consequently, since semantic pointers can act as transformations, and conceptual structure can be built up out of bound semantic pointers, our conceptual representations can include characterizations of transformations in them. That is, not only static properties, but also the kinds of changes associated with objects can be encoded in (or pointed to by) the lexical representation of that object. More precisely, a compressed description of an object’s possible transformations can be encoded in the lexical representation of that object. As with any “part” of such a representation, it would need to be dereferenced, likely by another part of the brain, to decode its semantics. Nevertheless, I believe this is an intriguing and natural way to include an element of time in our understanding of lexical conceptual representations, an element which is often missing in discussions of concepts.

To sum up, I have suggested that SPA conceptual structures can be very complex, including the sum of bound representations of properties, perceptual features, lexical relationships, dynamics, and so on. But, there seems to be an important problem lurking here. Namely, that in my previous description of a clean up memory, I noted that we could accurately decode about 8 role-filler pairs in

a network that was anatomically plausible. Certainly, more than 8 roles will be needed in order to include all of these kinds of information in a single semantic pointer. How then are we going to be able to encode sophisticated concepts in a biologically realistic manner using the resources of the SPA?

I would like to propose a solution to this problem, that turns out to allow the architecture to scale well even for complex structures. Namely, rather than constraining ourselves to a single clean-up memory for decoding, we can chain memories together. In short, I am suggesting that we can perform successive decodings in different clean-up memories. This would result in a chain of decoding, where we perform initial decoding in one memory, take the results of that decoding, and further decode those results a second memory. This process could then be repeated several times. At each stage of such a chain, the vector to be decoded comes from the clean-up of the previous stage, ensuring that full capacity of the current stage is available for further decoding of the semantic pointer. As a result, the effective capacity will scale as the power of the number of stages in such a chain, but each stage will only require a linear increase in the number of neurons. Let us consider a concrete example.

Consider a simple example of how the “dog” semantic pointer might be constructed. It might include role-filler pairs for a wide variety of information, as described earlier. Suppose for this simple example that there are only three roles. The representation of this pointer might then look something like:

$$\mathbf{dog} = \mathbf{isA} \circledast \mathbf{mammal} + \mathbf{hasProperty} \circledast \mathbf{friendly} + \mathbf{likes} \circledast \mathbf{bones}$$

Suppose that with a 100-dimensional semantic space, we can decode any elements in a three role representation with 99.9% accuracy. If we query our “dog” concept, wanting to know what it is (i.e., determining the filler in the “isA” role) we will discover that a dog is a mammal. That is, the result of this operation will be a clean version of the “mammal” semantic pointer.

We might then decode this new semantic pointer with a second memory in our chain. Suppose that this semantic pointer is the sum of three other role-filler pairs, perhaps

$$\mathbf{mammal} = \mathbf{isA} \circledast \mathbf{animal} + \mathbf{hasProperty} \circledast \mathbf{hair} + \mathbf{produces} \circledast \mathbf{Milk}$$

We could then decode any of the values in those roles also with 99.9% accuracy. That is, we could determine that a mammal is an animal, for instance. And so the chaining might continue with the results of that decoding. What is interesting to note here is that in the second memory we could have decoded any three

role-filler pairs from any of the three role-filler pairs in the memory earlier in the chain. So, effectively, our original concept can include nine role-filler pairs that we would be able to decode with near 100% accuracy (the expected accuracy is $99.9\% * 99.9\% = 99.8\%$).

In short, we can increase the effective number of role-filler pairs by chaining the spaces. As figure 4.11 shows, the scaling of a chained space significantly outperforms an un-chained space. It is clear from this figure that even with the same total number of dimensions, an un-chained space can not effectively decode very many pairs (the number of pairs is approximately linear in the number of dimensions). In contrast, the number of decodable pairs in the chained space is equal to the product of the number of decodable pairs in each space separately. In this example it means we can effectively decode 150 instead of 20 role-filler pairs with the same number of dimensions.

We can project the expected accuracy for much larger sets of pairs by turning to our earlier consideration of clean-up memory. For instance, we know that we can clean-up in a 500 dimensional space about 8 pairs from a vocabulary of 60,000 terms with 99% accuracy using the number of neurons consistent with observed connectivity (see figure 4.7). If we chain 4 such memories together, we would be able to decode $8^4 = 4096$ pairs with $.99^4 = .96$, or 96%, accuracy. This kind of scaling makes earlier concerns about encoding sophisticated concepts much less pressing.

Why do we get such good scaling? In short, it is because we have introduced a nonlinearity (the clean-up) into our decoding process that is not available in a single memory. That nonlinearity acts to align the output of one memory with the known values in the next semantic space in the chain. It may seem that this enforces a kind of hierarchy on our conceptual structure, but it does not. The reason I use the word “chain” instead of “hierarchy” is because there are many kinds of chains that could be implemented in this way, only some of which are hierarchies. For instance, the second memory in a two-level chain could be the same as the first (a recursive chain). Or, each possible role could have its own dedicated memory for decoding items of that role-type (a strict hierarchy). It is unclear which of these, or other, kinds of structure will be most appropriate for characterizing human-like semantic spaces. It is intriguing, however, that a hierarchy is a very natural kind of chain to implement, and that there is some evidence for this kind of structure to human concepts (Collins and Quillian, 1969). As well, parsimony makes it tempting to assume that because of the hierarchical nature of perceptual and motor systems (also used for capturing deep semantics), cognition will have a similar hierarchical structure. However, mapping any particular structure

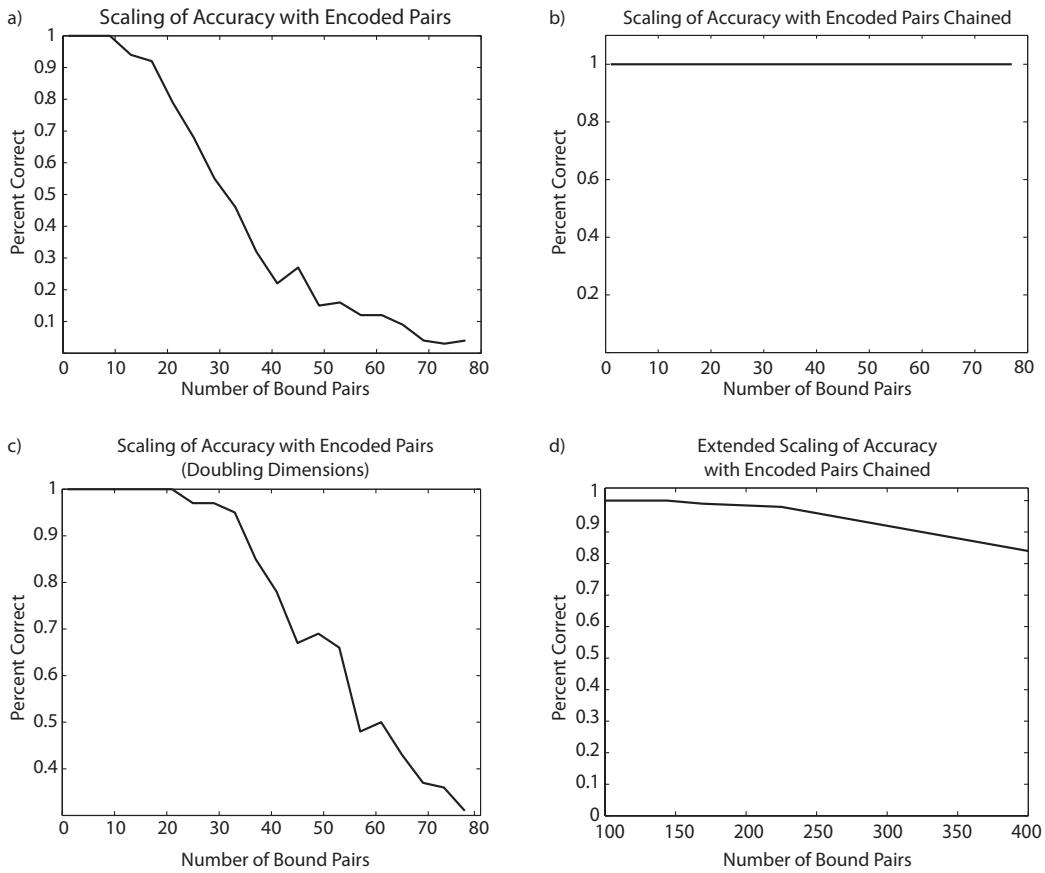


Figure 4.11: Factorization effects for encoding large conceptual structures. a) The accuracy versus number of encoded role-filler pairs rapidly decreases in a single memory. This is a 500-dimensional space with 60,000 possible terms. The accuracy falls to 3% for 80 pairs. b) The accuracy of a pair of memories that are chained. Each space is 500-dimensional and has 60,000 possible terms. The decoding accuracy is 100% for up to 80 pairs. c) The accuracy of a single memory with the same total number of dimensions as the chained space. This is a 1000-dimensional space with 60,000 possible terms. The decoding accuracy is 31% for 80 pairs. d) The extended scaling of the chained space between 100 and 400 pairs. The scaling begins to decline smoothly around 150 pairs, but is still 84% accurate for 400 pairs. All points are averages of 100 runs.

carefully to data on human concepts is beyond the scope of this book (see, for example, Rogers and McClelland (2004) for interesting work along these lines).

Regardless of how a specific mapping turns out, in general I believe chained semantic spaces provide a crucial demonstration of how syntax and semantics can be used together to represent very large, intricate conceptual spaces in ways that can be plausibly be implemented in the brain. These semantic spaces are tied directly to perceptual and motor experiences as described in the previous chapter, and can also be used to encode temporal transformations, language-like structures, and lexical relationships. Furthermore, we can transform these representations in structure-sensitive ways, and even learn to extract such transformations from past examples. The stage is now set for considering how to build a system which itself uses, transforms, and generates semantic pointer representations.

4.8 Nengo: Structured representations in neurons

This chapter introduced a method for constructing structured representations using semantic pointers. Semantic pointers themselves are high-dimensional vector representations. When we think of semantic pointers as acting like symbols, we can name them to keep track of different semantic pointers in the same high-dimensional space. The collection of named vectors in a space forms a kind of ‘vocabulary’ of known semantic pointers in that space. Structured representations can then be built out of this vocabulary. In this tutorial, I discuss how to run simulations in Nengo that create and use semantic pointer vocabularies.

- Open a blank Nengo workspace and create a network named ‘Structured Representation’.
- Create a ‘default’ ensemble in the network, name it ‘A’, give it 300 neurons, and make it 20 dimensions.

This ensemble is representing a vector space just as in the simpler low-dimensional (i.e., 2D and 3D) cases we considered earlier. To use semantic pointers, we need to work in higher-dimensional spaces.

- Open the network with *Interactive Plots*. Right-click the ‘A’ population and click ‘value’. Right-click the population again and click ‘semantic pointer’. You should now have two graphs.
- Run the simulation for about 1 second.

Displaying the ‘value’ graph in *Interactive Plots* shows the value of individual components of this vector.⁵ The ‘semantic pointer’ graph compares the vector represented by the ensemble to all of the elements of the vocabulary, and displays their similarity. Initially, the vocabulary contains no vectors and thus nothing is plotted in the semantic pointer graph. As well, since there is no input to the population, the value graph shows noise centered around zero for all components.

- Right-click the semantic pointer graph and select ‘set value’. Enter ‘a’ into the dialog that appears and press *OK*.
- Run the simulation again.

Using the ‘set value’ option of the semantic pointer graph does two things: 1) if there is a named vector in the set value dialog that is not part of the vocabulary, it adds a randomly chosen vector to the vocabulary of the network and associates the given name (e.g., ‘a’) with it; and 2) it makes the represented value of the ensemble match the named vector. The result is that the ensemble essentially acts a constant function that outputs the named vocabulary vector. As a result, when the simulation is run, the semantic pointer graph plots a 1, because the representation of the ensemble is exactly similar (i.e., equal) to the ‘a’ vector.

- Right-click the ‘semantic pointer’ graph and select ‘set value’. Enter ‘b’ into the ‘Set semantic pointer’ dialog and press *OK*.
- Run the simulation.
- Switch between setting ‘a’ and setting ‘b’ on the semantic pointer graph while the simulation is running.

Setting the value of the semantic pointer to ‘b’ adds a second, random 20-dimensional vector to the vocabulary. The ‘value’ plot reflects this by showing that the neural ensemble is driven to a new vector. Switching between the ‘a’ and ‘b’ vectors, it should be clear that although the vectors were randomly chosen initially, each vector is fixed once it enters the vocabulary. The semantic pointer graph changes to reflect which vocabulary item is most similar to the current representation in the ensemble.

⁵By default, the plot shows five components – to change this you can right-click the ‘value’ graph and choose ‘select all’ but be warned that plotting all twenty components may cause the simulation to be slow. Individual components can be toggled from the display by checking or unchecking the items labeled ‘v[0]’ to ‘v[19]’ in the right-click menu.

These are the vocabulary items that can be combined to form structured representations. To experiment with the binding (i.e., convolution) and conjoin (i.e., sum) operations, we require additional ensembles.

- In the ‘Structured Representation’ network, add a ‘default’ ensemble named ‘B’ with 300 neurons and 20 dimensions.
- Add a third ensemble named ‘Sum’ with the same parameters.
- Add a decoded termination to the ‘Sum’ ensemble. Name the termination ‘A’ and set the number of input dimensions to 20, and ‘tauPSC’ to 0.02.
- The coupling matrix of the termination should be set to an identity matrix by default (a grid of zeros except for a series of ones along the diagonal). Click *Set Weights* to make sure, and keep this default. Click *OK* twice.
- Add a second decoded termination to the ‘Sum’ ensemble. Name the termination ‘B’ and set all other parameters as for the ‘A’ termination.
- Add projections from ensembles ‘A’ and ‘B’ to the ‘Sum’ ensemble.
- Add a fourth ensemble to the network named ‘C’ with the same parameters as all of the previous ensembles.
- Drag the icon for the ‘Binding’ template from the bar on the left side of the screen into the network viewer for the ‘Structured Representation’ network.
- In the dialog box that is opened, enter the name ‘Bind’, use ‘C’ as the name of the output ensemble, and set 60 neurons per dimension. The two inversion options should not be checked.
- Project the ‘A’ and ‘B’ ensembles to the ‘Bind’ network that was created.

The ‘Sum’ ensemble should be familiar from the previous tutorial about transformations (section 3.8). However, the ‘Bind’ network uses a Nengo template to construct a subnetwork, which we have not done before. Nengo can create subnetworks by placing one network inside another and exposing selected terminations and origins of the inner network to the outer network. This is a useful technique for organizing a model, and it is used here to group the ensembles required to compute a binding operation within a single subnetwork element.

In general, the template library allows common network components to be created quickly and with adjustable parameters. The templates call script files that

advanced Nengo users can write to aid rapid prototyping of networks (scripting is the topic of the section 7.5 tutorial).

The ‘Bind’ network is a component that computes the circular convolution of its two inputs using nonlinear transformations as discussed in the preceding tutorial. To look at the subnetwork elements you can double-click it. You cannot see the connections into and out of the subnetwork when you open it. We can now continue to build a binding network.

- Open the *Interactive Plots* viewer. Right-click the background and select the ‘A’, ‘B’, ‘C’, ‘Bind’ and ‘Sum’ nodes if they aren’t already displayed.
- Show the semantic pointer graphs for the ‘A’, ‘B’, ‘C’, and ‘Sum’ nodes by right-clicking the nodes and selecting ‘semantic pointer’.
- Set the value of the ‘A’ ensemble to ‘a’ and the value of the ‘B’ ensemble to ‘b’, by right-clicking the relevant semantic pointer graphs.
- Right-click the semantic pointer graph of ensemble ‘C’ and select ‘show pairs’.
- Right-click the semantic pointer graph of ensemble ‘C’ and select ‘a*b’.
- Repeat the previous two steps for the ‘Sum’ ensemble, so that both ‘show pairs’ and ‘a*b’ are checked.
- Run the simulation for about 1 second, and then pause it.

The ‘C’ population represents the output of a neurally-computed circular convolution (i.e., binding) of the ‘A’ and ‘B’ input vectors, while the ‘Sum’ population represents the sum of the same two input vectors. The label above each semantic pointer graph displays the name of the vocabulary vectors that are most similar to the vector represented by that neural ensemble. The number preceding the vector name is the value of the normalized dot product between the two vectors (i.e., the similarity of the vectors).

In this simulation, the ‘most similar’ vocabulary vector for the ‘C’ ensemble is ‘a*b’. The ‘a*b’ vector is the analytically-calculated circular convolution of the ‘a’ and ‘b’ vocabulary vectors. This result is expected, of course. Also of note is that the similarity of the ‘a’ and ‘b’ vectors alone is significantly lower. Both of the original input vectors should have a low degree of similarity to the result of the binding operation. Toggling the ‘show pairs’ option of the graph makes the

difference even clearer. The ‘show pairs’ option controls whether bound pairs of vocabulary vectors are included in the graph.

In contrast, the ‘a’ and ‘b’ vectors have a relatively high (and roughly equal) similarity to the vector stored in the ‘Sum’ ensemble. The sum operation preserves features of both original vectors, so the sum is similar to both. Clearly, the conjoin and bind operations have quite different properties.

We have now seen how we can implement the two functions needed to create structured representations in a spiking network. However, to process this information, we must also be able to extract the information stored in these conjoined and bound representations. This is possible through use of an inverse operation.

- Set the value of the ‘A’ semantic pointer to ‘ $a*c+b*d$ ’.
- Set the value of the ‘B’ semantic pointer to ‘ $\sim d$ ’.
- Hide the ‘Sum’ and ‘C’ semantic pointer graphs.
- Display the ‘C’ semantic pointer graph again by right-clicking ‘C’ and selecting ‘semantic pointer’ (hiding and displaying the graph allows it to adjust to the newly added ‘c’ and ‘d’ vocabulary items).
- Right-click the graph of ‘C’ and select ‘select all’ (the letters ‘a’ through ‘d’ should be checked).
- Run the simulation for about 1 second.

In this simulation, we first set the input to a known, structured semantic pointer. The value ‘ $a*c+b*d$ ’ is the semantic pointer that results from binding ‘a’ with ‘c’ and ‘b’ with ‘d’ then summing the resulting vectors. This vector is calculated analytically as an input, but it could be computed with neurons if need be. The second input, ‘ $\sim d$ ’, represents the pseudo-inverse of ‘d’ (see section 4.2.1). The pseudo-inverse vector is a shifted version of the original vector that approximately reverses the binding operation. Binding the pseudo-inverse of ‘d’ to ‘ $a*c+b*d$ ’ yields a vector similar to ‘b’ because the ‘ $b*d$ ’ operation is inverted while the ‘ $a*c$ ’ component is bound with ‘d’ to form a new vector that is dissimilar to anything in the vocabulary and so is ignored as noise.

- Experiment with different inverse values as inputs to the ‘B’ ensemble (‘ $\sim a$ ’, ‘ $\sim b$ ’, or ‘ $\sim c$ ’). Note the results at the ‘C’ ensemble for different inputs.

- You can also try changing the input ‘statement’ given to the ‘A’ ensemble by binding different pairs of vectors together and summing all the pairs.
- Binding more pairs of vectors together will degrade the performance of the unbinding operation. If ‘B’ is set to ‘~a’, then setting ‘A’ to ‘a*b+c*d+e*f+g*h’ will not produce as clean an estimate of ‘b’ as when ‘A’ is set only to ‘a*b’.

Instead of naming the vectors ‘a’, ‘b’ and so on, you can name them ‘dog’, ‘subject’, and so on. Doing so makes it clear how this processing can be mapped naturally to the various kinds of structure processing considered earlier in this chapter. Note that the name ‘I’ is reserved for the identity vector. The result of binding any vector with the identity vector is the vector itself. The lowercase ‘i’ does not refer to the identity vector.

There are two competing constraints that determine the accuracy of information storage and retrieval in this network. First, the number of dimensions. With a small number of dimensions, there is a danger that randomly chosen vectors will share some similarity due to the small size of the vector space (see figure 3.2 in chapter 3). So it would be desirable to have very high dimensional spaces. However, the second constraint is the number of neurons. A larger number of dimensions requires a larger number of neurons to accurately represent and manipulate vectors. There are, of course, only a limited number of neurons in the brain. So, there is an unsurprising trade-off between structure processing power, and the number of neurons, as I explored in section 4.5. The network presented in this tutorial is using a rather low number of dimensions and neurons for its semantic pointer representations for the sake of computational performance.

Chapter 5

Biological cognition – control

5.1 The flow of information

To this point in the book, I have discussed two concepts central to cognitive science: semantics and syntax. Both notions are primarily concerned with properties of representations that give us insight into the kinds of functional role representations can play in a cognitive architecture. A third, crucial, but often over-looked, aspect of any cognitive architecture is control: the control of which representations to employ; the control of how to manipulate current representations given current context; the control of what next course of action to pursue, and so on. All of these allow representations to play their functional roles, and so all are central to making an architecture cognitive. In short, controlling the flow of information through the system in order to determine how to guide it through the world needs to be understood if we want to understand the flexible, robust, and adaptive nature of biological, cognitive systems.

I suspect that control, despite its intuitive importance, is somewhat under-examined because it does not become absolutely indispensable until we start to construct large and complex models. Consequently, many models – which focus on specific cognitive phenomena, or specific cognitive tasks – do not really face the problem of control. A majority of models in the behavioral sciences are built to explain tasks restricted to a single, often tightly constrained domain, be it reading, navigating, eye control, memory performance, pattern recognition, simple decision making, or what have you. Biological cognitive systems, in contrast, need to be able to perform all of these tasks, switching between them as appropriate. Consequently, any prospective cognitive architecture needs to specify: how infor-

mation can be routed to different areas of the brain to support a given task; how that same information can be interpreted differently depending on the context; how the system can determine what an appropriate strategy is; and so on.

In general, the process of control can be usefully broken down into two parts: 1. determining what an appropriate control signal (or state) is; and 2. applying that control signal to affect the state of the system. The first of these tasks is a kind of decision making. That is, determining what the next course of action should be based on currently available information. The second task is more of an implementational issue: how can we build a system that can flexibly gate information between different parts of the system in useful ways. For instance, if we are speaking on the phone and asked by a friend to report only what we are currently seeing, our brain needs to both decide what action to pursue (e.g., answering the friend's question, which necessitates translating visual to verbal information), and then actually pursue that action (e.g., allowing visual information to drive our language system to generate a report). If the same friend asked for a report on what we were currently smelling, then our brain would configure itself differently, to allow olfactory information to drive the language system. If we could not significantly change the flow of information between our senses and our language system based on slightly different inputs, we would often generate irrelevant responses, or perhaps nonsense. In a sense, this re-routing of information is a kind of cognitive action. The importance of rapidly reconfigurable control is perhaps even more obvious for guiding motor action.

In this chapter, I describe the aspects of the Semantic Pointer Architecture (SPA) most important for control. I begin by presenting the work of a member of my lab, Terry Stewart, on the role of the basal ganglia in action selection, with a focus on cognitive actions. This discussion largely focuses on determining the appropriate control signal. I then describe work pursued by Bruce Bobier, also in my lab, on how actions that demand the routing of information can be implemented in a neurally realistic circuit: this addresses the biologically plausible application of a control signal. Bruce's work focuses on routing information through the visual cortex to explain aspects of visual attention. However, I describe how lessons learned from this characterization of attention can be exploited throughout the SPA. Overall, this discussion forms the foundation of control in an example circuit that combines all aspects of the SPA, which I present in chapter 7.

5.2 The basal ganglia

The basal ganglia are a highly interconnected cluster of brain areas found underneath the neocortex and surrounding the thalamus (see figure 5.1a). For well over twenty years, they have been centrally implicated in selecting one from among several alternative actions available at any given time. This process is unsurprisingly called “action selection”. Damage to the basal ganglia (BG) is known to occur in several diseases of motor control, including Parkinson’s and Huntington’s diseases. Interestingly, these “motor” diseases have also been shown to result in significant cognitive defects (Frank and O’Reilly, 2006). Consequently, both neuroscientists (e.g., Redgrave et al., 1999) and cognitive scientists (e.g., Anderson et al., 2004) have come to understand the basal ganglia as being responsible for “action” selection broadly construed (c.f. Turner and Desmurget, 2010); that is, so as to include both motor and cognitive action (Lieberman, 2006, 2007).

The basal ganglia circuit shown on the left side of figure 5.1 highlights the connectivity accounted for by the classic Albin/Delong model of basal ganglia function (DeLong, 1990; Albin et al., 1989). This model is able to qualitatively account for many of the symptoms of Parkinson’s and Huntington’s diseases. In this model, there is a “direct pathway”, where excitatory inputs from cortex to the D1 cells in the striatum inhibit corresponding areas in GPi and SNr, which then in turn inhibit areas in the thalamus, and an “indirect pathway” from the D2 cells in the striatum to GPe, STN, and then GPi/SNr. However, more recent evidence shows other major connections, including a “hyperdirect” excitatory pathway straight from cortex to STN (Nambu et al., 2002), and other feedback connections, as shown in the schematic on the right side of the figure.

A more recent model of the basal ganglia has suggested that these extra connections can effectively underly action selection (Gurney et al., 2001). This newer model also takes advantage of the great deal of topological structure in the inhibitory connections in basal ganglia. Neurons in the striatum project to a relatively localized area in the GPi, GPe, and SNr, while the excitatory connections from STN are very broad (Mink, 1996). The Gurney et al. model, which forms the basis of our model, makes use of this structure to explain basic action selection.

To see how, consider the striatum, STN and GPi/SNr circuit shown in figure 5.2. This circuit includes the direct and hyperdirect pathways. In this figure, there are three possible actions, which have different “desirabilities” (or “utilities”) indicated by the input weights to the striatum and STN (0.3, 0.8, 0.5). The equally weighted inhibitory connections in the direct pathway cause the most active input to most greatly suppress the action to be selected. However, if not for the addi-

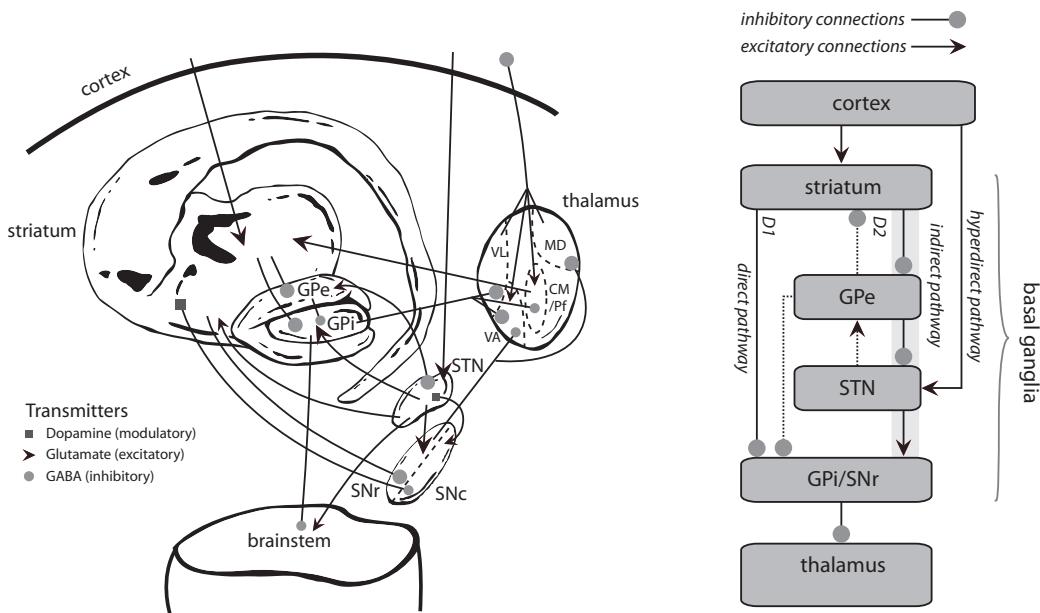


Figure 5.1: (Left) Major connections of the cortico-basal ganglia-thalamo-cortical system. Connections that are particularly massive are shown with larger markers. Abbreviations: CM: centromedian nucleus of the thalamus; Pf: parafascicular nucleus; VA: ventral anterior nucleus; VL: ventral lateral nucleus; MD: medio-dorsal nucleus; SNC: substantia nigra pars compacta; GPe: external globus pallidus; GPi: internal globus pallidus; SNr: substantia nigra pars reticulata; STN: subthalamic nucleus. (Right) A schematic diagram of the basal ganglia showing the standard direct/indirect pathway and the hyperdirect pathway. Other major connections that have been recently discovered are shown with dotted lines.

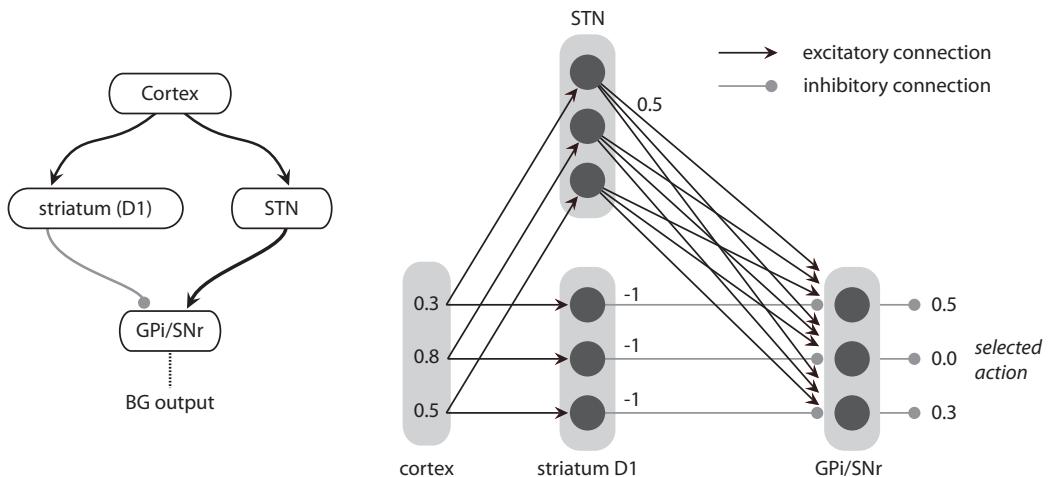


Figure 5.2: Action selection via the striatum D1 cells and the subthalamic nucleus (STN). Connections from the STN are all excitatory and set at a weight of 0.5. The input with the highest utility (0.8) causes the corresponding output in the globus pallidus internal/substantia nigra (GPI/SNr) to drop to zero, stopping the inhibition of that action. (Left figure after Gurney et al. (2001), fig. 4b).

tional excitation provided by the STN, all of the actions might be concurrently suppressed (meaning more than one action had been selected). These very broad STN connections thus take the input from the hyperdirect pathway, and combine it to provide a level of background excitation that allows only the most inhibited action to be selected.

It should be evident that the example shown in figure 5.2 has carefully selected utilities. In fact, if there are many actions with large utilities, or all actions have low utilities, then this circuit will not function appropriately. For this reason, a control system is needed to modulate the behaviour of these neural groups. Gurney et al. (2001) argue that the globus pallidus external (GPe) is ideally suited for this task, as its only outputs are back to the other areas of the basal ganglia, and it receives similar inputs from the striatum and the STN, as does the globus pallidus internal (GPI). In their model, the GPe forms a circuit identical to that in figure 5.2, but its outputs project back to the STN and the GPI. This regulates the action selection system, allowing it to function across a full range of utility values. This network is shown in figure 5.3.

The model discussed so far is capable of performing action selection and reproducing a variety of single-cell recording results from electrostimulation and

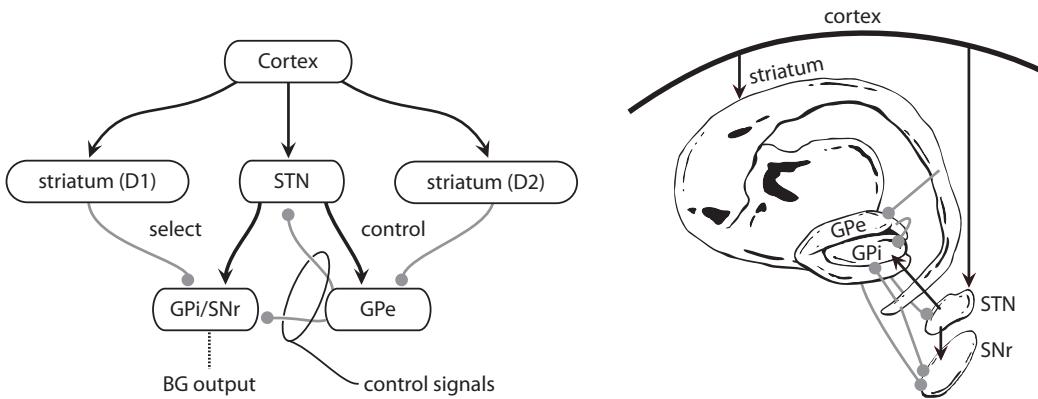


Figure 5.3: The model of action selection in the basal ganglia presented by Gurney, et al. (2001). The striatum D1 cells and the subthalamic nucleus (STN) are as in figure 5.2, while the striatum D2 cells and globus pallidus external form a modulatory control structure. (Left figure after Gurney et al. (2001), fig. 5).

lesion studies. However, it does so with rate neurons; that is, the neurons do not spike and instead continually output a numerical value based on their recent input. This makes it difficult to make precise timing predictions. Furthermore, the model has no redundancy, since exactly one neuron is used per area of the basal ganglia to represent each action. The original version of the model shown in figure 5.3 uses a total of 9 neurons to represent 3 possible actions, and if any one of those neurons is removed the model will fail.

However, given the resources of the NEF and the SPA, these short-comings can be rectified. In particular, the SPA suggests that instead of a single neuron representing potential actions, a mapping from a high-dimensional semantic pointer into a redundant group of neurons should be used. As well the NEF provides a means of representing these high-dimensional semantic pointers and their utilities in spiking neurons, and reproducing the transformations suggested by the original model. I return to the role of SPAs shortly. For the time being, I consider a simpler model that represents actions and utilities as scalar values. The ability of this new model to select actions is demonstrated in figure 5.4. There it can be seen that the highest utility action (B then C then A) is always selected (i.e., inhibited, and so related neurons stop firing).

Crucially, this new implementation of the model allows us to introduce additional neural constraints into the model which could not previously be included. In particular, the types of neurotransmitters employed in the excitatory and in-

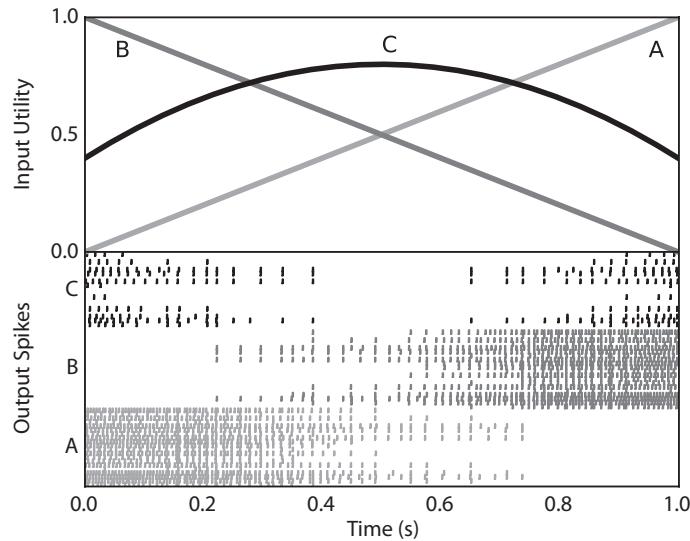


Figure 5.4: Spikes produced (bottom) for three possible actions (A, B, and C) as their utility changes (top). The highest utility action is selected, as demonstrated by a suppression of the spikes of the corresponding action. In order, the highest utility action is B then C then A.

Inhibitory connections of the model have known effects on the timing of a receiving neuron's response. All of the inhibitory connections involve GABA receptors (with time constants between about 6-10ms; Gupta et al. (2000)), while the excitatory ones involve fast AMPA-type glutamate receptors (with time constants of about 2-5ms; Spruston et al. (1995)). The time constants of these neurotransmitters have a crucial impact on the temporal behavior of the model. I discuss these temporal properties and their related predictions in section 5.7.

5.3 Basal ganglia, cortex, and thalamus

We have seen how a spiking neural model of the basal ganglia can be used to select simple actions. However, the purpose of the SPA is to provide a framework for building large-scale cognitive models. These, of course, require complex, sequenced actions, driven by sophisticated perceptual input. In fact, there is evidence that suggests that while single actions can be selected without basal ganglia involvement, chains of actions seem to require the basal ganglia (Aldridge et al., 1993).

Previous chapters demonstrated how the SPA can support representations sufficiently rich to capture the syntax and semantics of complex representations. Here, I consider how these same representations can be used by the basal ganglia to drive chains of cognitive behavior. I begin, in this section, with an overview of the central functions of cortex, basal ganglia, and thalamus, and their interrelations. In the rest of this chapter, I demonstrate how this architecture can implement sequences of fixed action, and subsequently consider how flexible sequences of action can also be realized in this architecture. I return to these issues, in chapter 7, where I present a system choosing among cognitive strategies, applying them, and using the results to drive motor control responses to perceptual input.

To construct such models, we can rely on the well-known and central cortex/basal ganglia/thalamus loop through the brain (see figure 5.5). Roughly speaking, the SPA assumes that cortex provides, stores, and manipulates representations, the basal ganglia map current brain states to future states by selecting appropriate actions, and the thalamus provides for real-time monitoring of the entire system.

Let me begin with a brief consideration of cortex. While cortex is obviously able to perform intricate tasks, in the SPA, most models are built out of combinations of four basic functions: integration (for working memory; section 6.2), multiplication (specifically convolution for syntactic manipulation; section 4.8), the dot product (for clean-up memory and other linear transformations; sections 4.5 and 3.8), and superposition, which is a kind of default operation. I have discussed each of these functions in detail in past tutorials, so will not describe them any further here. The SPA does not carry strong commitments about specific organization of these operations, and this is clearly an avenue for future work. Determining the most appropriate combination of these (and perhaps other) functions to mirror all cortical function is far beyond the scope of this book – as I have been careful to note, we have not yet built a brain! As well, learning clearly plays a large role in establishing and tuning these basic functions. I leave further consideration of learning until later (sections 6.5). My suggestion, then, is that cortex is an information processing resource that is dedicated to manipulating, remembering, binding, etc. representations of states of the world and body. I take this claim to be general enough to be both correct, and largely uninteresting. Far more interesting are the specific proposals for how such resources are to be organized, as captured in the many example models throughout the book.

As I was at pains to point out in the introduction to this chapter, pure processing power is not enough for cognition. In order for these cortical operations to be

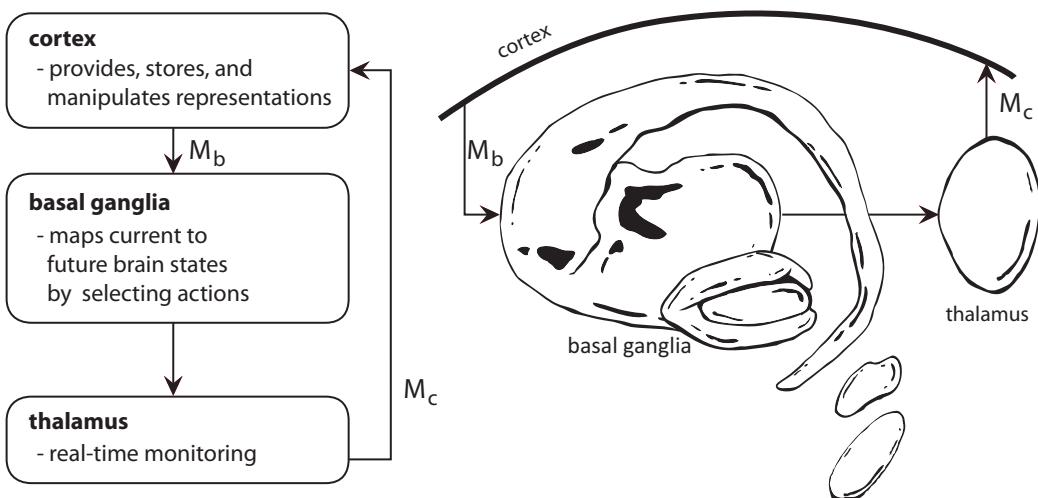


Figure 5.5: The cortex/basal ganglia/thalamus loop. In the schematic on the left, arrows indicate mappings between these three areas. The brain states from the cortex are mapped through the M_b matrix to the basal ganglia. Each row in such a matrix specifies a context for which the basal ganglia can choose an action. Thus the product of the current state and M_b determines how similar the current state is to each of the possible contexts. The output of the basal ganglia disinhibits the appropriate areas of thalamus, as described above. Thalamus, in turn, is mapped through the matrix M_c back to the cortex. Each column of this matrix specifies an appropriate cortical state that is the consequence of the selected action. The relevant anatomical structures are pictured on the right based on a simplified version of figure 5.1.

flexibly exploited, it is essential to control the flow of information between them. This is the role of the basal ganglia model that I have just described. While there is evidence that cortex can perform “default” control without much basal ganglia influence (Kim et al., 1997), the basal ganglia make the control more flexible, fluid, and rapid. As I reviewed above, the central location, and massive projections from (and back to, via thalamus) cortex make the basal ganglia ideal for playing a central role in action selection. All areas of neocortex, except the primary visual and auditory cortices, project to the basal ganglia. In short, in the SPA, as in other cognitive architectures, the basal ganglia effectively exploit the resources of cortex by selecting and implementing appropriate motor and cognitive actions, based on representations available in cortex itself.

Finally, because basal ganglia and cortex are in many ways diffuse, performing many functions over a wide area, and because they are built on top of older, more basic control systems (e.g., brain stem and thalamus), their outputs often need to be integrated with signals coming from elsewhere in the system. It would not do to be stuck “cognizing” about an auditory input while in imminent danger. The thalamus is structured in a way ideally suited for playing the role of a coordinator of these systems. For instance, all output from the basal ganglia goes through the thalamus before returning to cortex, to which thalamus projects broadly. As well, the thalamus receives direct projections from every sense (except smell), and from all cortical areas. In the thalamus, the cortical projections are organized by cortical region, providing a somewhat topographic map of cortex. There are some exceptions, however. Interestingly, the reticular nucleus of the thalamus forms a kind of shell around most of the thalamus, and thus communicates with and regulates the states of many other thalamic nuclei. Consequently, the thalamus is ideally structured to allow it to monitor a “summary” of the massive amounts of information moving through cortex and from basal ganglia to cortex. Not surprisingly, thalamus is known to play a central role in major shifts in system function (e.g. from wakefulness to sleep), and participates in controlling the general level of arousal of the system. In the models presented here, thalamus acts much like a basic relay (as it was long thought to be), because the cognitive coordination of the system can be accounted for by basal ganglia for the considered tasks. I suspect it has a much more important role to play in the overall system, but this role is largely untapped by the examples I consider. Nevertheless, its contribution to timing effects are important, so it is included in the considered models.

As depicted in figure 5.5, communication between cortex and basal ganglia consists of mapping the contents of current cortical states to the striatum and STN (the basal ganglia input nuclei) through the \mathbf{M}_b matrix. This mapping determines

the utilities that drive the basal ganglia model in section 5.2. One natural and simple interpretation of the rows of this matrix is that they specify the antecedent of a rule. Consider the rule “if there is an A in working memory then set working memory to B”. The \mathbf{M}_b matrix can examine the contents of working memory, and output a list of similarities between its rows and working memory with a simple linear transformation (i.e. $\mathbf{s} = \mathbf{M}_b \mathbf{w}$, where \mathbf{s} are the similarities between the each of the rows of \mathbf{M}_b and the vector \mathbf{w} , the input from working memory). That vector of similarities then acts as input to basal ganglia, which selects the highest similarity (utility) from that input.

The output from basal ganglia results in a release from inhibition of the connected thalamic neurons, which are then mapped back to cortex through \mathbf{M}_c . This matrix can be thought of as specifying the consequent of a rule, resulting, for example, in setting working memory to a new state B. More generally, the \mathbf{M}_c matrix can be used to specify any consequent control state given a current cortical state. This loop from cortex through basal ganglia and thalamus and back to cortex forms the basic control structure of the SPA.

Two points are in order regarding this rough sketch of the cortex-basal ganglia-thalamus loop. First, as discussed in section 7.4, the \mathbf{M}_b mapping is more general than the implementation of simple if-then rules. It provides for subtle statistical inference as well. Second, this additional inferential power is available partly because all of the representations being manipulated in this loop (i.e., of cortical states, control states, etc.) are semantic pointers. Let us now turn to consideration of a simple control structure of this form, to illustrate its function.

5.4 Example: Fixed sequences of action

A simple but familiar example of a fixed sequence of action is rehearsal of the Roman alphabet. This is an arbitrary sequence, with no systematic rule connecting one state to its successor (unlike counting, for instance). Consequently, we need to specify 25 rules to be able to traverse the entire sequence. All such rules would be of the form:

```
IF working memory contains letter + A
THEN set working memory to letter + B
```

where bold indicates that the item is a 250-dimensional semantic pointer. For present purposes, these pointers are randomly generated. The inclusion of **letter**

in each of the pointers provides minimal semantic structure that can be exploited to trigger actions appropriate for any letter (see section 5.6).

To implement the IF portion of the rule, an \mathbf{M}_b matrix with rows consisting of all the letter representations is constructed and embedded in the connection weights between cortical working memory and the basal ganglia input (using the methods detailed in section 3.8). This allows the basal ganglia to determine what rule is most applicable given the current state of working memory.

The THEN portion of the rule is then implemented by the \mathbf{M}_c matrix in a similar manner, where only one row is activated by disinhibition (as determined by the basal ganglia), sending a given letter representation to working memory. As working memory is being constantly monitored by the basal ganglia, this new state will drive subsequent action selection, and the system will progress through the alphabet.

To run the model, it is initialized by setting the working memory neurons to represent the **letter + A** semantic pointer. After this, all subsequent activity is due to the interconnections between neurons. Figure 5.6 shows the model correctly following the alphabet sequence. From the spiking pattern we see that the correct action for each condition is successfully chosen by turning off the appropriate inhibitory neurons in the GPi. The top plot is generated by comparing the semantic pointer of each of the 26 possible letters to the current semantic pointer in working memory (decoded from spiking activity) by using a dot product, and plotting the top eight results.

This model demonstrates that a well-learned set of actions with a specific representation as an outcome can be implemented in the SPA. However, this model is not particularly flexible. For instance, we might assume that our working memory is being driven by perceptual input, say from vision. If so, we would have a connection between our visual system and the working memory system that is driving action selection. However, if this is the case, then changing the visual input during the fixed action sequence will cause the sequence to shift suddenly to the new input. In fact, just leaving the visual input “on”, would prevent the model from proceeding through the sequence, since the working memory would be constantly driven to the visually presented input despite the actions of the basal ganglia, as shown in figure 5.7.

In short, the current model is not sufficiently flexible to allow the determined action to be one which actually changes the control state of the system. That is, we are currently not in a position to gate the flow of information between brain areas using the output of the basal ganglia. In this case, we could fix the problem if we could load visual input into working memory only when appropriate. That

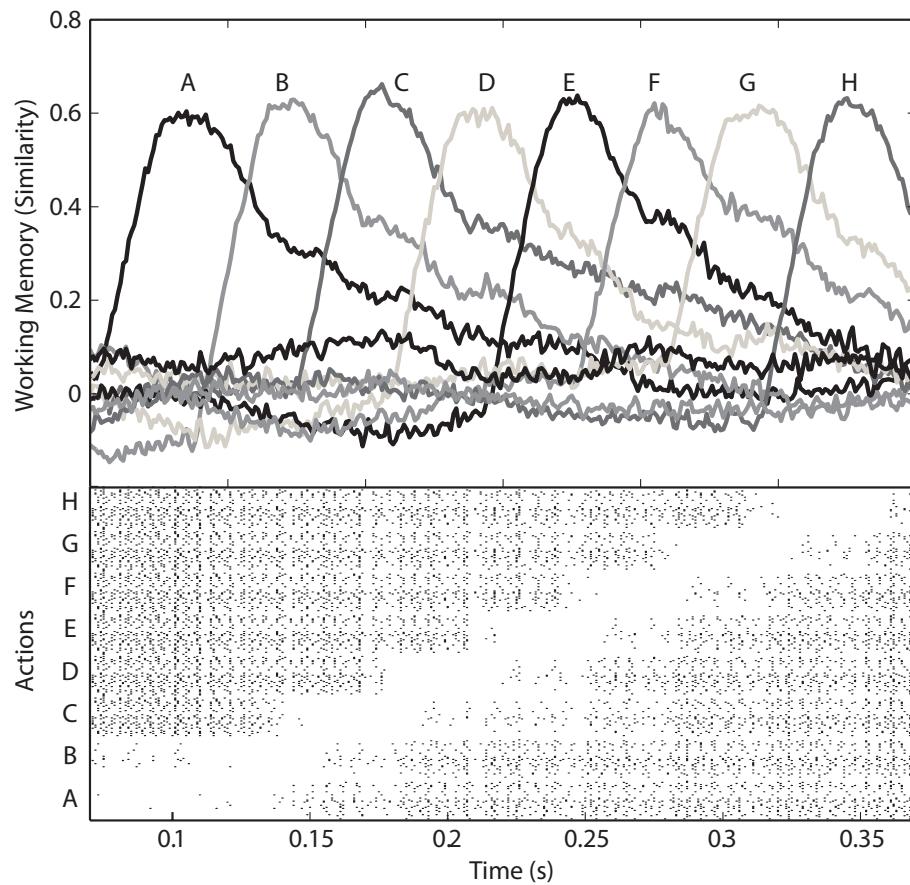


Figure 5.6: Rehearsal of the alphabet. (Top) Contents of working memory generated by taking the dot product of all possible semantic pointers with the decoded contents of working memory (top eight values are shown). (Bottom) The spiking output from GPi indicating the action to perform. The changes in activity demonstrate that the population encoding the currently relevant IF statement stops firing, disinhibiting thalamus and allowing the THEN statement to be loaded into working memory.

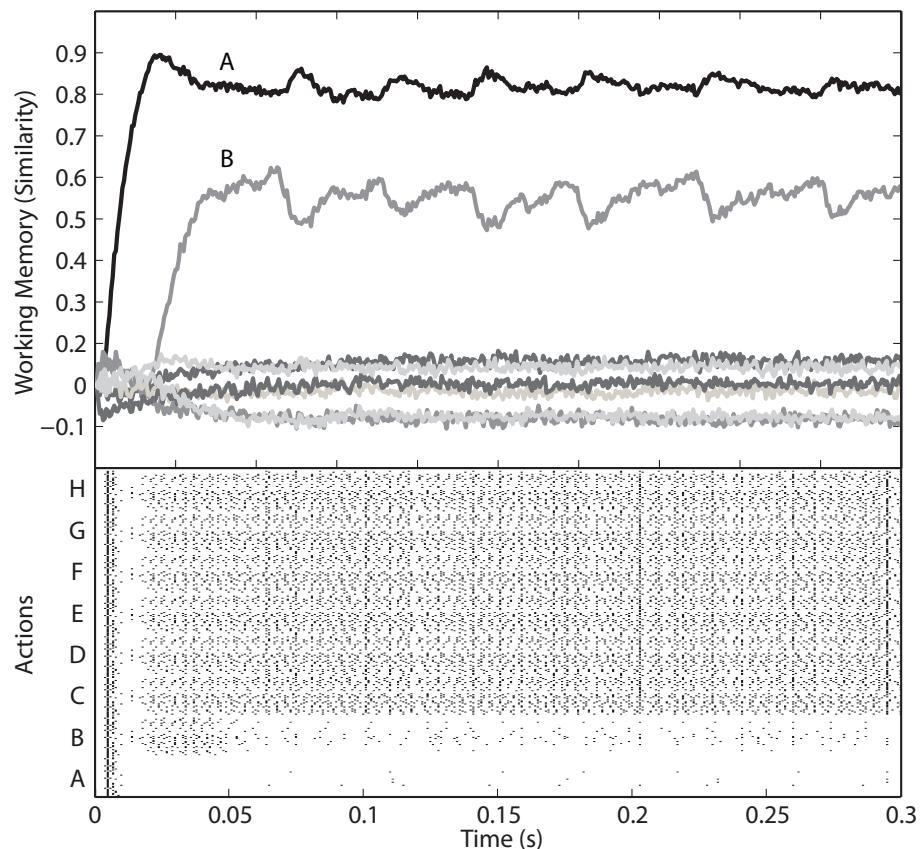


Figure 5.7: Inflexibility of the control structure. The (erroneous) result of connecting the visual system to working memory. (Top) The perceptual system (**letter + A**) continually drives working memory and prevents it from properly moving to **letter + B** and subsequent states. (Bottom) The spiking out from GPi reflects this same “frozen” state.

is, we would like basal ganglia to determine how information is routed in cortex, as well as being able to introduce new representations in cortex. I suspect that routing information flexibly through the brain is a fundamental neural process. I also suspect that this routing is sometimes called “attention”.

5.5 Attention and the routing of information

Attention has been most thoroughly studied in the context of vision. And, while visual attention can take on many forms (e.g., attention to color, shape, etc.), the spatial properties of visual attention are the most thoroughly examined. There are two main considerations when it comes to understanding visuospatial attention: selection and routing (notably analogous to the two aspects of any control problem). Selection deals with the problem of identifying what the appropriate target of the attentional system is given current task demands and perceptual features. Routing, in contrast, deals with how, once a target has been selected, a neural system can take the selected information and direct additional resources towards it. Figure 5.8 identifies the brain areas thought to be involved in both selection and routing. In this section, I only discuss the routing problem, which in general has not received as much consideration as selection (although see Gisiger and Boukadoum (2011) for a recent summary of potential neural mechanisms for routing).

In the last 15 years there have been several proposed models of attentional routing (Olshausen et al., 1993; Salinas and Abbott, 1997; Reynolds et al., 1999; Wolfrum and von der Malsburg, 2007; Womelsdorf et al., 2008). However, none of these uses biophysically plausible spiking neurons, most are purely mathematical models, and few characterize attentional effects beyond a single neuron or cortical area. Consequently, most, if not all, have been criticized as not being scalable (i.e. there are not enough neurons in the relevant brain structures to support the required computations). Given the importance of both scalability and biological plausibility to the SPA, Bruce Bobier in my lab has developed a model of attentional routing called the “Attentional Routing Circuit” (ARC) which addresses these issues.

The ARC draws on past models in several ways. For instance, it relies on nonlinearities to perform attentional routing. It also incorporates connectivity constraints, general architectural considerations, and shares a concern for explaining behavioral and psychophysical data with several of these models. It is most directly a descendant of the “shifter circuit” model (Olshausen et al., 1993). Nev-

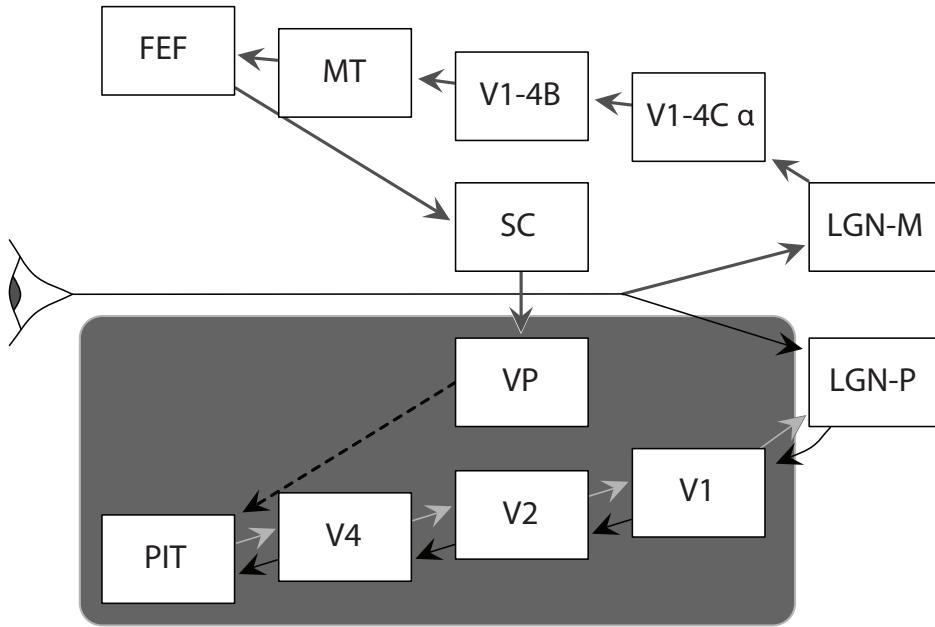


Figure 5.8: Anatomical structures relevant to the attentional routing circuit (ARC). The lateral geniculate nucleus (LGN) is a structure within the thalamus which receives information directly from the optic nerve. The LGN contains magnocellular and parvocellular cells which form the LGN-M and LGN-P layers respectively. The other structures shown are the posterior inferior temporal cortex (PIT), visual cortical areas (V1, V1-4B, V1-4C α , V2, V4), frontal eye fields (FEF), the middle temporal area (MT), superior colliculus (SC), and ventral pulvinar (VP). Feedforward visual information projects through the ventral stream areas (black lines). The pathway from LGN-M leading to FEF (dark gray lines) is one possible route through which the focus of attention (FOA) may be rapidly determined. Areas within the rounded gray box are used to describe the ARC. VP projects a coarse signal (dashed black line) indicating the location of the FOA to control neurons in PIT. Based on the VP signal, local control signals are computed in PIT. The results are then relayed to the next lower level of the ventral stream (light gray lines), where relevant local control signals are computed and again relayed.

ertheless, it improves upon the shifter circuit by being significantly more biologically plausible, scalable, and hence able to account for detailed neurophysiological findings not addressed by that model. Consequently, the ARC provides an especially good account of routing, and one that can be generalized to other parts of the SPA (see section 5.6). Let us consider the ARC in more detail.

As shown in figure 5.8, the ARC consists of a hierarchy of visual areas, for example V1, V2, V4, and PIT, which receive a control signal from a part of the thalamus called the pulvinar (VP). A variety of anatomical and physiological evidence suggests that pulvinar is responsible for providing a coarse top-down control signal to the highest level of the visual hierarchy (Petersen et al., 1985, 1987; Stepniewska, 2004).

A more detailed picture of the connectivity between the visual areas is provided by figure 5.9. In that figure, an example focus of attention that picks out approximately the middle third of the network is shown. To realize this effective routing (i.e. of the middle third of V1 up to PIT), VP provides a control signal to PIT indicating the position and size of the current focus of attention in V1. Control neurons in PIT use this signal to determine what connections to “open” between V4 and PIT, and then send their signal to control neurons in V4. The signal that is sent to the next lower level of the hierarchy (V4) is similarly interpreted by that level to determine what gating is appropriate, and then passed on (i.e., to V2). Again, the gating allows the flow of information only from those parts of the next lower level that fall within the focus of attention. And so this process of computing and applying the appropriate routing signal continues to V1.

To “open” a connection essentially means to multiply it by a non-zero factor. Consider a simple example. Suppose we have a random signal going from population A to population B in a communication channel (see section 3.8). At the inputs to population B, we might insert another signal that can be 0 or 1, coming from our control population, C. If we multiply the inputs to B by C, then we cause the representation in B to be either 0 (if C=0) or the current value of A (if C=1). Population C thus acts as a gate for the information flowing between A and B. The control neurons in ARC are performing a similar function.

In essence, the control neurons determine which lower-level (e.g. V4) neurons are allowed to project their information to the higher level (e.g., PIT). In the ARC, this gating is realized by the non-linear dendritic neuron model mentioned in section 4.2.2. This kind of neuron essentially multiplies parts of its input, and sums the result to drive spiking. Consequently, these neurons are ideal for acting as gates, and fewer such neurons are needed than would be in a two-layer network performing the same function.

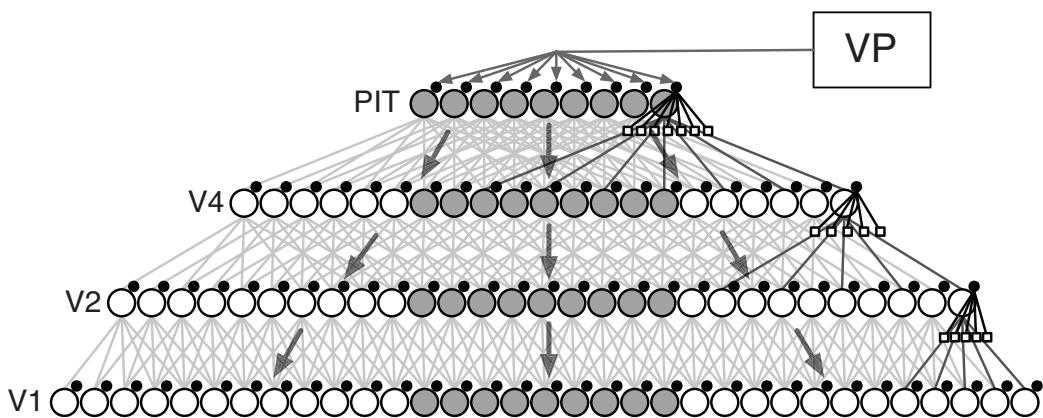


Figure 5.9: The architecture of the attentional routing circuit (ARC) for visual attention. Each level has a columnar and retinotopic organization, where columns are composed of visually responsive pyramidal cells (white circles) and control neurons (black dots). Filled gray circles indicate columns representing an example focus of attention. Neurons in each column receive feedforward visual signals (gray lines) and a local attentional control signal from control neurons (black lines). These signals interact nonlinearly in the terminal dendrites of pyramidal cells (small white squares, only some are shown). Coarse attentional signals from pulvinar (VP) are relayed through each level of the hierarchy downward to control neurons in lower levels (large gray arrows). Control connectivity is highlighted for the rightmost columns only, although other columns in each level have similar connectivity.

The mapping of these computational steps to cells in specific cortical layers of these visual areas is shown in figure 5.10. As shown here, neurons in layer V receive the top-down control signal specifying the size and position of the desired routing. These then project to layer VI neurons that determine an appropriate sampling and shift consistent with the desired routing. The results of this computation are sent to layer IV neurons, which act to gate the feedforward signals into layer II/III neurons, which project to higher levels of the visual hierarchy. A more detailed discussion of the anatomy and physiology underlying this mapping can be found in (Bobier, 2011).

This organization, and the computations it underwrites, is repeated throughout the model (consistent with most contemporary accounts of cortical organization). Consequently, the model, while detailed, is reasonably straightforward as its central features are simply repeated over and over throughout the width and depth of the hierarchy shown in figure 5.9.

Figure 5.11 shows an example of the shifting and resizing of information presented to V1 made possible by this mechanism. As can be seen in that figure, only the right side of the signal in V1 is sent to the top levels of the network's hierarchy. The model preferentially copies data from the nodes that are the focus of attention to higher levels, resulting in the observable scaling and shifting effects. It should be evident that the circuit is thus effectively taking the information on the lowest level of the hierarchy and “routing” it to always fit within the available resources at the highest level. There is ample psychophysical evidence that precisely this kind of processing occurs in the visual system (Van Essen et al., 1991). As well, such a model is able to account for several detailed observations about the neural mechanisms of attention.

For example, it accounts for the known increase in size of the receptive fields of neurons farther up the hierarchy, is consistent with the patchy connectivity in the visual hierarchy (Felleman et al., 1997), and captures the topographic organization of receptive fields in these areas (Tanaka, 1993). The ARC also accounts for the observation that the timing of attentional modulation of neural activity begins at the top of the hierarchy and proceeds down it (Mehta, 2000; Buffalo et al., 2010). Many other anatomical, psychophysical, and physiological properties of attention are also captured by the ARC (Bobier, 2011).

To demonstrate the biological plausibility of ARC, let me consider just one set of experiments in detail. In the experiment reported in Womelsdorf et al. (2006), recordings were taken from monkeys performing a spatial attention task (see figure 5.12). Specifically, the animals fixated a fixation point, after which a cue stimulus (S1) was presented indicating where the animal should covertly attend.

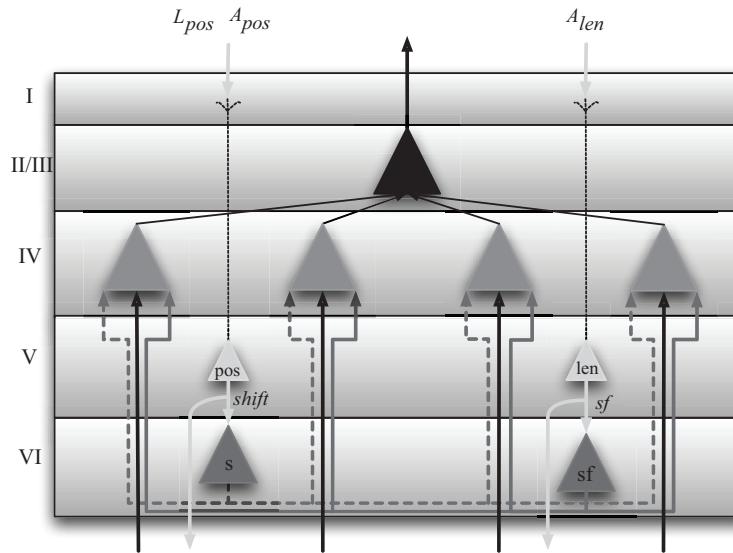


Figure 5.10: Proposed laminar arrangement of neurons in ARC for a single column. Global attention signals (length A_{len} and position A_{pos} of selected region) and a local signal indicating the position of the target in the current level (L_{pos}) are fed back from the next higher cortical level to layer-I, where they connect to apical dendrites of layer-V cells. The layer-V neurons relay this signal to the next lower area with collaterals projecting to control neurons in layer-VI that compute the sampling frequency (sf, solid lines) and shift (s, dotted lines). These signals, along with feedforward visual signals from retina (solid black lines from the bottom) are received by layer-IV pyramidal cells, where the routing occurs (i.e., layer-VI signals gate the feedforward signals). Cells in layer-II/III (black) pool the activity of multiple layer-IV neurons and project the gated signal to the next higher visual area.

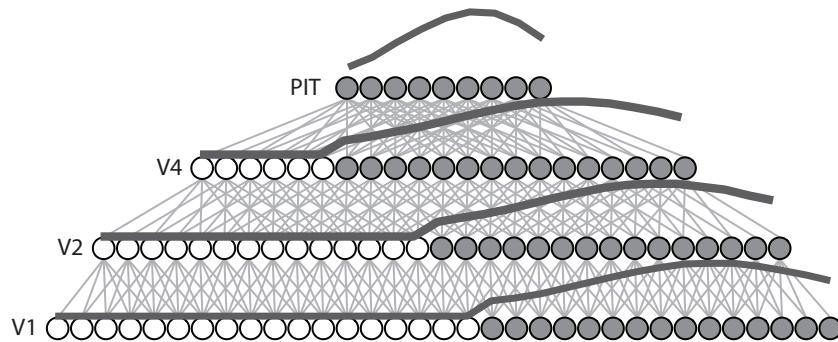


Figure 5.11: An example of routing in the ARC. Levels of the visual hierarchy are laid out as in figure 5.11. Above each of these levels, the thick line indicates the decoded signal represented by the neurons at that level. The darkened circles in each level indicate the focus of attention in the network. In this example, the focus of attention has the effect of both shifting and scaling the original data from V1 to fit within PIT.

Following a delay, three stimuli were presented, one at the target location, and two distractor stimuli (S_2 and S_3) one inside and one outside of the recorded cell's receptive field. The animal had to indicate when it saw a brief change in the stimulus at the cued location, some random interval after the three stimuli were presented. During the interval, the animal was taken to have sustained spatial attention to S_1 , and the receptive field of the cell was mapped. This experimental design allowed Womelsdorf et al. to map the receptive field during sustained states of selective attention to the stimuli inside the receptive field or to the stimulus outside the receptive field.

To compare the model to these experimental results, we ran analogous numerical experiments on the model. We used same methods of fitting the spiking data to determine neuron receptive fields as in the experiment. The main difference between the model and experimental runs is that in the model all spikes could be collected from all neurons. As a result, we could run the experiment on 100 different realizations of the model. Each version has the same general ARC architecture, but the neurons themselves are randomly chosen from a distribution of parameters that matches the known properties of cells in the relevant parts of cortex. Consequently, rather than having just over a hundred neurons from two animals as in the experiment, we have thousands of neurons from hundreds of model-animals. This means that we have a much better characterization of the overall distribution of neuron responses in the model than is available from the

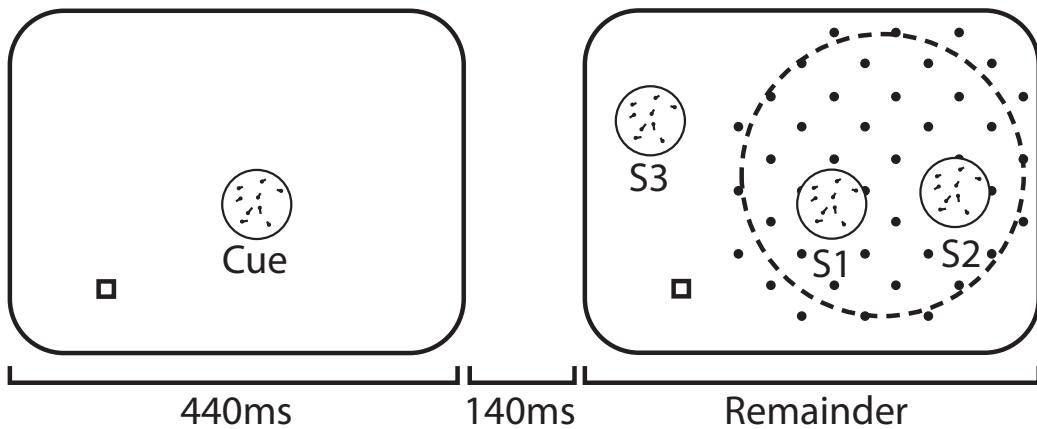


Figure 5.12: The Womelsdorf task (Womelsdorf et al., 2006, 2008). A trial begins when a monkey foveates in a small square area near the fixation point (small square). Then a stationary random dot pattern appears for 440ms (the cue). After a brief delay period, three moving random dot patterns are shown. Two of these (S1 and S2) were placed within the neuron's receptive field (dotted circle). The third (S3) was placed outside the receptive field. The receptive field of the neuron was mapped during the delay between the cue and stimulus change with high contrast probes, that were randomly placed on a grid (black dots). The experiment allowed systematic probing of changes in receptive fields for different attentional targets.

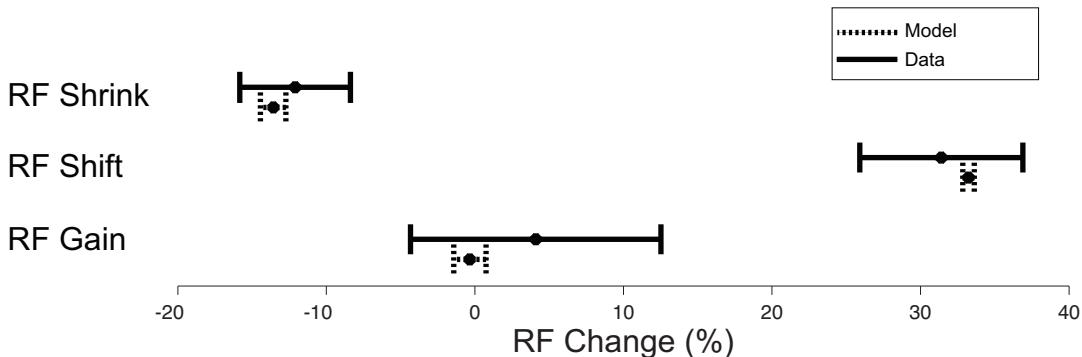


Figure 5.13: Attentional effects (mean and 95% confidence intervals) for 100 model-animals (dashed lines) and data from Womelsdorf et al. (2008) (solid lines). Because we can record from many more spiking neurons in the model, the distribution is much better characterized, and so the confidence intervals are tighter. These results demonstrate that the ARC quantitatively accounts for the receptive field (RF) shrink, shift, and gain observed experimentally.

experimental data for the animals, and we also have a distribution over animals, which is not available from the experimental data as there are only two monkeys.

To compare the model and data, we considered the three main effects described in the experimental work. These effects were seen by comparing attention being directed at a stimulus inside the receptive field to attention being directed at a stimulus outside the receptive field. The effects were: 1) a change in the peak firing rate; 2) a shift of the receptive field center, and 3) a change in the receptive field size. In each case, several statistics were calculated to compare model and experimental data (see Bobier (2011) for details). Overall, the model and data match extremely well on each of these effects, as shown in figure 5.13.

A similar comparison to neurophysiological experiments from Treue and Martinez-Trujillo (1999) and Lee and Maunsell (2010) is performed in Bobier (2011). The first of these experiments demonstrates that the width of tuning to visual features is not affected by attention, although the gain of the neuron responses is affected. The second experiment demonstrates that neuron gain responses are best explained as response gain and not contrast gain. Both sets of experimental results are well captured by the ARC, when the same analyses are used on the experimental and model spiking data.

One final note about this model is that there are only two free parameters. One governs the width of the routing function, and the other the width of feedfor-

ward weights. Both of these parameters are set to match the known receptive field sizes in visual cortex independently of any particular experiment. That is, they were in no way tuned to ensure the ability of the model to predict any specific experiment. The remaining parameters in the model are randomly chosen between different model-animals from distributions known to match general physiological characteristics of cells in visual cortex. This means that the model does not simply capture the mean of the population (of cells or animals), but actually models the *distribution* of responses, which is more difficult. Nevertheless, the model is able to provide a good characterization of changes in several subtle aspects of individual cell activity across a variety of experiments from different labs.

Overall, these results suggest that the detailed mechanism embodied by ARC for routing information through cortex is plausible in its details. The fact that ARC is specified to the level of single spiking cells allows the same analysis applied to the animal data to be applied to the model. Hence matching of results is not a consequence of different analysis methods, but rather of a similarity in the underlying spike patterns. More generally, ARC provides insight into how information can be flexibly routed through the brain. That is, it provides a biologically plausible method for *applying* control, which was missing from our model of action selection.

5.6 Example: Flexible sequences of action

With the ARC characterization of routing in hand, we can return to the model of section 5.4, and repair its shortcomings. As you will recall, the model as presented was unable to appropriately ignore visual input (see figure 5.7). However, if we allow thalamus to generate gating signals that control the flow of information between visual input and working memory, we can prevent such errors.

As mentioned, the model of visual attention presented in the preceding section provides a biologically realistic mechanism for exactly this function. While it is clearly geared to a different part of cortex, it demonstrates how thalamic gating signals (in that case, from pulvinar) can be used to control the flow of information through a cortical circuit. As well, there is nothing physiologically unusual about visual cortex, so it would be unsurprising to see similar functions computed in similar ways in other parts of cortex. In short, this means that computing a nonlinearity between a thalamic input and cortical signals can be considered a plausible and effective means of routing information in many areas of cortex. So, we can introduce a similar circuit into our action-selection model, and provide for

much more flexible control.

The addition of this mechanism means that the model can selectively ignore or be driven by the visual input (see figure 5.14). Notably, in this network a much more general rule is employed than was used before. Specifically, there is now a rule of the form

IF visual cortex contains **letter**+?
THEN copy visual cortex to working memory

This rule can be selectively employed, so the contents of visual cortex can be selectively used to drive future actions. Note that this “copy visual cortex” command is the specification of a control state that consists in gating the information between visual inputs and working memory (unlike the simple representation-to-representation mapping in the previous model). This demonstrates a qualitatively new kind of flexibility that is available once we allow actions to set control states. In particular, it shows that not only the content of cortical areas, but also the communication between such areas, can be controlled by our cognitive actions.

There is a second interesting consequence of being able to specify control states. Notice that the specified rule applies to every letter, not just the one that happens to be in the visual input at the moment. This allows for rules to be defined at a more general category level than in the previous example. This demonstrates an improvement in the flexibility of the system, in so far as it can employ instance specific, or category specific rules. More precisely, the rule can be categorically defined, but apply specifically (i.e., the previous model could have a rule that maps all letters to one other representational state, but not a single rule that systematically maps each letter to a different representational state).

In fact, routing can provide yet more flexibility. That is, it can do more than simply gate information flow between different cortical areas. In the simple alphabet model above, the routing action was essentially “on” or “off” and hence gate-like. However, we can use the same neural structure to actually *process* the signals flowing between areas. Recall that the method of binding semantic pointers is to use circular convolution, which is a linear transformation followed by multiplication. As described in the section on attention, gating can also be accomplished by multiplication (and linear transformations are simple and will not interfere with the multiplication). If we allow our gating signal to take on values between 0 and 1, we can use the same gating circuits to bind and unbind semantic pointers, not only routing, but concurrently processing signals. Essentially, we can give the gating signals useful content.

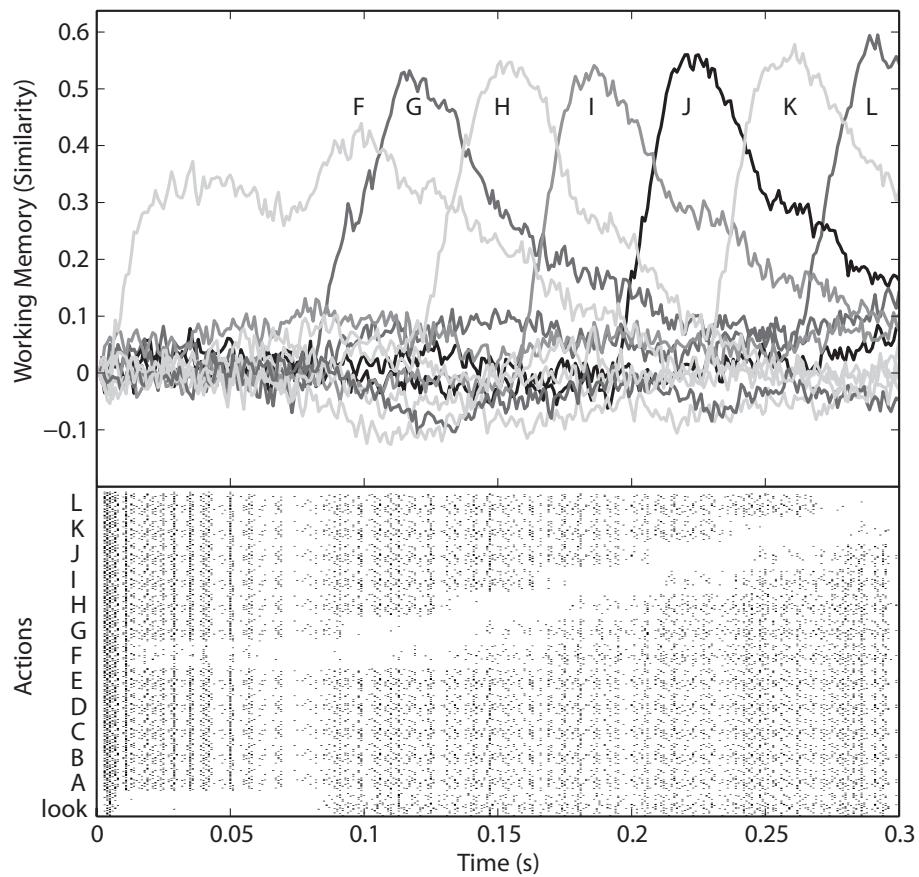


Figure 5.14: Routing information. Contents of working memory are shown on top. The similarity of vocabulary vectors is found with respect to the decoded value of the working memory ensemble. The lower half of the graph shows spiking output from GPi indicating the action to perform. The *look* action takes information from visual cortex (in this case, **letter + F**) and routes it to working memory. Visual input stays constant throughout this run (as in figure 5.7).

Introduce this simple extension means that the same network structure as above can be used to perform syntactic processing. So, for example, we can implement a dynamic, controlled version of the question answering network described in section 4.3. In this network, we define semantic pointers that allow us to present simple language-like statements and then subsequently ask questions about those statements. So, for example, we might present the statement

statement + blue ⊗ circle + red ⊗ square

to indicate that a blue circle and red square are in the visual field. We might then ask a question in the form

question + red

which would be asking “What is red?”. To process this input, we can define the following rules

IF the visual cortex contains **statement+?**
THEN copy visual cortex to working memory

which simply gates the visual information to working memory as before. We can also define a rule that performs syntactic processing while gating

IF visual cortex contains **question+?**
THEN apply visual cortex to the contents of working memory

Here, “apply” essentially indicates that the contents of visual cortex are to be convolved with the contents of working memory, and the result is stored in the network’s output. More precisely, the contents of visual cortex are moved to a visual working memory store (to allow changes in the stimulus during question answering, as above), and the approximate inverse (a linear operation) of visual working memory is convolved with working memory to determine what is bound to the question. This result is then stored in an output working memory to allow it to drive a response. The results of this model answering two different questions from the same remembered statement are given in figure 5.15. These two generic rules can answer any question provided in this format.

Notably, this exact model can reproduce all of the control examples presented to this point. This means that the introduction of these more flexible control structures does not adversely impact any aspects of the simpler models’ performance as described above. This is crucial to claims of flexibility. The flexibility of the SPA needs to be independent of its particular use: flexibility needs to reside in

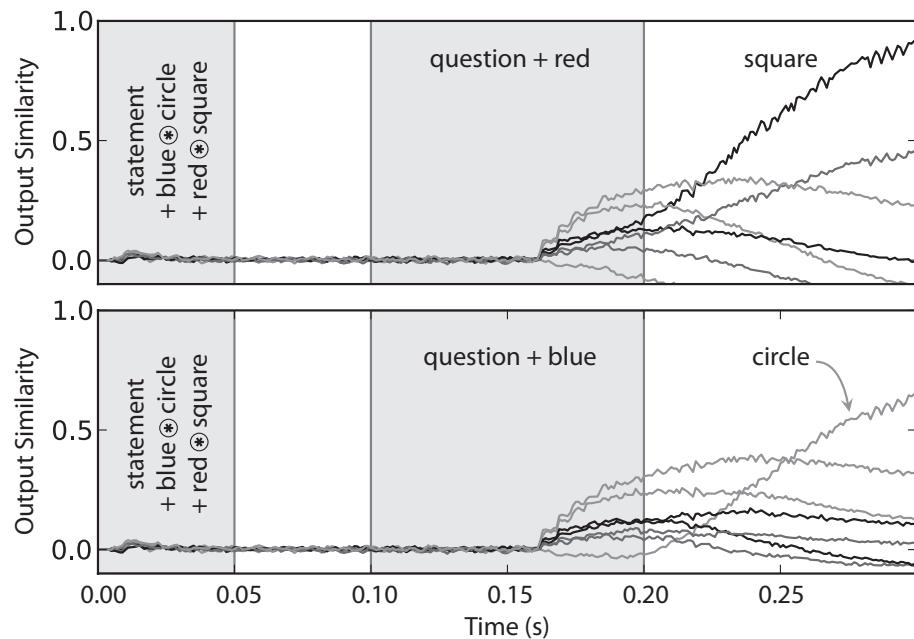


Figure 5.15: Answering two different questions starting from the same statement. gray areas indicate the period during which the stimuli were presented. The similarity between the contents of network's output and the top 7 possible answers is shown. The correct answer is chosen in both cases after about 50ms.

the overall design of the system, not the task specific changes introduced by a modeller.

In addition, we can be confident that this control circuit will not adversely affect the scaling of the SPA. Only about 100 neurons need to be added for each additional rule in the basal ganglia. Of those, about 50 need to be added to striatum, which contains about 55 million neurons (Beckmann and Lauer, 1997), 95% of which are input (medium spiny) neurons. This suggests that about one million rules can be encoded into a scaled-up version of this model. In combination with the reasonable scaling of the representational aspects of the SPA (see section 4.5), this suggest that the SPA as a whole will scale well.

So, the detailed neural mechanisms for routing in attention explored in the previous section can provide a general account of routing in a wide variety of circuits. This is important because it means that this routing mechanism lends biological detail to the SPA *and* the inclusion of this mechanism in the SPA helps make this form of routing a plausible general mechanism. That is, we can show that this detailed, spiking approach to routing can be usefully integrated into a functional large-scale model, unlike past neural mechanisms that have been suggested to account for routing (Gisiger and Boukadoum, 2011).

5.7 Timing and control

Because the model of the basal ganglia that we have presented is constructed using a variety of biological constraints, we are able to ask questions of this model that have not been addressed adequately in past models. Specifically, because we know the kinds of neurons, their spiking properties, and the temporal properties of the neurotransmitters in basal ganglia, we can make specific predictions about the timing of action selection (Stewart et al., 2010a).

For example, Ryan and Clark (1991) showed that in the rat basal ganglia the output neurons stop firing 14 to 17ms after a rapid increase in the utility of one of the possible actions. It is a simple matter to run such an experiment on the basal ganglia model. To keep things simple, we can include two possible actions, A and B, and rapidly change the utility by directly changing the input to striatum (i.e., by controlling the best matching action to current cortical activity). An example run of such a model is shown in figure 5.16a, which results in a cessation of firing in approximately 15ms.

More interestingly, we can explore the effects of the difference in utility between the two actions on the length of time such firing persists. These more infor-

mative results are presented in figure 5.16b, where it can be seen that the latency can increase to about 38ms for actions that are only slightly different in utility after the change. For the largest utility differences, the latency drops to about 14ms, the same lower bound as the Ryan and Clark experiment. As far as I am aware, this latency profile as a function of utility differences remains an untested, but strong prediction of the model.

We can also look at the effects on timing of the basal ganglia in the context of the entire cortex-basal ganglia-thalamus loop. In fact, this timing is implicitly demonstrated by figure 5.6. There it can be seen that in the fixed action selection case, it takes about 40ms for the system to switch from one action to another. This is more fully characterized in figure 5.17 where the mean timing is shown over a range of time constants of the neurotransmitter GABA. GABA is the main neurotransmitter of the basal ganglia, and has been reported to have a decay time constant of between 6 and 11ms (Gupta et al., 2000). We have identified this range on the graphs with a gray bar. On this same graph, we have drawn a horizontal line at 50ms because this is the standard value assumed in most cognitive models for the length of time it takes to perform a single cognitive action (Anderson et al., 1995a; Anderson, 2007).

Interestingly, the cycle time of this loop depends on the complexity of the action being performed. Specifically, the black line in figure 5.17 shows the simplest fixed action cycle time, whereas the gray line shows the more complex flexible action cycle time, like those discussed in section 5.6. The main computational difference between these two types of action is that the latter has a control step, which can re-route information through cortex. The cycle time is affected by increasing from about 30-45ms in the simple case, to about 60-75 ms in the more complex case. These two instances clearly bracket the standard 50ms value. Notably, this original value was arrived at through fitting of behavioural data. If the tasks used to infer this value include a mix of simple and complex decisions, arriving at a mean between the two values described here would be expected.

Together, these timing results help to highlight some of the unique properties of the SPA. For example, these elements of the SPA can help provide an explanation of the genesis of certain “cognitive constants” that would not otherwise be available, and it can address available neural data about the impact of utility on selection speed. These explanations cannot be provided by a basal ganglia model implemented in rate neurons, because they do not include the relevant timing information (e.g., neurotransmitter time constants). They are also not available from more traditional cognitive modelling approaches, such as ACT-R, that have determined such constants by fits to behavioural data. As well, these explanations are

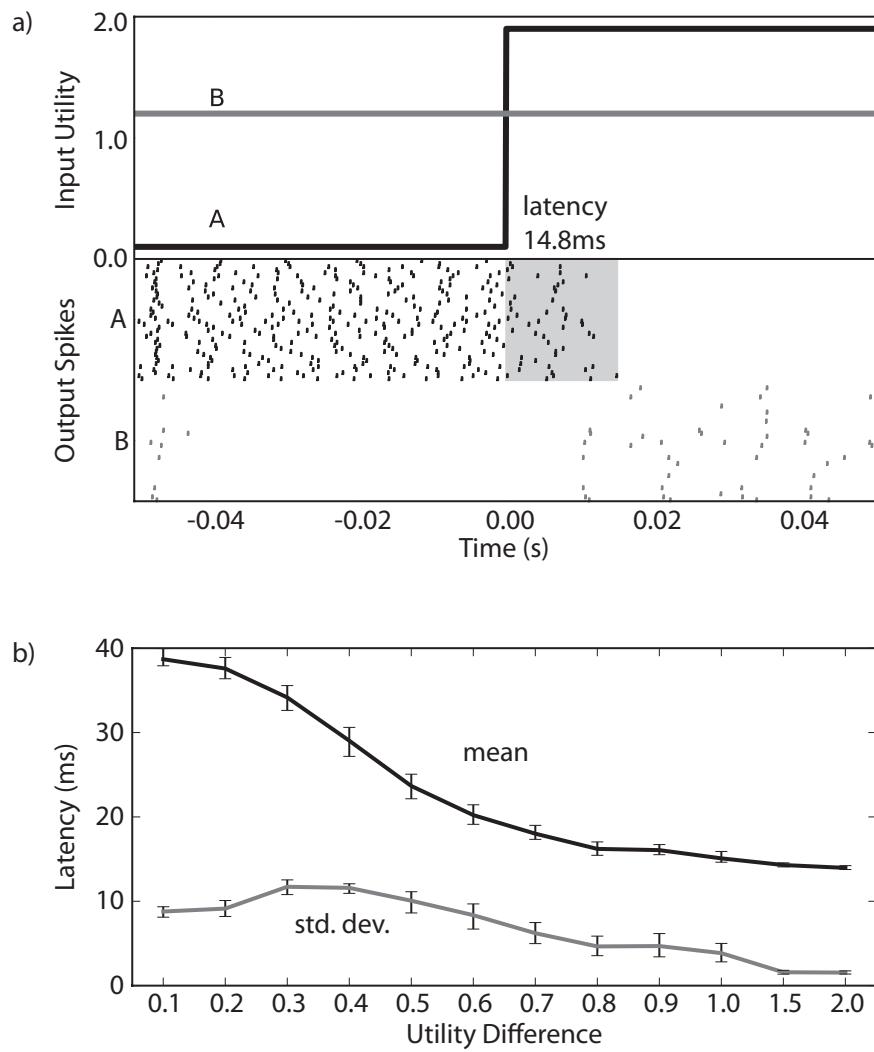


Figure 5.16: Timing predictions for changes in action utility. a) An example of the delay time between a rapid change in utility (top) and the change in spiking activity choosing a new action (bottom). Here, firing for action A stops 15.1ms after the change in utility (hence it is chosen). b) Averages and standard deviations of such changes over many utility differences. Error bars are 95% confidence intervals over 200 runs.

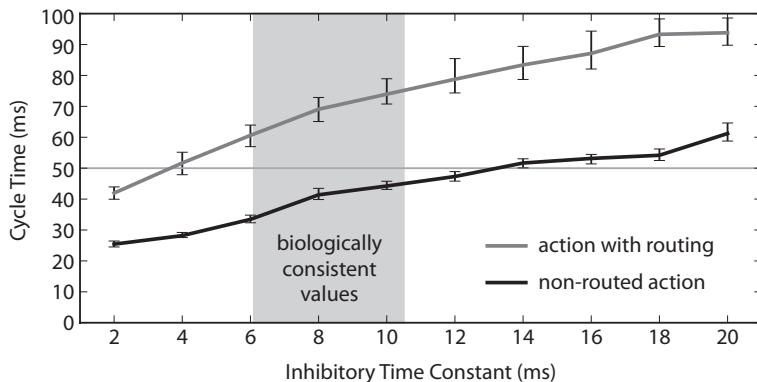


Figure 5.17: Effects of neurotransmitter time constant on the timing of cognitive actions. This plot shows the effect that increasing the time constant of the GABA neurotransmitter has on cycle time and compares a simple, non-routed action to a more complex action involving routing. Biologically plausible values for the time constant of the neurotransmitter are highlighted with the gray bar. The gray horizontal line indicates a 50ms cycle time, which is usually assumed by cognitive models in their matches to behavioral data (Anderson, 2007).

not available from other neurally-inspired architectures that include basal ganglia as an action selector, such as Leabra, because the winner-take-all mechanisms in such models do not have temporal constraints.

In general, the dynamical properties of the SPA do not merely help us match and explain more data, they also suggest behavioural and neurobiological experiments to run to test the architecture. Behaviorally, the SPA suggests that it should be possible to design experiments that distinguish the kinds of cognitive action that take longer (or shorter) to execute with reference to their complexity. Physiologically, the SPA suggests that it should be possible to design experiments that manipulate the length of time neurons in basal ganglia fire after actions are switched. Results from such experiments should provide a more detailed understanding of the neural underpinnings of cognitive control.

In general, the central theme of this chapter, “control”, is tightly tied to neural dynamics. Although I have discussed control more in terms of flexibility of routing information because of my interest in presenting an architecture that can manipulate complex representations, the dynamical properties underlying such control are ever-present and unavoidable. In many ways, measuring these dynamical properties is simpler than measuring informational properties, because *when*

an event occurs (such as a spike or a decision) is more amenable to quantification (e.g., using a clock) than determining *what* such an event represents (as such representations may be complex). But, of course, the “when” and the “what” *together* determine behavior. As a result, the dynamics of such processes provide a kind of independent measure of the underlying information processing being performed. Since we are interested in architectures that realize such processes, those that can be constrained by empirical data relating to both dynamical and informational properties are subject to stricter constraints. This is why the ability of the SPA to relate to temporal constraints, while performing interesting information processing, is an important strength of the approach.

There are, of course, many more kinds of temporal constraints than those I have discussed in this chapter. In the next chapter, I consider temporal constraints related to learning and memory. Though the behavior under consideration is different, the theme is the same: the resources of the SPA put it in a position to provide a biologically realistic account of the detailed dynamical and information processing processes underlying cognition. In support of that theme, let me consider one more example – one which combines flexible sequences of action, routing, and binding to solve a classic task in cognitive science.

5.8 Example: The Tower of Hanoi

The Tower of Hanoi task involves three pegs and a fixed number of disks of different sizes with holes in them such that they can be placed on the pegs. Given a starting position that stacks disks from largest to smallest on the first peg, the goal is to move all of the disks to the third peg, subject to the constraints that only one disk can be moved at a time and a larger disk cannot be placed on top of a smaller disk (see figure 5.18). The Tower of Hanoi task has a rich history in cognitive science as a good example of human problem solving (e.g., Simon, 1975), and as a result, there are a variety of symbolicist cognitive models that match expert human behaviour well (e.g., Altmann and Trafton, 2002).

There are also several neural network models that solve the task, but these typically treat the problem as a pure optimization problem, and show that an abstract network architecture (e.g., a Hopfield network, or a three-layer network) can solve a problem that can be mapped to the task (e.g., Kaplan and Gzelis, 2001; Parks and Cardoso, 1997). To the best of my knowledge, there are no biologically-based neural architectures that have been shown to capture the details of human performance. This is likely because a model must implement rule following, planning,

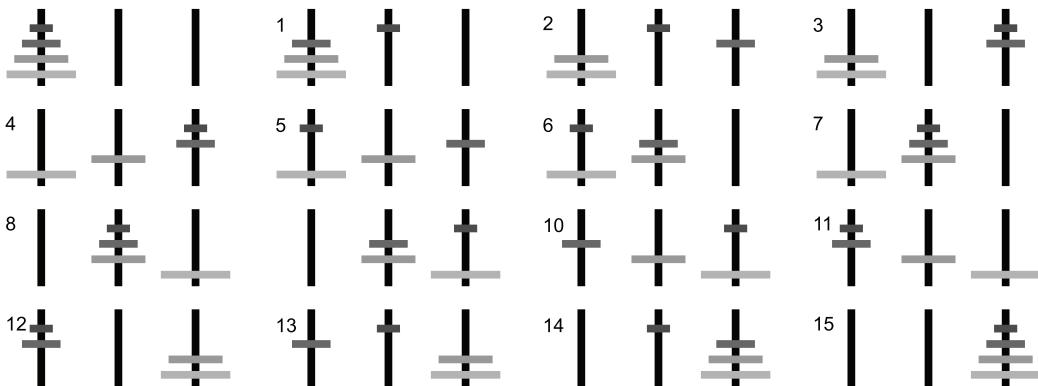


Figure 5.18: The Tower of Hanoi task for four disks. Each move made by an expert solver is shown. Simon’s (1975) original “Sophisticated Perceptual Strategy” results in these moves, and is the basis for the neural model presented here.

and goal recall in order to solve the task in a human-like manner. Of course, the point of the SPA is to provide a means of constructing just such models that conform to the known anatomy, connectivity, and neural properties of the basal ganglia, thalamus, and cortex. Terry Stewart has recently exploited these properties of the SPA to construct just such a model (Stewart and Eliasmith, 2011).

There are many algorithms that can be used to produce the series of moves shown in figure 5.18. As a result, one focus of cognitive research on this task is determining which algorithm(s) people are using. This is done by examining factors such as the time taken between steps and the types of errors produced. In this SPA model, the basic algorithm used is the “Sophisticated Perceptual Strategy” suggested by Simon (1975). As a subject who employs this algorithm, we start with the largest disk that is not in the correct location. We then examine the next smaller disk. If it blocks the move we want to make, then our new goal is to move that disk to the one peg where it will not be in the way. We then iterate this algorithm, going back to previous goals once we have accomplished the current one.

To implement this algorithm, we need to keep track of three things: the disk we are currently attending to, the disk we are trying to move, and the location we are trying to move it to. To keep track of these varying states, we can introduce three cortical working memory circuits, which are called ATTEND, WHAT, and WHERE for ease of reference (see figure 5.19).

In addition, the algorithm requires the storage and recall of old goals regarding

which disk to place where. Storing a single goal such as “disk 4 on peg C” would be easy: we could simply add the vectors together ($\text{disk4} + \text{pegC}$) and store the result. However, multiple goals cannot be stored in this manner, as $(\text{disk4} + \text{pegC}) + (\text{disk3} + \text{pegB})$ cannot be distinguished from $(\text{disk4} + \text{pegB}) + (\text{disk3} + \text{pegC})$. This, of course, is another instance of the binding problem discussed in section 4.2. So, to store a set of goals, we compute the sum of the bound vectors $\text{disk4} \otimes \text{pegC} + \text{disk3} \otimes \text{pegB}$. Then, to recall where we wanted to place a particular disk (e.g. disk3), we can unbind that disk from that goal state.

Finally, we need to consider what input and output is needed for the model. Here, the focus is on the cognitive operations, so we do not need to consider motor or perceptual systems in detail. As a result, we can assume that perceptual systems are able to supply the location of the currently attended object (`ATTEND_LOC`), the location of the object we are trying to move (`WHAT_LOC`), and the final end goal location of the object we are trying to move (`GOAL_LOC`). Similarly, the two output motor areas in the model will have the results of deliberation that indicate what disk should be moved (`MOVE_WHAT`) and where it should be moved to (`MOVE_WHERE`).

We have now specified the essential architecture of the model, and can turn to characterizing the set of internal actions the model needs to perform, and the conditions under which it should perform each action. These rules define the \mathbf{M}_b (IF) and \mathbf{M}_c (THEN) matrices from figure 5.5. For each action, we determine what state cortex should be in for that action to occur, and connect the cortical neurons to the basal ganglia using the resulting \mathbf{M}_b . We then determine what cortical state should result from being in the current cortical state, and connect thalamus to cortex using the resulting \mathbf{M}_c . A full listing of the 16 rules used in the final model is in appendix C.1.

Let us consider a few steps in this algorithm for illustration. First, the model will `ATTEND` to the largest disk (placing the disk4 vector into the `ATTEND` memory). Next, the model forms a goal to place disk4 in its final location, by routing `ATTEND` to `WHAT` and `GOAL_LOC` to `WHERE`. The model now has disk4 in `ATTEND` and `WHAT` and pegC in `WHERE`. Next, it checks if the object it is trying to move is in its target location. If it is ($\text{WHERE} = \text{WHAT_LOC}$), then it has already finished with this disk and needs to go on to the next smallest disk (loading disk3 into `WHAT` and routing `GOAL_LOC` to `WHERE`).

If the disk in `WHAT` is not where it is trying to move it (i.e., `WHERE` is not equal to `WHAT_LOC`), then it needs to try to move it. First, it looks at the next smaller disk by sending disk3 to `ATTEND`. If it is attending to a disk that is not the one it is trying to move (`ATTEND` is not `WHAT`) and if it is not in the way (`AT-`

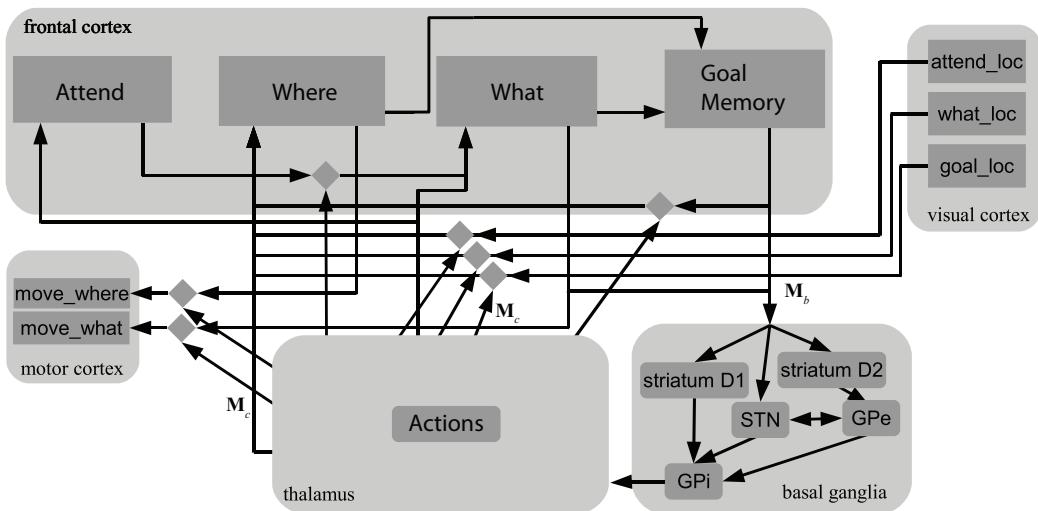


Figure 5.19: The architecture used for the Tower of Hanoi. As described in more detail in the text and appendix C.1, elements in frontal cortex act as memories, with the goal memory including the ability to bind items to keep track of multiple goals. Visual and motor cortices are treated as inputs and outputs to the model, respectively. The basal ganglia and thalamus are as described in figure 5.3. Note that diamonds indicate gates that are used for routing information throughout cortex, and are largely the targets of selected actions. M_b is defined through the projections to the basal ganglia, and M_c is defined through the multiple projections from thalamus to cortex (only two of which are labeled with the matrix for clarity). The model has a total of about 150,000 neurons.

TEND_LOC is not WHAT_LOC or WHERE), then attend to the next smaller disk. If it is in the way (ATTEND_LOC=WHAT_LOC or ATTEND_LOC= WHERE), then it needs to be moved out of the way.

To do this, the model sets a goal of moving the disk to the one peg where it will not be in the way. The peg that is out of the way can be determined by sending the value pegA + pegB + pegC to WHAT and at the same time sending the values from WHAT_LOC (the peg the disk it is trying to move is on) and ATTEND_LOC (the peg the disk it is looking at is on) to WHAT as well, but multiplied by -1. The result will be pegA + pegB + pegC - WHAT_LOC - ATTEND_LOC, which is the third peg. This algorithm, with the addition of a special case for attending the smallest disk (disk1 can always be moved, since nothing is ever in its way, and if the model working with disk1 without finding anything in the way, then it can move the disk it is trying to move), is sufficient for solving Tower of Hanoi.

However, this algorithm does not make good use of the memory system, so it results in rebuilding plans each time a disk is moved. To address this, we can first add a rule to do nothing if RECALL is not the same as WHERE. This occurs if the model has set a new goal, but there has not been enough time for the memory to hold it. Next, we can add rules for the state where the model has just finished moving a disk. Instead of starting over from the beginning, the model sends the next largest disk to ATTEND and WHAT and routes the value from RECALL to WHERE. This recalls the goal location for the next largest disk and continues the algorithm.

To fully implement this algorithm, including better use of the memory system, requires specifying 16 actions in the two **M** matrices (see table C.2). The behavior of the model is captured by figure 5.20. The model is able to successfully solve the Tower of Hanoi, given any valid starting position and any valid target position. It does occasionally make errors, and recovers from them (our analysis of these errors is ongoing). Figure 5.20a shows several examples of successful action selection, where the spiking output from the basal ganglia to the thalamus is shown. As can be seen, different groups of neurons stop firing at the same time, releasing the inhibition in the thalamus and allowing that particular action to be performed. Unfortunately, it is difficult to directly compare this kind of single cell response with human data, because such data is generally scarce, and does not exist for this particular task to the best of my knowledge.

However, we can compare the model to human performance in other ways. For instance, Anderson, Kushmerick, and Lebiere (1993) provide a variety of measures of human performance on this task. Figure 5.20b compares the time taken between each move in the case where no mistakes are made. There are only

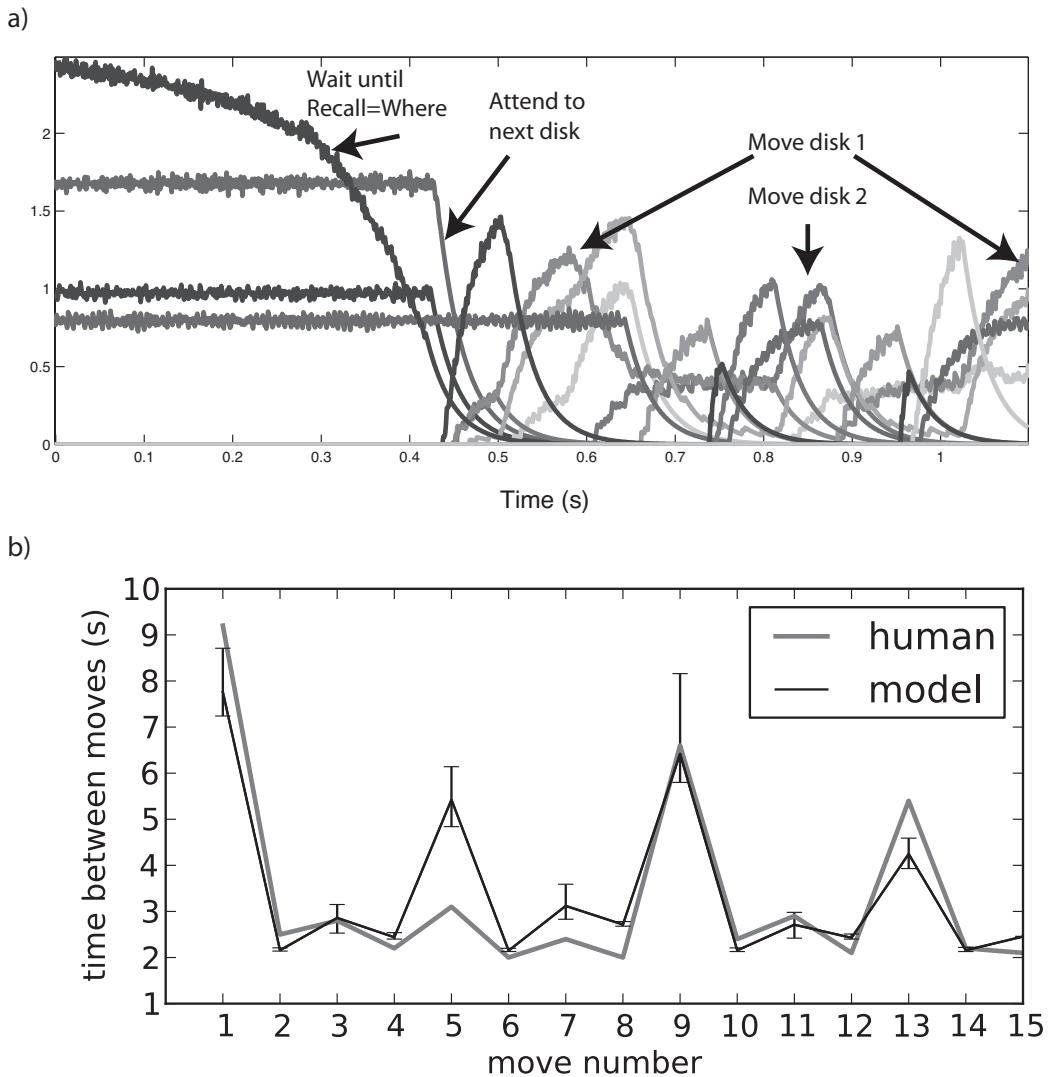


Figure 5.20: Behavior of the Tower of Hanoi model. a) The decoded spiking activity from basal ganglia to thalamus. This activity shows which actions are selected as the task progresses. Different shades of grey indicate the activity of basal ganglia neurons associated with different actions. Some actions are labelled as examples. b) Behavioral performance of the model compared to human subjects.

two free parameters for the model: the amount of time needed to move a disk (1.8 seconds) and the memory weighting factor that determines how quickly new information is encoded in memory (0.08). All other timing parameters are taken from the neurophysiology of the various brain regions (e.g. time constants of the relevant neurotransmitter types, membrane time constants, etc.). As a result, the timing behavior of the system is not the result of a many parameter fit to the human data, but rather falls naturally out of the dynamics intrinsic to the SPA.

As can be seen, in both the human and model data, steps 1, 9, and 13 show longer pauses as a new set of goals are established. As well, shorter pauses on steps 3, 7, and 11 are also consistent across the model and data. The only point of disagreement is step 5, where the model is taking longer than the human. We suspect this is because humans are employing a heuristic shortcut that we have not included in the set of rules used in the model (analogous to the use of the memory system not originally included). In any case, the overall agreement between the model and human data is quite good, suggesting that this is the first spiking, neurally based model to capture many major features of biological cognition on this task.

However, one main point of constructing a biological model is to make better contact with neural data, not just to explain behavioral data. As I have mentioned, the low-level spiking data is not available, but high-level fMRI data is. As shown in figure 5.21, the SPA model matches well to the fMRI activity recorded in a variety of cortical areas. While a symbolist model of this data (ACT-R; Anderson et al., 2005) has been shown to have a similar fit, there are crucial differences between how the SPA generates these results and how ACT-R does.

First, like most cognitive models, ACT-R will provide the same prediction for each run of the model. As a result, the model predicts only mean performance. In contrast, the SPA model predicts a distribution of results (as shown by the standard deviation plotted in figure 5.21). The SPA model is thus much more amenable to modeling individual differences found in populations of subjects.

Second, the SPA generates the predictions based on a biophysical property of the model: neurotransmitter usage (see figure 5.21b). That is, fMRI data is predicted by neurotransmitter usage, which is filtered by a fixed model of how such fluctuations demand energy and give rise to the blood-usage-based BOLD signal (i.e. the BOLD filter). It is the BOLD signal that is actually measured by MRI machines. In contrast, the ACT-R model relies on the proportion of time a module is used to generate its predictions. There is thus no specification of a physiological process that underwrites the BOLD signal in ACT-R predictions. This also has the consequence that for the ACT-R fit to the fMRI data the BOLD filter is fit

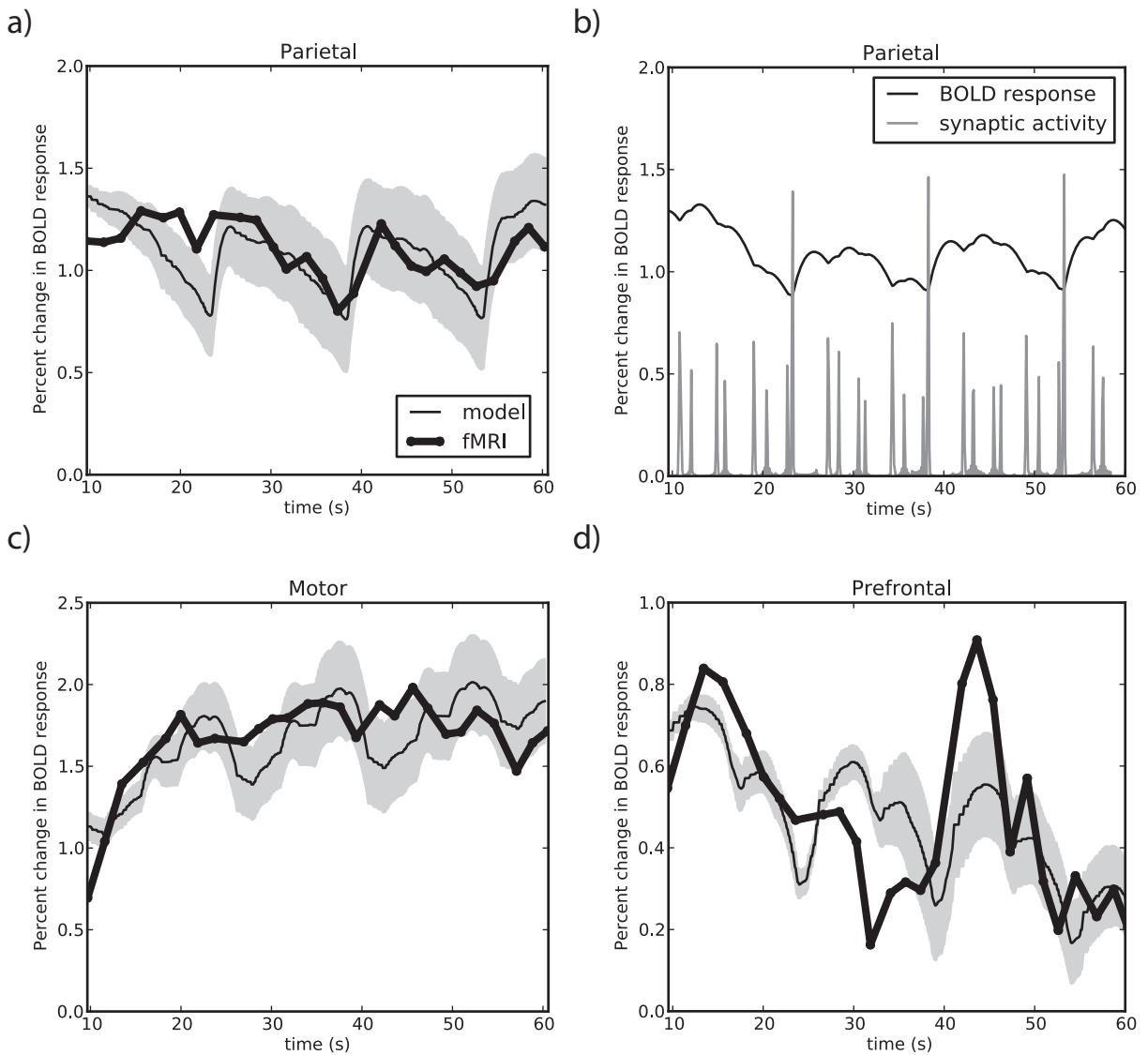


Figure 5.21: fMRI results comparing humans data to the SPA model during a Tower of Hanoi-like task. Human fMRI data is from Anderson et al. (2005). Scans were taken when subjects pressed a button indicating a move. The first button press was at 10 seconds. Thick black lines are the average human fMRI data. Thin black lines are the average model data. a) Parietal cortex activity. One standard deviation is shown in grey. b) The parietal activity from a single run. Gray lines indicate the synaptic activity (neurotransmitter usage) that is used to generate the fMRI prediction, which is shown in black. c) Motor cortex activity. d) Prefrontal cortex activity.

differently for each cortical area, which is difficult to justify physiologically.

Overall, the Tower of Hanoi model provides a concrete example of how the SPA can help bridge the gap between the biological substrate, and high-level cognitive behavior. While the model is anatomically and physiologically constrained and run at the level of individual spiking neurons, it demonstrates rule following, planning, and goal recall. This allows the model to be simultaneously compared to both neural data from fMRI, and more traditional reaction time data typically gathered in cognitive psychology.

5.9 Nengo: Question answering

This tutorial brings together methods used in each of the previous tutorials, as well as introducing a basic control structure. This will allow us to construct a moderately complex model that relies on SPA methods for characterizing representation, transformation, and control. The specific goal of this tutorial is to build a network that will output answers to questions based on supplied statements and questions. The statements and questions are supplied to the network as semantic pointers placed in a ‘visual cortical area’ and the output answers are semantic pointers generated by the model and sent to a ‘motor cortex’.

The final question answering network is somewhat complex, so I will introduce it in three stages. The first network that we make binds two semantic pointers together, and then binds the result with the pseudo-inverse of a third semantic pointer. Note that this is very similar to the network in the preceding tutorial (section 4.8). However, in this network, unbinding is accomplished by calculating the relevant inverse as part of the operation, rather than having an inverted semantic pointer supplied.

- Open a blank Nengo workspace and create a new network named ‘Question Answering’.
- Create five new ‘default’ ensembles with the names ‘A’, ‘B’, ‘C’, ‘D’, and ‘E’. Use 300 neurons and 20 dimensions per ensemble.
 - If ensemble creation is slow, once the first ensemble is made it can be copied and pasted to form the other four ensembles. Select ‘A’ and use the right-click menu to copy the ensemble. Then right-click inside the network to paste (and rename) the ensemble. This will result in the same neurons in each ensemble, instead of each being randomly

generated, but that will not significantly affect the operation of this network.

- Drag the ‘Binding’ template into the network. Set *Name* to ‘Bind’ and *Name of output ensemble* to ‘D’. Use 30 neurons per dimension and don’t invert either input.
- Project from the ‘A’ and ‘B’ ensembles to the ‘A’ and ‘B’ terminations on the ‘Bind’ network.
- Drag another ‘Binding’ template into the network. Set *Name* to ‘Unbind’ and *Name of output ensemble* to ‘E’. Use 30 neurons per dimension and check the box to invert input A but not B.
- Project from the ‘C’ ensemble to the ‘A’ termination on the ‘Unbind’ network and from the ‘D’ ensemble to the ‘B’ termination of the ‘Unbind’ network.
 - To select multiple network elements to arrange them, hold *Shift* while dragging around items to select them. Then release *Shift* and move the selected items.
- Open *Interactive Plots* and ensure the semantic pointer graphs for ‘A’, ‘B’, ‘C’, ‘D’, and ‘E’ are shown.
- Use *Set value* on the semantic pointer graph of ‘A’ and set it to ‘RED’. Similarly, set the value of ‘B’ to ‘CIRCLE’, and the value of ‘C’ to ‘RED’.
- Run the simulation for about 1 second.
 - Depending on your computer, this may run slowly. To speed up the simulation, you can decrease the ‘recording time’ to 0.5 (click the gray arrow in the middle bottom of the window). You can also use fewer neurons in the ensembles, but the simulations may not work well.

As seen in the previous tutorial, binding the ‘A’ and ‘B’ inputs together results in a semantic pointer in ‘D’ that does not share a strong similarity to any of the vocabulary vectors. However, the added ‘Unbind’ network takes the generated semantic pointer in ‘D’ and convolves it with the inverse of ‘C’, which is the same as one of the inputs, resulting in an estimate of the other bound item.

- Alternate setting the value of ‘A’ to ‘RED’ and ‘BLUE’, the value of ‘B’ to ‘CIRCLE’ and ‘SQUARE’, and the value of ‘C’ to any one of these four labels. Run and pause the simulation between changes (this is not necessary if the simulation runs quickly).

The inputs to the ‘A’ and ‘B’ ensembles can be thought of as forming the statement ‘D’. The statements in this case could be interpreted as temporary associations, such as “squares are blue” and “circles are red”. The input to the ‘C’ ensemble then poses a question about the statement. Thus an input of ‘SQUARE’ asks “what is associated with squares” and the expected answer is “blue”.

This network is not a very impressive demonstration of cognitive ability since answers to questions are always provided at the same time as the questions themselves. However, the network is good start because it forms a bound representation from its input and then successfully extracts the originally bound elements.

The usefulness of this operation becomes apparent when memory is introduced to the network, so we will now add a memory buffer.

- Remove the projection from ‘D’ to ‘Unbind’ by clicking the ‘B’ termination on ‘Unbind’ and dragging it off the ‘Unbind’ network.
- Drag the ‘Integrator’ template from the template bar into the ‘Question Answering’ network.
- In the template dialog, set *Name* to ‘Memory’, *Number of neurons* to 400, *Number of dimensions* to 20, *Feedback time constant* to 0.4, *Input time constant* to 0.01, and *Scaling factor* to 1.0.
- Connect the projection from ‘D’ to the ‘input’ termination on ‘Memory’.
- Project from the ‘X’ origin of ‘Memory’ to the ‘B’ termination on ‘Unbind’.

The integrator network is quite simple, but behaviorally very important. This is because it can act to prolong the availability of information in the system, and hence act as a memory. With no input, the present state of the ensemble is projected back to itself, allowing it to retain that state. If input is present, it moves the system to a new state until the input ceases, at which point the new state is retained. So, without input the integrator acts as a memory, and with input it acts to update its current state based on recent history. In either case, it helps to track changes in the environment. Crucially, the decay of the memory is much slower than (though related to) the decay of the feedback.

In previous tutorials, we have included default time constants for all synapses (in ‘tauPSC’), but the models have been largely feedforward. As a result, the dynamics have been relatively straightforward. However, the integrator has a feedback connection, and so dynamics become more subtle (see the discussion of the third NEF principle in section 2.3.3). In this model we have set the feedback time constant to be artificially high (400ms), which gives a relatively stable memory (several seconds). However, a more physiologically realistic time constant of 100ms (a time constant associated with NMDA receptors) would require us to increase the number of neurons for the same memory performance, and would significantly slow the simulation. So, I consider this setting to only be a useful simplification for explanatory purposes.

Before proceeding, I will note that it is possible that the following simulation gets some of the expected answers wrong. This is because I have attempted to use as few neurons as possible to allow the network to run more quickly. Because the vocabulary and neuron tunings are generated randomly, some networks may not work perfectly. Increasing the number of neurons and the number of dimensions can alleviate any problems, but will affect the performance of the simulation.

- Open *Interactive Plots* and show the same graphs as before if they are not visible by default. (To see the ‘Memory’ network, you have to close and reopen the *Interactive Plots*, but be sure to save the layout before doing so.)
- Right-click the semantic pointer graph for ‘E’ and choose ‘select all’.
- Set ‘A’ to ‘RED’, ‘B’ to ‘CIRCLE’, and ‘C’ to ‘RED’.
- Run the simulation for 0.5 seconds.

This results in the same behavior as before, where the result in ‘E’ is ‘CIRCLE’.

- With the simulation paused, set ‘A’ to ‘BLUE’ and ‘B’ to ‘SQUARE’.
- Run the simulation for another half second and pause.

During the first second of the simulation, the network receives inputs from the ‘A’ and ‘B’ ensembles and binds them (i.e., resulting in $RED \circledast CIRCLE$). This is stored in the ‘Memory’ ensemble. During the next second, a second binding (i.e., $BLUE \circledast SQUARE$) is presented. It too is added to the ‘Memory’ so the value stored in the integrator memory after two seconds is approximately $RED \circledast CIRCLE + BLUE \circledast SQUARE$. As a result, the output ‘E’ remains as ‘CIRCLE’ because the ‘C’ input has stayed constantly equal to ‘RED’ and so unbinding will still result in ‘CIRCLE’, which is bound to ‘RED’ in the memory.

- Right-click the semantic pointer graphs for ‘A’ and ‘B’ and select ‘release value’. Run the simulation for 0.5 seconds.

You will notice that the result in ‘E’ is still ‘CIRCLE’ because the memory has not changed, and it already encoded the inputs.

- Change ‘C’ to a different vocabulary vector (‘BLUE’, ‘CIRCLE’, ‘SQUARE’).
- Run the simulation for 0.5 seconds and pause.
- Repeat the previous steps for each vocabulary vector.

For each of these cases, the memory is encoding the past state of the world, and only the question is changing in ‘C’. The result in ‘E’ changes because it continues to show the result of unbinding the semantic pointer in memory with the question input. The network is now performing question answering, since it can remember pairs of items and successfully recall which items are associated with each other. So it has internally represented a simple structure, and can answer queries about that structure.

However, we have not yet accomplished our original goal, which is to have statements and questions supplied through a single ‘visual input’ channel and produce replies in a ‘motor output’. The final stage of building the question answering network introduces the control structures needed to shift between a state of accepting new information to store in memory and a state of replying to questions about those memories.

- Remove the ‘Question Answering’ network from the workspace.
- Create a new network named ‘Question Answering with Control’.
- Drag the ‘Network Array’ template from the template bar into the network. Set *Name* to ‘Visual’, *Neurons per dimension* to 30, *Number of dimensions* to 100, *Radius* to 1.0, *Intercept (low/high)* to -1 and 1 respectively, *Max rate (low/high)* to 100 and 300, *Encoding sign* to ‘Unconstrained’, and *Quick mode* to enabled.
- Repeat the previous step to create two additional network arrays named ‘Channel’ and ‘Motor’, with the same parameters as the ‘Visual’ network.
- Drag the ‘Integrator’ template into the main network. Set the name of the integrator to ‘Memory’ and give it 3000 neurons, 100 dimensions, a feedback time constant of 0.4, an input time constant of 0.01, and a scaling factor of 1.0.

The network array template creates a subnetwork of many low-dimensional ensembles that together represent a high-dimensional vector. Specifically, each internal ensemble represents only one dimension (the number of dimensions per ensemble can be changed through scripting). In general, neural ensembles seem to represent many dimensions, however, the network array can be constructed much more quickly for computational reasons.¹ Since we have increased our dimensionality to 100, performance will degrade significantly if we do not use network arrays (unless you have a fast computer). As well, we have employed them in a way that should affect the performance of the network. This increase in dimensionality is needed because it allows for more complex structures to be created while lowering the odds that semantic pointers will be similar through coincidence (see section 4.5).

The integrator template automatically makes use of network arrays to create integrators with 8 or more dimensions. This is both for computational and stability reasons. High-dimensional stable integrators are much more difficult to make than low-dimensional ones.

We can now begin to connect the circuit:

- Add a decoded termination to the ‘Channel’ network called ‘visual’. Set tauPSC at 0.02, *Input Dim* to 100 and use the default coupling matrix (the identity matrix).
- Create a projection from the ‘Visual’ network array to the ‘visual’ termination.
- Create a projection from the ‘Channel’ network to the ‘input’ termination on the ‘Memory’ network.
- Drag the ‘Binding’ template into the ‘Question Answering with Control’ network. Set *Name* to ‘Unbind’, *Name of output ensemble* to ‘Motor’, and *Number of neurons per dimension* to 30. Enable the *Invert input A* option.
- Project from the origin of the ‘Memory’ network to the ‘B’ termination of the ‘Unbind’ network.

¹With a network array, a low-dimensional ensemble can be replicated such that neuron tuning curves and decoding weights only need to be computed for a single dimension of the network. Disabling ‘quick mode’ will create random tuning curves for all the neurons in the network. This is extremely time-consuming, so it is highly recommended that quick mode be enabled.

- Project from the origin of the ‘Visual’ network to the ‘A’ termination of the ‘Unbind’ network.

At this stage, we have roughly recreated the basic ‘Question Answering’ network, albeit with much larger populations representing a higher-dimensional space. The remaining elements of the network will introduce action selection and information routing.

- Drag the ‘Basal Ganglia’ template into the network. Set the name to ‘Basal Ganglia’, the number of actions to 2, and the time constant to 0.01.

This created a subnetwork which is structured like the basal ganglia model described earlier (section 5.2). To see the elements of this model, double-click the subnetwork.

- Drag the ‘BG Rule’ template onto the new ‘Basal Ganglia’ network. Set *Rule Index* to 0, *Semantic Pointer* to ‘STATEMENT’, *Dimensionality* to 100, *tauPSC* to 0.01, and enable *Use Single Input*.

Choosing *Use Single Input* sends any input to basal ganglia to all three input elements of the basal ganglia (i.e., striatal D1 and D2 receptors, and STN).

- Drag another ‘BG Rule’ template onto the basal ganglia network. Set *Rule Index* to 1, *Semantic Pointer* to ‘QUESTION’, *Dimensionality* to 100, *tauPSC* to 0.01, and enable *Use Single Input*.
- Project from the ‘Visual’ network to both the ‘rule_00’ and ‘rule_01’ terminations on the ‘Basal Ganglia’ network.

The basal ganglia component receives projections from ensembles representing semantic pointers, assesses the similarity of the input to the set of ‘rule antecedents’ which are also vocabulary vectors, and computes a winner-take-all-like function. In this model, the basal ganglia receives projections from the ‘Visual’ network, then outputs the vector (0, 1) if the input is similar to the semantic pointer designated ‘STATEMENT’ or the vector (1, 0) if the input is similar to the semantic pointer ‘QUESTION’ (recall that the output of basal ganglia inhibits the selected action).

- Drag a ‘Thalamus’ template into the main network. Set *Name* to ‘Thalamus’, *Neurons per dimension* to 30 and *Dimensions* to 2.

- Add a termination to the ‘Thalamus’ network. Name it ‘bg’, set *Input Dim* to 2, and *tauPSC* to 0.01. Click *Set Weights* and set the diagonal values of the matrix (the two that are 1.0 by default) to -3.0.
- Create a projection from the output origin of ‘Basal Ganglia’ to the ‘bg’ termination of ‘Thalamus’.

The large negative weights between basal ganglia and thalamus reflect the inhibitory input of basal ganglia. Thus, when an action is selected, its activity goes to zero, which will be the only non-inhibited channel into the thalamus, releasing the associated action consequent. We can now define action consequents.

- Drag a ‘Gate’ template into the main network. Name the gate ‘Gate1’, set *Name of gated ensemble* to ‘Channel’, *Number of neurons* to 100, and *tauPSC* to 0.01.
- Add a decoded termination to the ‘Gate1’ ensemble. Set *Name* to ‘thalamus’, *Input Dim* to 2 and click *Set Weights*. Set the weights to [1 0].
- Create a projection from the ‘xBiased’ origin on the thalamus network to the termination on ‘Gate1’.

In this model, each consequent of the thalamus results in ungating information flow in the ‘cortex’. Essentially, the non-selected outputs of the basal ganglia inhibit the thalamus, whereas the selected outputs stop inhibiting connected parts of thalamus, allowing those parts to fire. This thalamic firing in turn inhibits a cortical gate, which then stops preventing a flow of information into the network it controls. So far, we have set up the circuit that will allow ‘Visual’ information to flow into the ‘Channel’ when ‘STATEMENT’ is presented to the basal ganglia.

- Drag a second ‘Gate’ template into the main network. Name the gate ‘Gate2’, set *Name of gated ensemble* to ‘Motor’, *Number of neurons* to 100, and *tauPSC* to 0.01.
- Add a decoded termination to the ‘Gate2’ ensemble. Set *Name* to ‘thalamus’, *Input Dim* to 2 and click *Set Weights*. Set the weights to [0 1].
- Create a projection from the ‘xBiased’ origin on the thalamus network to the termination on ‘Gate2’.

We have now created a second action consequent, which allows information to flow to the ‘Motor’ cortex from the ‘Unbind’ network when ‘QUESTION’ is presented to basal ganglia. We can now run the model.

- Open *Interactive Plots* and display only the ‘Visual’ and ‘Motor’ nodes (hide other nodes by right-clicking on them).
- Turn on the semantic pointer graphs for the ‘Visual’ and ‘Motor’ ensembles by right-clicking these nodes and selecting ‘semantic pointer’.
- Set the value of the ‘Visual’ semantic pointer to ‘STATEMENT+RED*CIRCLE’ by right-clicking the semantic pointer graph.
- Run the simulation for 0.5 seconds of simulation time and pause.
- Set the value of the ‘Visual’ semantic pointer to ‘STATEMENT+BLUE*SQUARE’. Run for another half second and pause.
- Set the value of the ‘Visual’ semantic pointer to ‘QUESTION+BLUE’.
- Run the simulation until the semantic pointer graph from the ‘Motor’ population is settled (this should take under half a second).
- Set the value of the ‘Visual’ semantic pointer to ‘QUESTION+CIRCLE’. You can substitute ‘RED’, ‘SQUARE’, or ‘BLUE’ for ‘CIRCLE’.
- Run the simulation and repeat the previous step for each possible question.

If the question answering model has been properly constructed, it will successfully store bound pairs of vectors given inputs that have been summed with the semantic pointer designating statements. If you are unable to get the network to function, be sure to check all of the weight settings in particular. If all else fails, the ‘question-control.py’ script can be run from the *File→Open* menu.

There are several ways you might go beyond this tutorial. For example, you could increase the vocabulary of the network to determine how many items it can store in memory before being unable to successfully recall items. You could add additional rules to the basal ganglia to incorporate other actions. You could determine what happens when there are multiple answers to a question, and so on.

The completed question answering network contains about 20 000 neurons and demonstrates rudimentary examples of the main SPA elements (except learning). These same elements are used to construct the advanced models in this book (e.g., Tower of Hanoi, Raven’s Matrices, etc.).

- ???make sure semantic pointer graphs are automatically updated...

Chapter 6

Biological cognition – memory and learning

6.1 Extending cognition through time

I consider memory and learning together because both allow biological systems to extend their cognition through time – both, that is, allow neural systems to adapt to the intricacies of their environment based on experience. The memory of recent or long past events can be very important for determining an appropriate behaviour in the present. Learning is often thought of as the fine-tuning of behaviour based on feedback, again resulting in current behaviour being sensitive to events that occurred in the past. Learning can generally be characterized as the accumulation of memories.

Unlike in a traditional computer architecture, memory and learning are tightly coupled to computation in biological brains. For many, this marks one of the most fundamental differences between neural computation and computation in a standard digital computer. Consequently, any neural architecture would be incomplete without a consideration of the role of memory and learning.

Conceptually speaking, different kinds of learning and memory are difficult to cleanly map to specific neural mechanisms. Traditionally, in the neural network literature, when researchers speak of “learning”, they are referring to changing the connection weights between neurons in the network. The ubiquitous “learning rules” proposed by these researchers are a means of specifying how to change connection weights given the activity of neurons in the network. However, notice that the example of learning that we have seen earlier in the book (section 4.6)

looks like a standard case of learning, but depends only on an active memory system. That is, no connection weights are changed in the network even though the system learns a general rule that allows past experience to change what behaviors are chosen in the present.

Similarly, the traditional contrast between short-term (or “working”) memory¹ (which is often taken to be activity in a recurrent network), and long-term memory (which is often taken to be postsynaptic connection weight changes), is difficult to sustain in the face of neural evidence. For instance, there are a wide range of time scales of adaptive processes influencing neural firing in cortex (Ulanovsky et al., 2004). Some short-term (seconds to minutes) changes are related to postsynaptic weight changes (Whitlock et al., 2006), and others are not (Varela et al., 1997; Romo et al., 1999b). Consequently, some “short-term” memories might be stored in the system in virtue of connection weight changes, and others might be stored in virtue of pre-synaptic processes, or stable network dynamics. The time scales we use to identify kinds of memory behaviorally seem pick out more than one neural mechanism.

Because of such complexities in the mapping between kinds of memory and neural mechanisms, I am not going to provide a general description of memory and learning. Instead, I provide specific examples of how activity-based and connection weight-based mechanisms for adaptation can be employed by the Semantic Pointer Architecture. This provides for an admittedly modest selection of adaptive mechanisms at play in the brain, but it gives an indication of how these two common forms of adaptation integrate naturally with other elements of the architecture.

More specifically, this chapter addresses two challenges. The first is related to activity-based explanations of working memory, and the second is related to learning connection weight changes. The first challenge is to extend current neural models of working memory to make them more cognitively relevant. There are many neurally plausible models of working memory that rely on recurrently connected networks (Amit, 1989; Zipser et al., 1993; Eliasmith and Anderson, 2001; Koulakov et al., 2002; Miller et al., 2003). These models are able to explain the sustained firing rate found in many parts of cortex during the delay period

¹Sometimes these terms are distinguished and sometimes they are not. Additional distinctions do not challenge the point being made here, which is a consequence of the fact that kinds of memory are typically behaviourally defined, and the timescales of different neural mechanisms (some activity-based, some connection-based) overlap. Consequently, there are medium-time-scale behaviors that are activity-based, and others that are connection-based (and no doubt others that are a combination of the two).

of memory experiments. Some are able to explain more subtle dynamics, such as ramping up and down of single cell activity seen during these periods (Singh and Eliasmith, 2006). However, none of these models address complex working memory tasks that are essential for cognition, such as serial working memory (i.e. remembering an ordered list of complex items). This is largely because methods for controlling the loading of memories, clearing the network of past memories, and constructing sophisticated representations, have not been devised. In the next two sections, I describe a model that addresses many of these limitations.

The second challenge is to introduce a biologically realistic mechanism for connection weight changes that can learn manipulations of complex representations. In the last two sections of this chapter, I describe a spike-timing-dependent plasticity (STDP) rule that is able to learn linear and non-linear transformations of the representations used in the SPA. I demonstrate the rule by showing that it can be used to learn the binding operation employed in the SPA (i.e., circular convolution), that it can be used to learn action selection in the model of the basal ganglia (section 5.2), and that it can be used to learn different reasoning strategies in different contexts to solve a language processing task. The tutorial at the end of this chapter demonstrates how to use this rule in Nengo.

6.2 Working memory

Working memory is typically characterized as an actively engaged system used to store information that is relevant to the current behavioral situation. Typical tests of working memory in monkeys consist of having a monkey fixate at a central location, and presenting a target stimulus somewhere in the periphery of the monkey’s visual field. The stimulus is removed, a delay of about 3 to 6 seconds long is then initiated, and finally the fixation target changes to indicate to the monkey that it should respond by saccading or pointing to the remembered location of the stimulus. Many experiments like these have been used to characterize the activity of single cells during working memory tasks (e.g., Romo et al., 1999a). And, many models have been built that explain a wide variety of the properties of single cell tuning curves observed during such tasks (Zipser et al., 1993; Amit et al., 1997; Koulakov et al., 2002; Miller et al., 2003; Singh and Eliasmith, 2006; Machens et al., 2010).

However, it is not uncommon for much greater demands to be put on working memory. Rather than encoding and recalling a single item, many situations require encoding the order in which several items occur (e.g., a telephone number). As

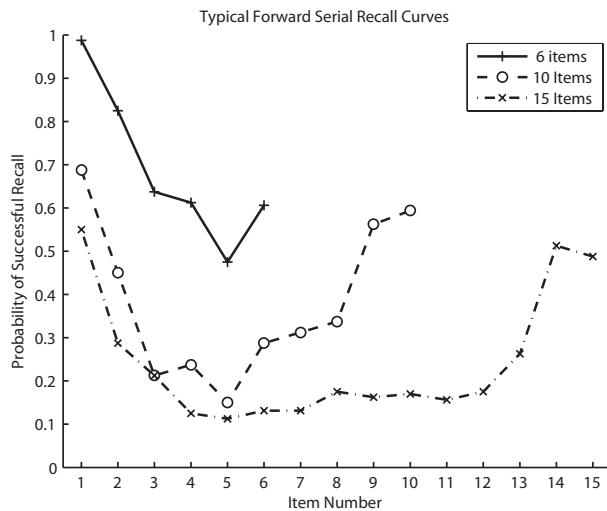


Figure 6.1: Recall accuracy from human behavioural studies showing primacy and recency effects (data from Jahnke (1968)). Participants were asked to recall ordered lists of varying length.

a result, cognitive psychologists have studied these more demanding tasks since the 1960s (Jahnke, 1968; Rundus, 1971; Baddeley, 1998). The ability to store and recall items in *order* is called serial working memory. Unlike the simple single target case, serial working memory is seldom studied in animal single cell studies (Warden and Miller, 2007), and there are no spiking single cell models that I am aware of (although there are some neurally inspired models, e.g., Beiser and Houk (1998); Botvinick and Plaut (2006)). It is thus both crucial to have serial working memory in a cognitive architecture, and unclear how such a function can be implemented in a biologically realistic network.

In studying serial working memory, two fundamental regularities have been observed: primacy, and recency. Both primacy and recency can be understood by looking at an example of the results of a serial recall task, like those shown in figure 6.1. In this task, subjects are shown lists of items of various lengths, and asked to recall the items in their original order. Primacy is identified by the observation that items appearing earlier in the list have a greater chance of being recalled accurately, regardless of the length of the list. Recency is identified by the observation that the items most recently presented to subjects have an increased chance of being recalled as well. Together, primacy and recency account for the typical U-shaped response probability curve seen in serial working memory tasks.

Interestingly, this same U-shape is seen in free recall tasks (where order information is irrelevant; see figure 6.6). So it seems likely that the same mechanisms are involved in both free recall and serial recall. In fact, as I discuss in more detail in the next section, it seems likely that all working memory behavior can be accounted for by a system that has serial working memory. In contrast, models of working memory that can account for the behavior of monkeys on single item tasks cannot account for serial working memory behavior. Consequently, serial working memory is of more fundamental importance when considering human cognition.

Recently, Xuan (pronounced “Sean”) Choo in my lab has implemented a spiking neural network model of serial working memory using the resources of the SPA. Based on a consideration of several models from mathematical and cognitive psychology (Liepa, 1977; Murdock, 1983, 1993; Henson, 1998; Page and Norris, 1998), Xuan has proposed the ordinal serial encoding (OSE) model of working memory (Choo, 2010). In the next section, I present several examples of how this model can account for human psychological data on serial and free recall tasks. First however, let us consider the component parts, and the basic functions of the model.

The encoding and decoding of items to be remembered by the OSE model is depicted in figure 6.2. As can be seen there, the model consists of two main components, an input working memory and an episodic memory. These are taken to map onto cortical and hippocampal memory systems respectively. Both memories are currently modeled as neural integrators (see section 2.3.3). A more sophisticated hippocampal model (e.g., Becker, 2005) would improve the plausibility of the system, although it would presumably not significantly change the resulting performance. Notably, modeling these memory systems as neural integrators amounts to including past models of single item working memory as a component of this model (such as the model proposed in Singh and Eliasmith (2006)).

The input to the model consists of a semantic pointer representing the item, and another semantic pointer representing its position in a list. These two representations are bound, using a convolution network (section 4.2), and fed into the two memory systems. Simultaneously, the item vector alone is fed into the two memories to help enforce the item’s semantics for free recall. Each memory system adds the new representation to the list that is currently in memory, and the overall representation of the sequence is the sum of the output of the two memories.

Decoding of such a memory trace consists of subtracting already recalled items from the memory trace, convolving the memory trace with the inverse of

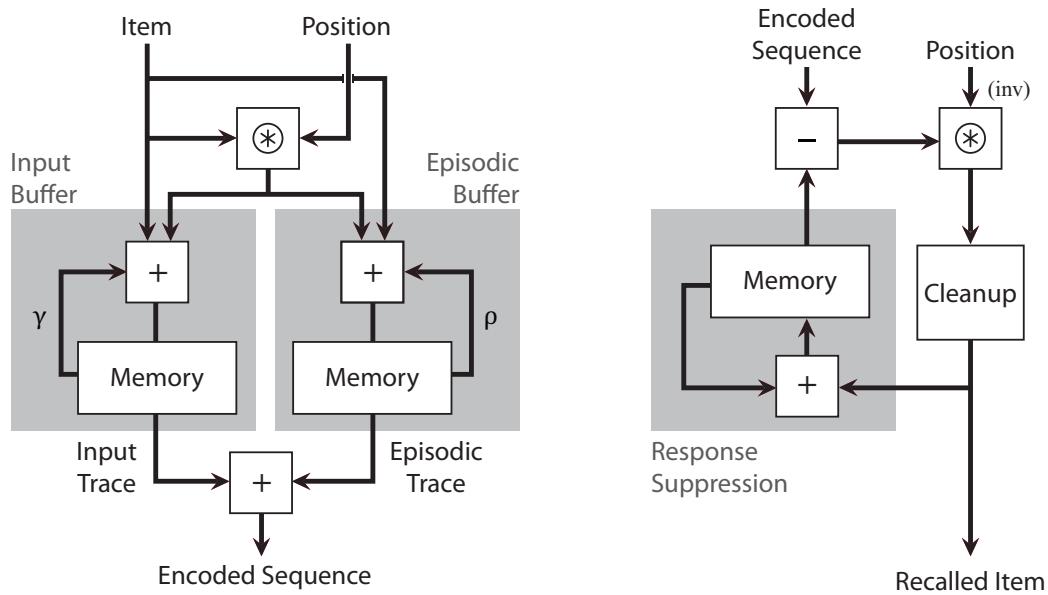


Figure 6.2: Network level diagrams of the OSE encoding network (left) and the OSE item recall network (right). The encoding network receives an item and position semantic pointer as input, which are bound and sent to both memory systems. This is added to the current memory state. Each memory stores the information by integrating it over time, although the decay rates (γ and ρ) are different. The decoding network begins with the originally encoded memory and decodes it by unbinding the appropriate position. The result is cleaned up and placed in a memory that tracks already recalled items. The contents of that memory is subtracted from the originally encoded sequence before additional items are recalled.

the position vector, and passing the result through a clean-up memory (section 4.5). In short, the decoding consists of unbinding the desired position from the total memory trace, and cleaning up the results. Concurrently, items which have already been generated are subtracted so as to not be generated again. In the case of free recall (where position information is not considered relevant), the unbinding step is simply skipped (this can be done by setting the position vector to be an identity vector). The equations describing both the encoding and decoding processes can be found in appendix B.3.

As can be seen from figure 6.2, this model has two free parameters, ρ and γ . These govern the dynamic properties of the two different memory systems captured by the model. Specifically, ρ captures the effect of rehearsing items early in the list during the encoding process, and is associated with hippocampal function. The value of this parameter was set by fitting an experiment conducted by Rundus (1971) in which the average number of rehearsals was found as a function of the item number in a serial list tasks. The other system, influenced by γ , characterizes a generic cortical working memory process, and experiences a fading of its memorized representation over time. The value of this parameter was set by reproducing the simple working memory experiment described in Reitman (1974) and choosing the value that allowed the accuracy of the model matches that of humans. Reitman's experiment showed that human subjects could recall on average 65% of the items that they recalled immediately after list presentation after a 15 second delay (when rehearsal was not permitted).

It is important to note that the two free parameters of this model were set by considering experiments which are not being used to suggest that the model is a good model of serial recall. That is, they were set independently of the test experiments considered in the next section. Consequently, the performance of the model on these new tasks in no way depends on tuning parameters to those tasks.

Finally, it is worth highlighting that the preceding description of this model is reasonably compact. This is because we have encountered many of the underlying elements – the necessary representations, the basic transformations, and the dynamical circuits – before in the Semantic Pointer Architecture. So, in many ways this working memory system is not a new addition to the SPA, but rather a recombination of functions already identified as central to the architecture (e.g., integration, binding, and clean-up). This is encouraging because it means our basic architecture does not need to get more complicated in order to explain additional, even reasonably sophisticated, functions.

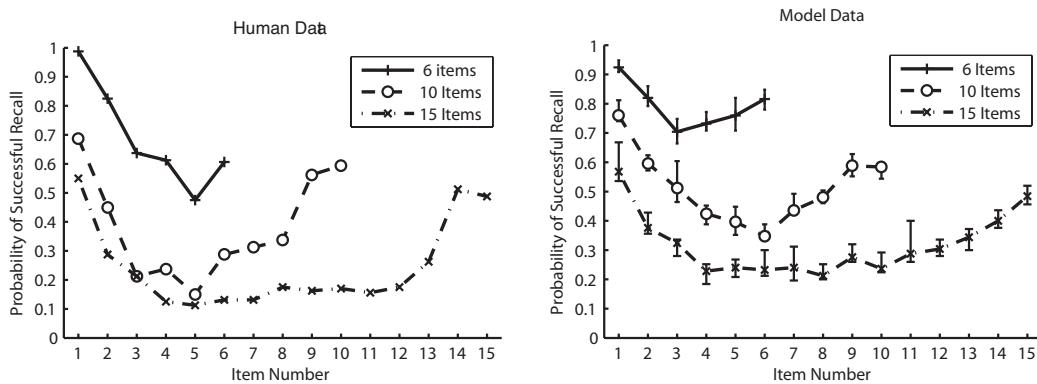


Figure 6.3: A comparison of human and model data on serial list recall tasks. The model data is presented with 95% confidence intervals while for human data only averages were reported. The human data is from Jahnke (1968).

6.3 Example: Serial list memory

Figure 6.3 shows the OSE model capturing the basic recency and primacy effects observed in the human data, shown earlier in figure 6.1. This demonstrates not only that the model captures basic features of serial memory, but that setting the parameters of the model using different experiments did not limit the system’s ability to explain this behavior.

As is clear from the fact that the model (and the subjects) do not have perfect recall accuracy, mistakes are made in recalling serial lists. These mistakes have been analyzed in several experiments (Henson et al., 1996). Interestingly, these same experiments were taken to show that recurrent network models would not be able to reproduce the human data (Botvinick and Plaut, 2006). However, as made clear in Botvinick and Plaut (2006), this is not the case. The model presented here supports that conclusion, and extends it to show that recurrent models can address more dynamical aspects of serial memory, such as delayed recall, and a wide variety of recall demands, such as reverse and free recall, not considered previously. Returning to the empirical work that characterizes error, figure 6.4a shows a comparison of the transposition gradients found in the model with those measured from human subjects. The transposition gradient measures the probability of recalling an item outside of its correct position in a list. Both the model and the human data show that errors are most likely to occur in positions near the original item position, as might be expected.

As you might also expect, the similarity between items in a list can affect

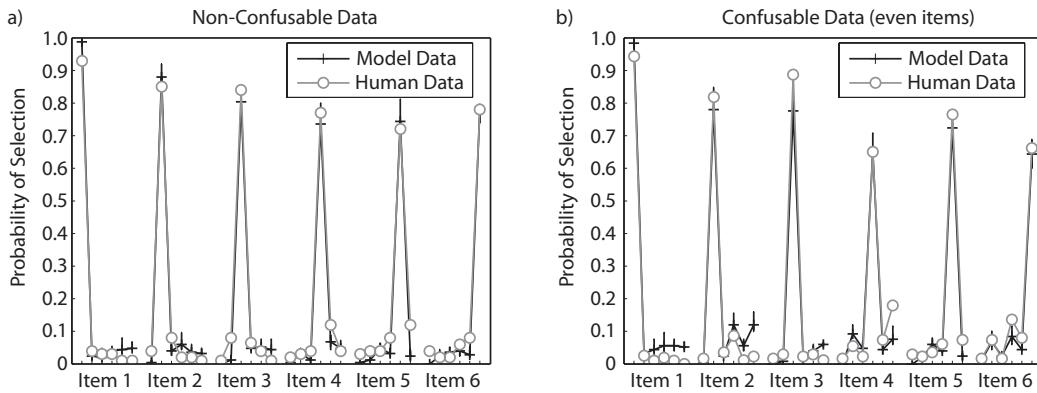


Figure 6.4: Transposition gradients for serial recall tasks. Given the task of recalling six items in order, these plots show the probability of selecting each item for each of the six possible positions in the list. The transposition gradients thus show the probabilities of errors involving recalling an item outside its proper position. a) The six items in the list are all easy to distinguish. b) Even numbered items in the list are similar and more easily confused. Human data is from Henson et al. (1996).

how well items can be recalled. Henson et al. (1996) also designed an experiment in which they presented subjects with lists containing confusable and non-confusable letters. Because the stimuli were heard, confusable letters were those which rhymed (e.g. “B”, “D”, and “G”), while non-confusable letters did not (e.g., “H”, “K”, and “M”). The experimenters presented four different kinds of lists to the subjects: lists containing all confusable items, those containing confusable items at odd positions, those containing confusable items at even positions, and those containing no confusable items. The probability of successful recall for these lists for both the human subjects and the model are shown in figure 6.5; a comparison of the transposition gradients from the model and human subjects on a recall task with confusable items at even positions is given in figure 6.4b..

Again, this example shows that the model does a good job of capturing behavioral data, both the likelihood of successful recall and the pattern of errors that are observed in humans. The same model has also been tested on several other serial list tasks, including delayed recall tasks, backwards recall tasks, and combinations of these (Choo, 2010), although I do not consider these here

More important for present purposes, the same model can also explain the results of free recall experiments. As shown in figure 6.6, the accuracy of recall

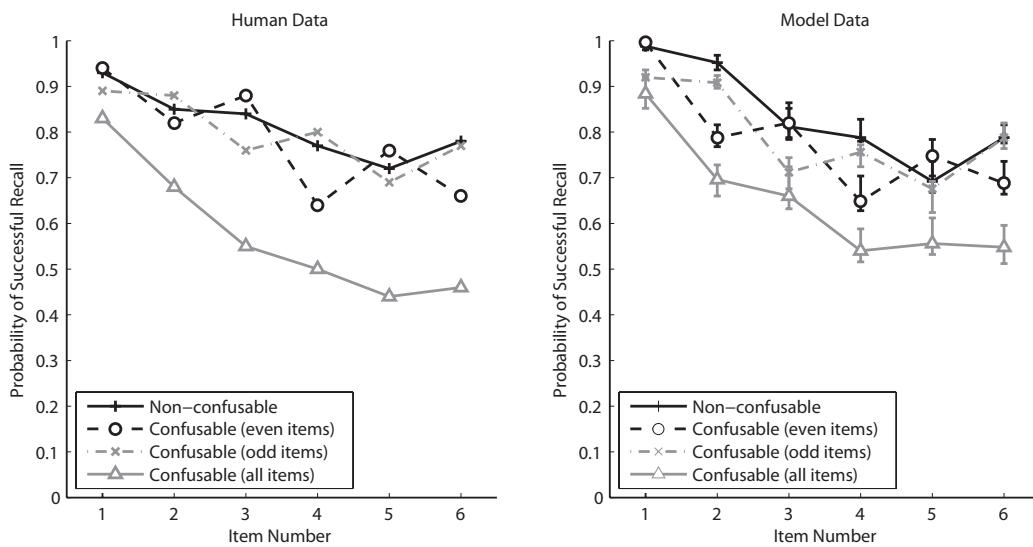


Figure 6.5: Serial recall of confusable lists. These graphs plot human and model performance recalling lists containing four different patterns of easily confusable versus non-confusable items. Model data includes 95% confidence intervals, while for human data only averages were reported. The human data is from Henson et al. (1996), experiment 1.

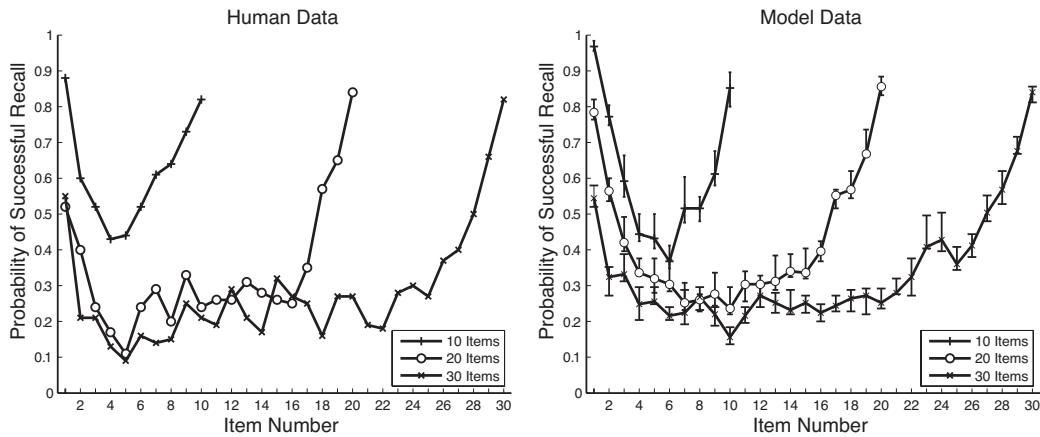


Figure 6.6: Human and model data for free recall tasks. Unlike the serial recall task of figure 6.3, subjects can repeat the contents of the lists in any order. The free recall results continue to demonstrate primacy and recency effects with U-shaped curves. Human data is from Postman and Phillips (1965).

for the model is very similar to that of humans for a wide variety of list lengths. In these tasks, there is no constraint on the order in which items are recalled from memory. Nevertheless, both the model and human data show the typical U-shaped response probability curves.

Taken together, these examples of the OSE model performance on a variety of tasks (none of which were used to tune the model) can make us reasonably confident that some of the principles behind human working memory are captured by the model. Because it uses the same kinds of representations as the rest of the SPA, we can be confident that it will integrate easily into larger scale models. I take advantage of this in the next chapter.

However, before leaving consideration of this model I want to highlight what I think is perhaps its most theoretically interesting feature: namely, that this model only works if it is implemented in neurons. As demonstrated by figure 6.7a, if we directly simulate the equations that describe this model (see appendix B.3), it is unable to accurately reproduce the recency and primacy effects observed in the human data. Initially, it seemed that this failure might have been caused by the semantic pointer vectors in the system becoming arbitrarily long as additional items were added to the memory trace. Consequently, we also implemented the model using standard vector normalization, which guarantees that the vectors always have a constant length. But again, as seen in figure 6.7b, the model is unable

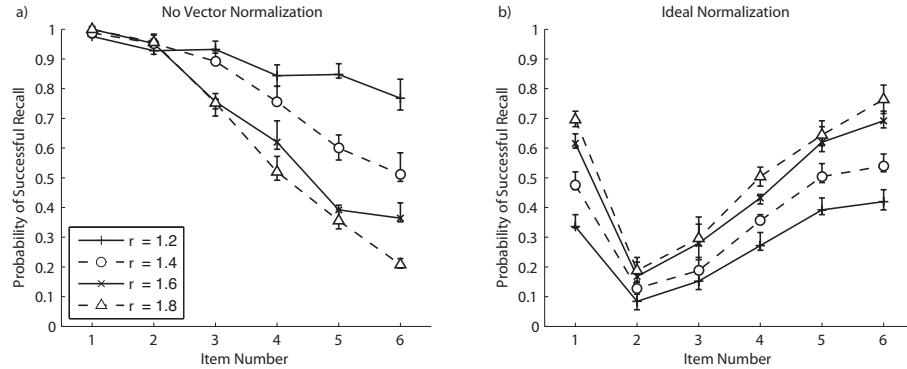


Figure 6.7: Non-neural implementations of the recall model. a) The recall model is implemented by directly evaluating the equations in appendix B.3. Vectors are not normalized and can grow arbitrarily long. b) The recall model is again implemented without spiking neurons, but vector lengths are held constant using ideal vector normalization. Varying the parameters does not help. The effects of changing ρ are shown.

to capture the human data (i.e. neither case has the appropriate U-shaped curve, although normalization is closer).

Consequently, we realized that one of the main reasons that this model is able to capture the human data as it does, is that the individual neurons themselves saturate when participating in the representation of large vectors. This saturation serves as a kind of “soft” normalization, which is neither ideal mathematical normalization, nor a complete lack of normalization. Instead, it is a much more subtle kind of constraint placed on the representation of vectors in virtue of neuron response properties. And, crucially, this constraint is directly evident in the behavioral data (i.e., it enables reconstructing the correct U-shaped curve). This is theoretically interesting, because it provides an unambiguous example of the importance of constructing a neural implementation for explaining high-level psychological behavior. All too often, researchers consider psychological and neural level explanations to be independent: a view famously and vigorously championed by Jerry Fodor (Fodor, 1974). But in this case, the dependence is clear. Without constructing the neural model, we would have considered the mathematical characterization a failure, and moved on to other, likely more complex, models. However, it is now obvious that we would have been doing so unnecessarily. And

unnecessary complexity is the bane of intelligible explanations.²

6.4 Biological learning

I mentioned earlier that the NEF methods are useful partly because they do not demand that learning be central to designing models (section 2.3.4). But, of course, this does not mean that learning is not an important aspect of our cognitive theories – I have emphasized this point too (section 1.4). So, in this section I consider not only how learning can be included in SPA-based models, but how our understanding of learning can be enhanced by adopting a method that allows for the direct construction of some networks. As well, I have been at pains to argue for the biological plausibility of the methods I adopt throughout the book, and will continue this trend in my consideration of learning.

It is difficult to specify what counts as a “biologically plausible” learning rule, as there are many mechanisms of synaptic modification in the brain (Feldman, 2009; Caporale and Dan, 2008). However, the vast majority of characterizations of synaptic plasticity that claim to be biologically plausible still adhere to Donald Hebb’s (1949) suggestion that: “When an axon of cell A is near enough to excite cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A’s efficiency, as one of the cells firing B, is increased” (p. 62). Or, more pithily: “Neurons that fire together wire together.” Most importantly, this means that any modification of synaptic connection weights must be based on information directly available to the cell whose weight is changed (i.e., it must be based on pre-synaptic and postsynaptic activity alone). Unfortunately, most learning rules stop here. That is, they introduce variables that stand-in for pre- and postsynaptic activity (usually these are “firing rates”), but they typically do not work with actual neural activity (i.e. temporal spiking patterns).

This is problematic from a plausibility perspective because there is increasingly strong evidence that the modification of synaptic strength can be highly dependent on the *timing* of the pre- and postsynaptic spikes (Markram et al., 1997; Bi and Poo, 1998). But, rules that employ firing rates do not capture relative spike timing. Such spike-driven learning has become known as STDP (spike-timing dependent plasticity; figure 6.8). STDP refers to the observation that, under certain circumstances, if a presynaptic spike occurs before a postsynaptic spike, there is

²Another example of the importance of building neural implementations to match psychological data can be found in Hurwitz (2010).

an increase in the likelihood of the next presynaptic spike causing a postsynaptic spike. However, if the postsynaptic spike precedes the presynaptic spike, then there is a decrease in the likelihood of the next presynaptic spike causing a postsynaptic spike. Or, more succinctly, a presynaptic spike followed closely in time by a postsynaptic spike will potentiate a synapse, and the reverse timing depresses it – ultimately, this can be thought of as a more precise statement of “fire together, wire together”.

The learning rules proposed to explain STDP computationally, focus, as the original experiments did, on comparing two spikes at a time. However, in order to explain more recent plasticity experiments, it has become necessary to consider triplets of spikes (Pfister and Gerstner, 2006). Specifically, Pfister and Gerstner show that by adding an additional depression with a pre-post-pre triplet, and additional potentiation with post-pre-post triplets, a better fit to the experimental data is achieved. In particular, triplets allow the rule to capture frequency as well as timing effects.

Interestingly, there is also recent evidence that it is not specifically post-synaptic *spikes* themselves that drive this plasticity. Instead, other indicators of post-synaptic activity like synaptic potentials and dendritic calcium flux are much more important for generating synaptic plasticity (Hardie and Spruston, 2009). Consequently, non-spiking activity of the cell may be a more appropriate measure of post-synaptic activity for learning rules.

In addition, there has long been evidence that there are homeostatic mechanisms in individual neurons that are important for learning (Turrigiano and Nelson, 2004). Intuitively, these mechanisms ensure that neurons do not increase their weights indefinitely: if two neurons firing together causes them to wire more strongly, there is no reason for them to stop increasing their connection strength. Homeostatic mechanisms have been suggested that ensure that this kind of synaptic saturation does not occur. Some have suggested that the total connection strength a neuron can have over all synapses is a constant. Others have suggested that neurons monitor their average firing rate and change connection strengths to ensure that this rate stays constant. Still others have suggested that a kind of ‘plasticity threshold’, which changes depending on recent neuron activity and determines whether weights are positively or negatively increased for a given activity level (Bienenstock et al., 1982). It is still unclear what the sub-cellular details of these mechanisms might be, but their effects have been well-established experimentally.

Trevor Bekolay, a member of my lab, has recently devised a learning rule that incorporates many of these past insights, and is able to both reproduce the

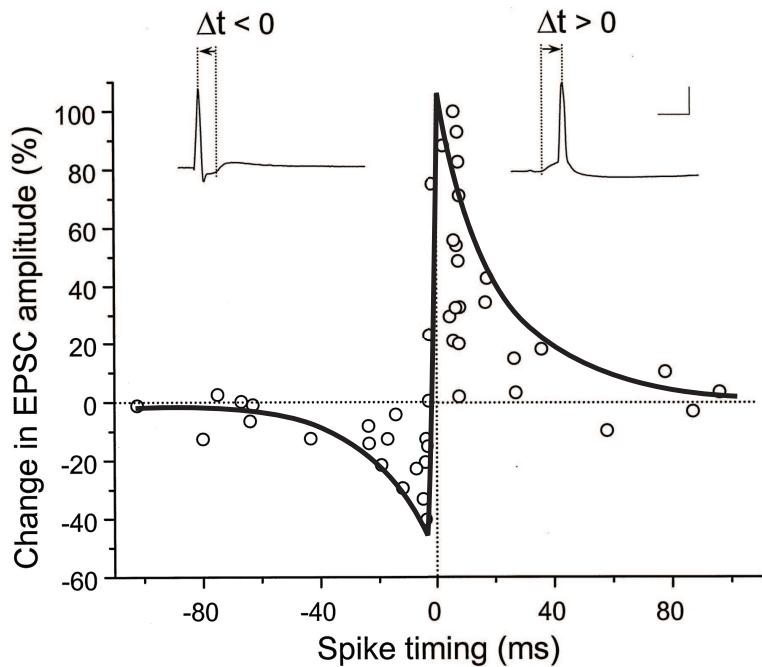


Figure 6.8: Spike-timing dependent plasticity (STDP). This figure shows the critical window of spike-timing for affecting synaptic weight change. The percentage change in the postsynaptic current (PSC) amplitude at 20–30 min after the repetitive spiking (60 pulses at 1 Hz) is plotted against the spike timing. The small inset images above indicate that when the spikes generated by a neuron regularly came before PSCs induced in it, the amplitude of subsequent PSCs at that synapse decreased, and when the spikes came after the induced PSCs, the amplitude increased. The thick line indicates a commonly used “adaptation function” that summarizes this effect. (Adapted from Bi and Poo (1998) with permission).

Figure 6.9: Reproduction of STDP timing and frequency results with Trevor’s rule.

relevant physiological experiments and learn high-dimensional vector transformations. The mathematical details of this rule can be found in appendix B.4. This rule is based on two past rules, the well-known BCM rule that characterizes a homeostatic mechanism (Bienenstock et al., 1982), and recently proposed rule from my lab that learns using post-synaptic activity in combination with error information (ref macneil???)

Trevor’s new rule simultaneously addresses limitations of both standard Hebbian learning rules and STDP. In particular, unlike most standard rules this rule is able to account for precise spike time data, and unlike most STDP (and standard) rules it is able to relate synaptic plasticity directly to the vector space represented by an ensemble of neurons. That is, the rule can tune connection weights that compute nonlinear functions of whatever high-dimensional vector is represented by the spiking patterns in the neurons. Unlike past proposals, Trevor’s rule is able to do this because it relies on an understanding of the NEF decomposition of connection weights. This decomposition makes it evident how to take advantage of high-dimensional error signals, which past rules have either been unable to incorporate or attempted to side-step (Gershman et al., 2010). Thus, Trevor’s rule can be usefully employed in a biologically plausible network that can be characterized as representing and transforming a vector in spiking neurons – precisely the kinds of networks we find in the SPA (section 6.7 describes a tutorial using this rule).

To demonstrate the biological plausibility of this rule, figure 6.9 shows that the rule reproduces both the timing and frequency results determined in STDP experiments. This makes it clear that the rule is able to function appropriately in a spiking network, and has a sensitivity to spike timing that mimicks our best current characterization of learning *in vivo*. More than this, however, we would like to demonstrate that the rule can learn known functions.

Like many other rules, this rule can incorporate error information coming from other parts of the system, to affect the weight changes in a synapse. There is good evidence for this kind of modulatory learning in several brain areas. For instance, the stability of the oculomotor integrator is dependent on retinal slip information, but only after it has been processed by other parts of brain stem (Askay et al., 2000; Ikezu and Gendelman, 2008, , ch. 5)). More famously, modulatory learning is also subserved by the neurotransmitter dopamine in cortex and basal ganglia. Specifically, there is strong evidence that some dopamine neurons increase their

firing rate both when rewards that are not predicted occur, and when predicted rewards do not occur (Hollerman and Schultz, 1998). Consequently, it has been suggested that dopamine can act as a modulatory input to cortex and parts of the basal ganglia (e.g., striatum), helping to determine when connections should be changed in order to account for unexpected information in the environment. It is now well-established that this kind of learning has a central role to play in how biological systems learn to deal with contingencies in their world (Maia, 2009). Unsurprisingly, such learning is important in SPA models that seek to explain such behavior (see section 6.5).

So, while the source of the error signal may be quite varied, it is clear that biological learning processes can take advantage of such information. To demonstrate that Trevor’s rule is able to do so, figure 6.10 shows a simple example of applying this rule to a scalar representation, with feedback error information. The left-hand side of the figure shows the structure of a simple circuit that generates an error signal, and the right hand side shows an example run during which the improvement in the receiving population’s ability to represent the input signal is evident. To keep things simple, we have included the generation of the error signal in the network. However, the error signal could come from any source, internal or external to the network.

This example demonstrates that the rule can learn a simple communication channel, but of course much more sophisticated nonlinear functions are important for biological cognition. A particularly important and sophisticated transformation in the SPA is circular convolution: recall that it is used for binding, unbinding, and content sensitive control. Given the central role of convolution in the architecture, it is natural to be concerned that it might be a highly specialized, hand-picked, and possibly unlearnable transformation. If this was the case, that would make the entire architecture seem much less plausible. After all, if binding cannot be learned, we would have to tell a difficult-to-verify evolutionary story about its origins. Fortunately, binding can be learned, and it can be learned with this same learning rule used in to learn a communication channel.

Figure 6.11 shows the results of this rule being applied to networks with the same structure as that shown in figure 6.10a, but which is given examples of binding of 3-dimensional vectors that generate the error signal. These simulations demonstrate that the binding operation we have chosen is learnable using a biologically realistic, spike-time sensitive learning rule. Specifically, these results show that the learned network is at least as good as an optimal NEF network with the same number of cells.

This simple demonstration does not settle the issue of how binding is learned,

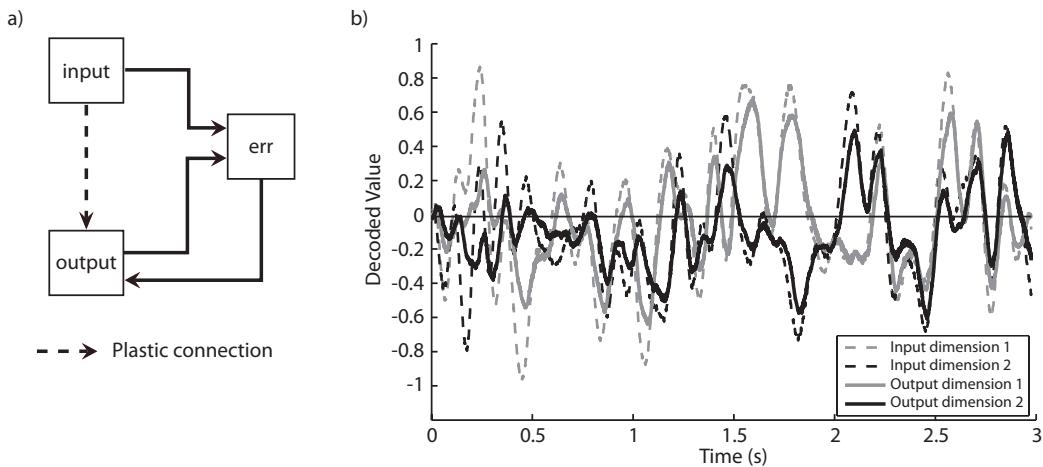


Figure 6.10: A learning network structure and sample run. a) The network consists of an initially random set of connections between an input and an output population that uses Trevor's rule to learn a desired function (dashed line). The 'err' population calculates the difference, or error, between the input and output populations, and projects this signal to the output population. b) A sample run of the network learns a simple 2-dimensional communication channel between the input and the output. Over time, the decoded value of the output population (solid lines) begins to follow the decoded value of the input population (dashed lines), meaning that they are representing the same values.

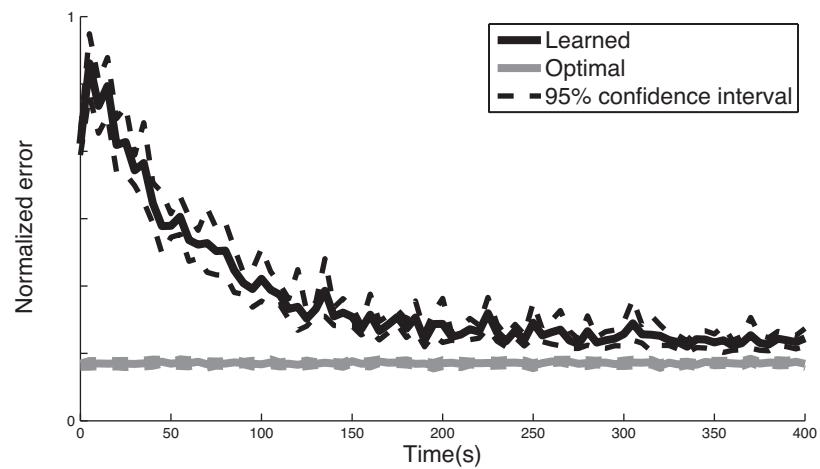


Figure 6.11: Learning circular convolution of vectors. This graph shows the use of Trevor’s rule to learn a multi-dimensional, nonlinear, vector function. Specifically, the network learns the binding operator (circular convolution) used in the SPA on 3D input vectors. As can be seen, the learned network does as well as the one whose weights are optimally computed using the NEF methods. (note: rate rule, 1300 neurons in control, 500 in learning... placeholder???).

of course. For instance, this network is learning the circular convolution of 3-dimensional, not 500-dimensional, vectors. And, notice that it takes about 200s of simulation time to do so. This is actually quite fast, but we have shown that as the dimensionality of the vectors being bound increases, the length of time to learn convolution increases very quickly (Bekolay and Eliasmith, 2011), although we do not yet know the precise relation.

As well, there are important questions to answer regarding how the error signal would be generated, how the error signals can be targeted to the appropriate systems that perform binding, and so on. In fact, it should not be surprising that telling a story about how binding is learned might be difficult: many of the more cognitive behaviors that require binding are not evident in people until about the age of 3 or 4 (e.g., analogy (Holyoak and Thagard, 1995), false belief (Bunge, 2006), multi-place relations in language (Ramscar and Gitcho, 2007), etc.). Consequently, I would argue that this demonstration should allay concerns that there is anything biologically or developmentally implausible about choosing circular convolution (or nearly any kind of vector multiplication) as a binding operation, but it still leaves open the majority of questions related to the cognitive development of binding.

6.5 Example: Learning new actions

The examples of learning that I have presented so far, can be best characterized as learning how to manipulate representations. That is, given some representation, a random initial transformation, and information about a desired transformation in the form of an error signal, the learning rule will update the initial transformation to approximate the desired transformation. This kind of learning may be common and appropriate for much of cortex (e.g. when learning to identify a new object, or extract specific features out of the input). However, it does not directly address a central concern for animals: learning how to act so as to gain benefits from the environment.

In this section, I describe some recent work, also from Trevor Bekolay, which demonstrates how his learning rule can be integrated directly into the model of the basal ganglia discussed in section 5.2. Extending the basal ganglia model with this kind of rule results in a way of tuning the cortical-striatal connections, to allow the system to adapt to changing reward contingencies in the environment (Bekolay and Eliasmith, 2011). That is, it allows the system to learn new ways to control its behavior, depending on what is most rewarding.

The basal ganglia is understood to play a central role in this kind of “instrumental conditioning” (see Maia (2009) for a review). Instrumental conditioning has been discussed in psychology since the 19th century, and identifies cases of an animal changing its behavior in order to give rise to desired results. This is now often thought of as an instance of reinforcement learning, an approach to learning that characterizes how an agent can learn to behave so as to maximize rewards and minimize punishment in an environment. Famously, some of the algorithms of reinforcement learning have been mapped to specific neural signals recorded from animals involved in instrumental conditioning (Schultz et al., 1997).

As briefly mentioned in the previous section, the neurotransmitter dopamine is often singled out as carrying error signals relevant for such learning. More specifically, dopaminergic neurons in the substantia nigra pars compacta (SNC) and ventral tegmental area (VTA) have been shown to mediate learning in the cortical-striatal projection (Calabresi et al., 2007). These neurons fire in a manner that seems to encode a reward prediction error (Bayer and Glimcher, 2005). That is, their firing increases if the animal gets an unexpected reward, and is suppressed if the animal gets no reward when it is expecting one. This information is precisely what is needed for the animal to determine how to change its future actions in order to maximize its reward.

Conveniently, we can simply augment the previous basal ganglia model (figure 5.1) to include these additional dopamine neurons, and their relevant connections to striatal neurons (see figure 6.12). Because dopamine is largely modulatory, the general functioning of the model is not affected in any way, but we can now explore various means of exploiting the dopamine signal (by using our rule), to learn new actions given environmental contingencies. Because it is a modulatory neurotransmitter, we can think of dopamine as providing a channel of input separate from excitatory cortical input, which does not directly cause neurons to fire more or less, but rather informs the synapse how to change given recent rewards (and recent cortical and striatal activity).

For present purposes, we can explore simple but common reinforcement learning tasks used in animal experiments called “bandit tasks”. These tasks are so named because the rewards provided are similar to those given by slot machines (i.e., “one-armed bandits”). For example, in a 2-armed bandit task, a rat enters a simple maze like that shown in figure 6.13a, and must decide whether to take its chances getting rewarded at the left or right reward sites. The reward sites have different probabilities of providing a reward, so the animal is faced with a fairly complex task. It is never guaranteed a reward, but instead must be sensitive to the probability of reward.

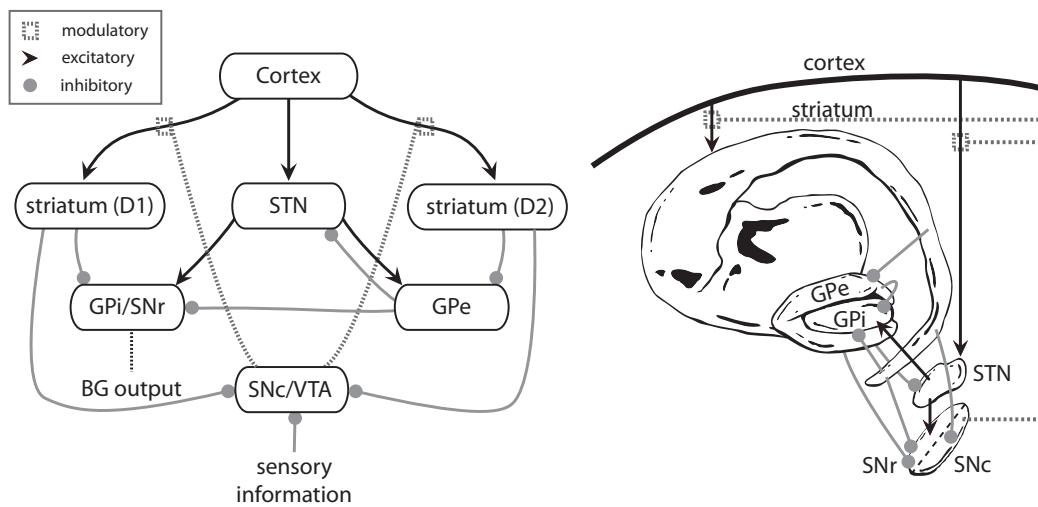


Figure 6.12: The basal ganglia with dopaminergic projections included. The substantia nigra pars compacta (SNC) and ventral tegmental area (VTA) which contain efferent dopamine neurons has been added to the basal ganglia circuit from figure 5.1. The modulatory dopamine connections are shown with dotted lines. These are typically taken to carry information about reward prediction error, based on current sensory information.

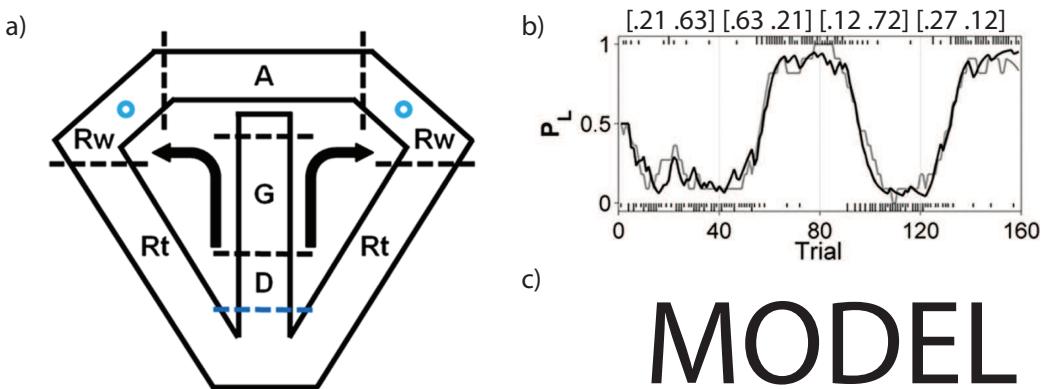


Figure 6.13: The two-armed bandit task. a) The maze used to run the task. The rat waits at D during a delay phase. A bridge between G and A then lowers allowing the animal to move (the go phase). The animal reaches the decision point A during the approach phase, and turns either left or right. Reward is randomly delivered at Rw during the reward phase. Finally, at Rt the animal returns to the delay area during the return phase. b) Experimental data from a single trial (black line), and a mathematical reward model (light gray). The values in brackets above the graph indicate the probability of getting a reward at the left and right reward sites. The animal clearly learns to favor the more rewarded site as it switches over the course of the trial. c) The behavioral response of the spiking basal ganglia model with learning applied in the striatum. The model also learns to favor the more rewarded location as reward contingencies change. (Figures a) and b) adapted from Kim et al. (2009) with permission ???get rid of gray line and colors)

To examine the animal's ability to track these reward probabilities, the likelihood of reward is manipulated during the task. As shown in figure 6.13b, the animal adjusts its behavior to more often select the site that is rewarded with the greatest frequency. The model does the same thing, slowly adjusting its choices of which way to turn to reflect learned reward contingencies. This occurs in the model because the model adjusts which cortical state matches the precondition for the actions by changing the connection weights between cortex and striatum (i.e., those that implement the M_b matrix in the model).

Because the striatum is implicated in action selection, and it receives a dopamine signal that indicates prediction errors, it is a natural structure to record single cell responses from during such tasks. As shown in figure 6.14a, single neurons in the ventral striatum have responses that vary depending on which phase of the task the animal is currently in. These same kind of responses are evident in the neurons in the striatal cells of the model, as demonstrated in figure 6.14b.

While the task considered here is simple, the model is clearly consistent with both behavioral and neural data from this task. Given that the model is identical to that used for implementing much more sophisticated action selection in other contexts (e.g. the Tower of Hanoi), these results are an encouraging indication that the same learning rule employed here can support the learning of more sophisticated state/action mappings as well. So, I take this model to show that the SPA can naturally incorporate reinforcement signals to learn new actions in a biologically realistic manner.

However, this is only one of many kinds of learning accomplished by the brain. As well, this example does not show how low-level learning rules might relate to high-level cognitive tasks. In the next section, I consider a more cognitive application of the same rule.

6.6 Example: Learning new syntactic manipulations

I have considered how, in an abstract sense, Trevor's rule can be used to learn the cognitive binding operation fundamental to the SPA. And I have considered how, in a specific non-cognitive case, the same rule can be used in the basal ganglia to learn appropriate actions given changing environmental contingencies. Now I consider how this rule can be used in a specific cognitive task to support learning of context-dependent reasoning.

In my earlier discussion of the Raven's Progressive Matrices (RPM) in section 4.6, I provided an example of induction that showed how we could build a

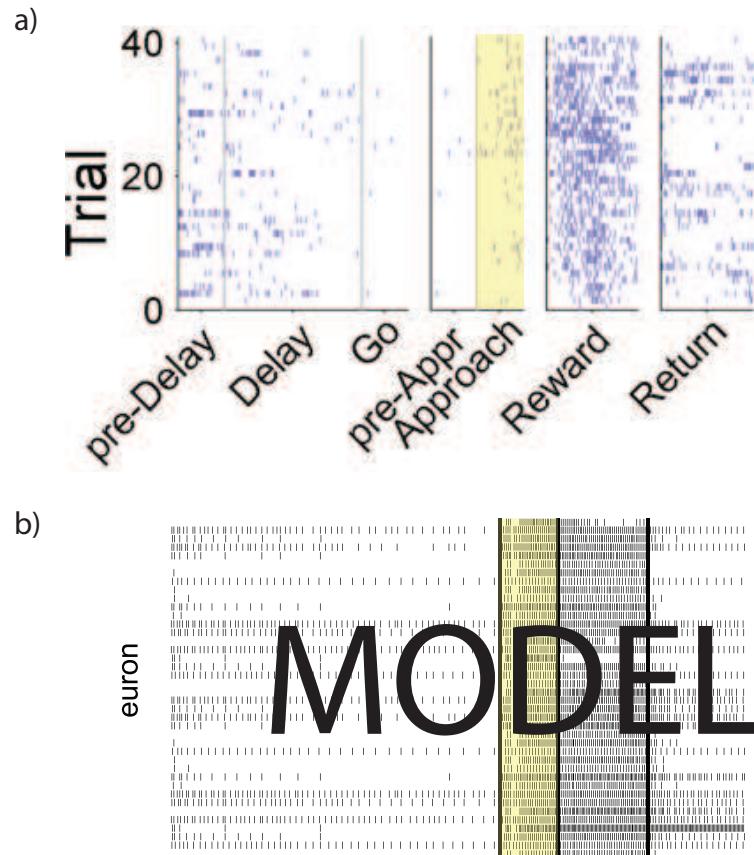


Figure 6.14: A comparison of spiking data from the rat striatum to the model. a) Spike train data over several trials recorded from a single neuron in the ventral striatum. b) A comparison to the spike trains of several neurons in a single trial from the ventral striatum of the extended basal ganglia model. The model and data share several properties, such as a slight decrease in firing during the delay phase, a small increase in firing during the end of the approach phase, and vigorous firing during the reward phase, with a decrease to delay phase levels during the return phase. (Figure a) from Kim et al. (2009) with permission)

system that could learn a new rule based on past examples. I emphasized there that no connection weights were changed in the system, despite this performance. Instead, the system relied on a working memory to store a rule over the presentation of several examples. This has the advantage of being both very rapid, and immediately accessible for further induction. However, it has the disadvantage that once the task is complete, the induced rule is essentially forgotten. In people, however, effective reasoning strategies can be stored in long-term memory and used in the very distant future.

In this section, I consider a similar kind of learning to that in the RPM, but with an emphasis on addressing this longer-term behavior (Eliasmith, 2004, 2005c). For variety, I consider a different task: the Wason card selection task (Wason, 1966). This task is challenging because it requires symbolic reasoning and strategy changes across contexts. In the Wason task, participants are given a conditional rule of the form “if P, then Q” (see figure 6.15). They are then presented with four cards, each of which has either P or not-P on one side, and either Q or not-Q on the other. The visible sides of each card show P, not-P, Q, and not-Q. The task is for the participant to indicate all of the cards that would have to be flipped over to determine whether the conditional rule is true (i.e., is being obeyed). The logically correct answer is to select the cards showing P and not-Q.

There are interesting psychological effects in such a task. If the task is given in an abstract form, such as “if a card has a vowel on one side, then it has an even number on the other”, the majority of participants choose P (the vowel) and Q (the even number) (Cheng and Holyoak, 1985; Oaksford and Chater, 1994). However, if the task is presented using familiar content (remember that most participants are undergraduates), such as “if a person is drinking beer, then that person must be over 21 years old”, the majority choose P (drinking beer) and not-Q (under 21) (Cox and Griggs, 1982). The change in mean performance of subjects is huge, for example going from 15% correct on an abstract task to 81% correct on a familiar version (Johnson-Laird et al., 1972). In other words, structurally identical tasks with different contents lead to differing performance. Explaining this “content effect” requires understanding how symbolic manipulation strategies may change based on the content of the task.

Initial explanations of Wason task performance were based on just the familiarity of the contexts. However, later explanations suggested that the distinction between deontic and non-deontic situations was of greater importance. In deontic situations the rule expresses a social or contractual obligation. In such situations, human performance often matches the logically correct choices, regardless of familiarity or abstractness (Sperber et al., 1995). To explain this, Cosmides

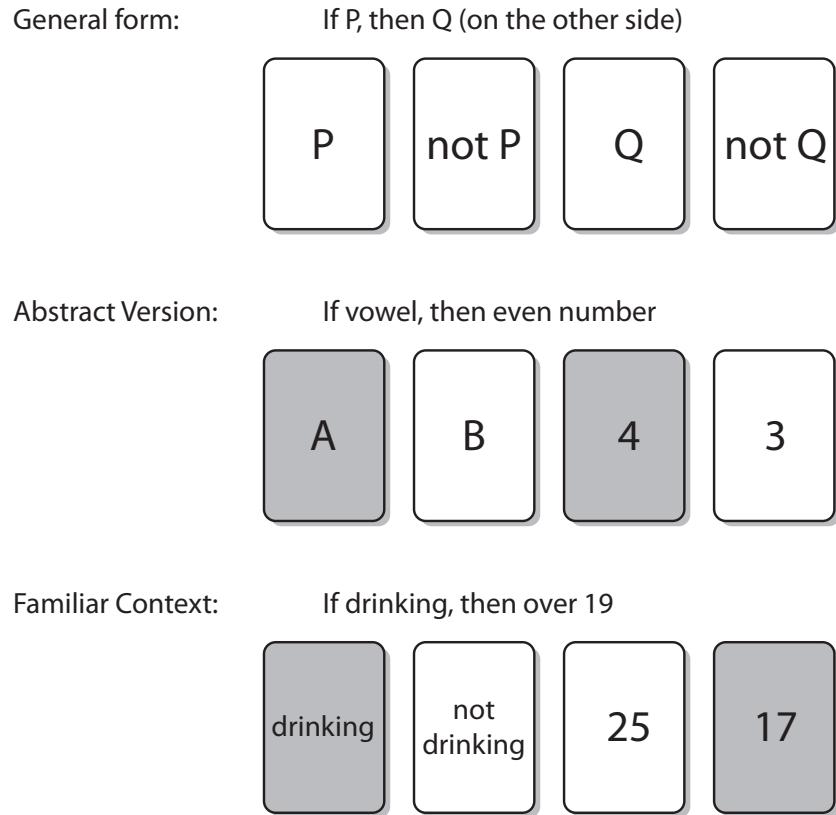


Figure 6.15: The Wason selection task. Four cards are presented to the participant. Each card has information on both sides. A rule is stated that relates the information on both sides of the cards (e.g., if there is a vowel on one side, then there is an even number on the other side). The participant must select the cards to flip over that confirm whether the rule is being followed. The top row shows the general pattern of cards used in the task. Because the rules are always material conditionals, the correct response is always to flip over the first and last cards. However, people respond differently given different contents (showing a “content effect”). Darkened cards indicate the most common selections.

(1989) developed social contract theory (SCT), a view inspired by evolutionary psychology. SCT proposes that natural selection has produced special-purpose, domain specific mental algorithms for solving important recurring adaptive problems. Alternatively, Cheng and Holyoak (1985) proposed that people reason in all cases using neither context-independent rules of inference nor memory of specific experiences, but rather using abstract, context-sensitive knowledge structures induced from ordinary life experiences (some of which relate to obligation and some of which do not): they called these knowledge structures “Pragmatic Reasoning Schemas” (PRS).

Both of these theories use context sensitivity to explain why different formulations of the selection task elicit different responses. However, the theories differ in the source of these rules: PRS schemas are induced through experience while SCT algorithms are genetically endowed. Indeed, Cosmides has challenged any theory based on induction to lay out the mechanistically defined domain-general procedures that can take modern human experience as statistically encountered input, and produce observed domain specific performance in the selection task as output. Responses to this challenge have been made, but none present a neural mechanism. In this section, we will see such a mechanism that is consistent with PRS. And, I will suggest that this mechanism meets the challenge that Cosmides has offered to domain general hypotheses.

In short, this mechanism consists in the combination of induction as in the RPM model, and learned associations that relate contexts to appropriate syntactic transformations. Both the induced transformation and the context-to-transformation mapping are encoded into a long term associative memory using the rule. Figure 6.16 indicates the network architecture and function, and highlights the population where this mechanism is hypothesized to be. The neuroanatomical labels on various parts of the model are supported by a variety of imaging studies (e.g., Goel, 2005; Parsons and Osherson, 2001; Parsons et al., 1999; Kalisch et al., 2006; Adolphs et al., 1995; see Eliasmith (2005c) for details).

The inductive process is similar to that in section 4.6, and determines the correct transformation in a given context. Essentially, b represents the context information and c represents the model’s current guess at an appropriate transformation. The rule changes the connections that map context onto the appropriate transformation, so as to match the feedback from j (i.e., by minimizing the error, which is the difference between the output of c and j).

What may seem familiar about this learning, is that it employs an error feedback signal to inform the modification of connection weights on an input signal. Here, the feedback is provided by the difference between j and c , and the input sig-

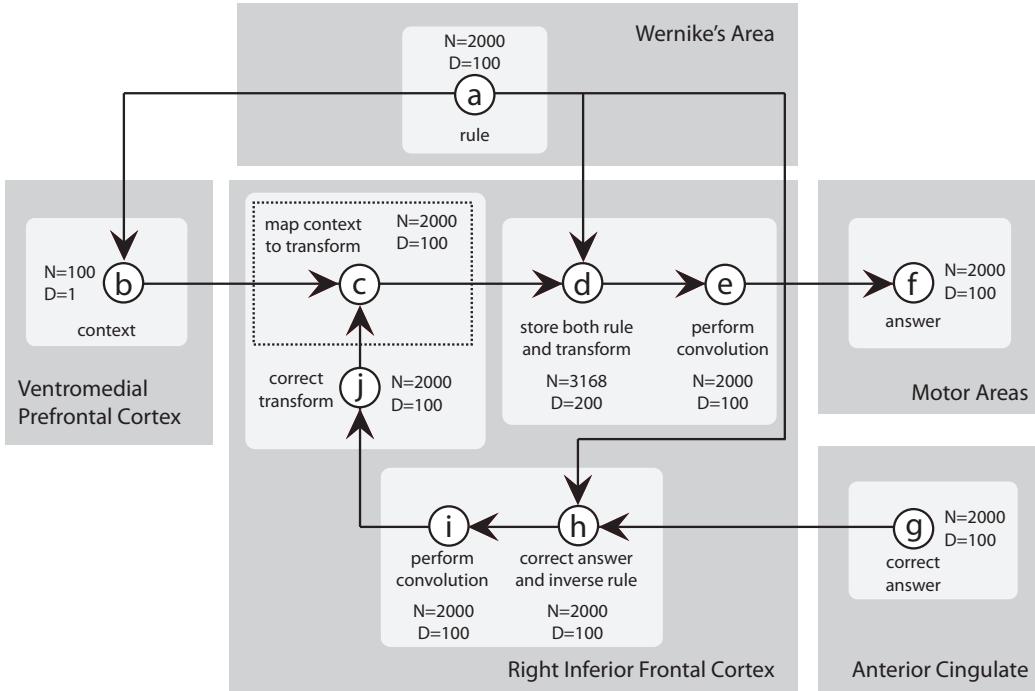


Figure 6.16: The Wason task network. Circles indicate neural populations (N is the number of neurons, D is the dimensionality of the value represented by the neurons). In this network a represents the input rule, b represents the context (based on the semantics of a), c associates the context with a transformation (i.e., a reasoning strategy), d and e apply the strategy (i.e., compute the convolution of the rule and the transformation), f is the result of the transformation (i.e., the answer), g is feedback indicating the correct answer, h and i induce the relationship between the correct answer and the rule (i.e., convolve the inverse of the answer with the rule), and j represents the induced transformation. Arrows indicate neural connections. Population c is recurrently connected. The dotted rectangle indicates the population where associative learning occurs.

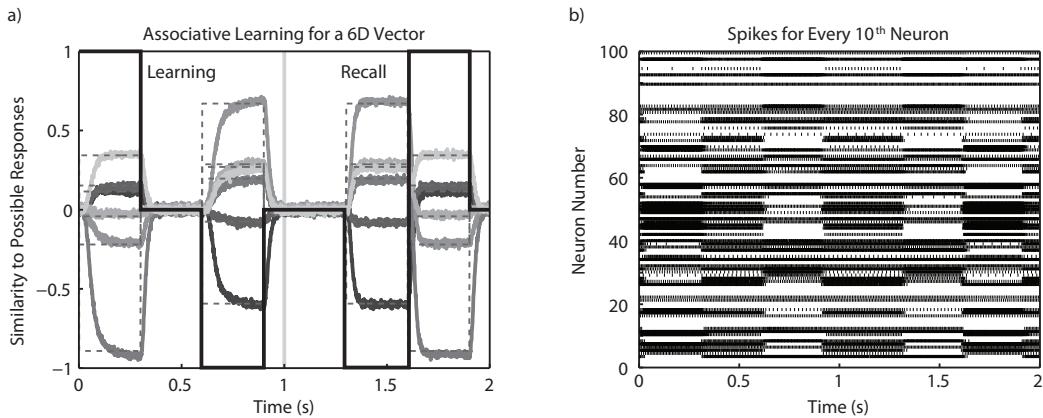


Figure 6.17: Associative learning using Trevor’s rule in a spiking network for two 6D vectors. a) The thick black line indicates the input context signal. There are three contexts, +1, -1, and 0. The grayscale lines each indicate the represented value of one element of the 6D vector. The dotted lines indicate the ideal answers. During the first half of the simulation, two random vectors (coming from j in figure 6.16) are being learned under two different contexts. During the second half of the simulation, only the context signal is applied to the population, which then responds with the vector encoded in that context. The network successfully associates the context signal with a close approximation to the original random vector. b) The spike trains of the neurons in the associative memory population (for the first half of the run, and every tenth neuron, for legibility). There are 1000 neurons in this population. The synaptic weights are initialized to random values.

nal is the context from *b*. Trevor’s rule performs exactly this kind of learning for arbitrary vector transformations. Consequently, we should not be surprised that the model can successfully learn the mapping from context to reasoning strategy using a biologically realistic rule. The results of performing the context-strategy association alone is shown in figure 6.17. However, the context signal in this case is 1-dimensional and the vector it is mapped to is only 6-dimensional for simplicity. In the full model, the context is a vector derived from the semantics of the rule to be manipulated, and the representations and transformations are 100-dimensional.

The representations used in the model are of the form:

rule = relation \circledast implies + antecedent \circledast vowel + consequent \circledast even

for abstract rules, and

rule = relation ⊗ implies + antecedent ⊗ drink + consequent ⊗ over21

for facilitated rules. Base semantic pointers are chosen randomly, and are bound for combinatorial concepts (e.g. **notOver21** = **not ⊗ over ⊗ 21**).

To perform the basic task, the model must learn the appropriate transformation for each of the two contexts. Figure 6.18 shows the entire simulation. The first half of the simulation is a training period during which the model is presented with a rule and the appropriate response. The context is determined by examining the content of the rule, and the ‘correct response’ is determined by direct feedback from the environment. The model is then tested by presenting one rule at a time, without learning or feedback. As shown, the model learns the appropriate, distinct responses under the abstract and facilitated contexts.³

It may seem odd that the model is taught the ‘logically incorrect’ response during the abstract context. However, I am assuming that this is plausible because the ‘logically incorrect’ answer is learned as a result of past experiences in which a bi-conditional interpretation of the “if-then” locution has been reinforced (Feeney and Handley, 2000). The supposition is that when parents say, for example, “If you eat your peas, then you get some ice cream”, they actually mean (and children learn) a bi-conditional relation (i.e., if they don’t eat their peas, they don’t get ice cream). In the Wason task, the ‘correct’ answer is always assumed to be a material conditional (i.e., if they don’t eat their peas, they may or may not get ice cream). I am assuming that the bi-conditional interpretation is a default that is used in the abstract, unfamiliar context. In the facilitated context, the expected response is the ‘logically correct’ answer **alcohol** and **not ⊗ over21**, because this is assumed to be learned through experiences where the content specific cues indicate that this specific rule needs a material conditional interpretation. This characterization is consistent with PRS.

Nevertheless, this kind of result is not especially exciting, since it could be taken to merely show that we can teach a neural net-like model to memorize past responses and reproduce them in different contexts. But, much more is going on in this model. Specifically, we can show that here, as in the RPM model (see section 4.6), the system has the ability to syntactically generalize within a context. Syntactic generalization, as I discussed earlier, has often been identified as a hallmark of cognition.

³This performance is 95% reliable (N=20) over independent simulations. The variability is due to the fact that the 25 different symbol vectors are randomly chosen at the beginning of each run.

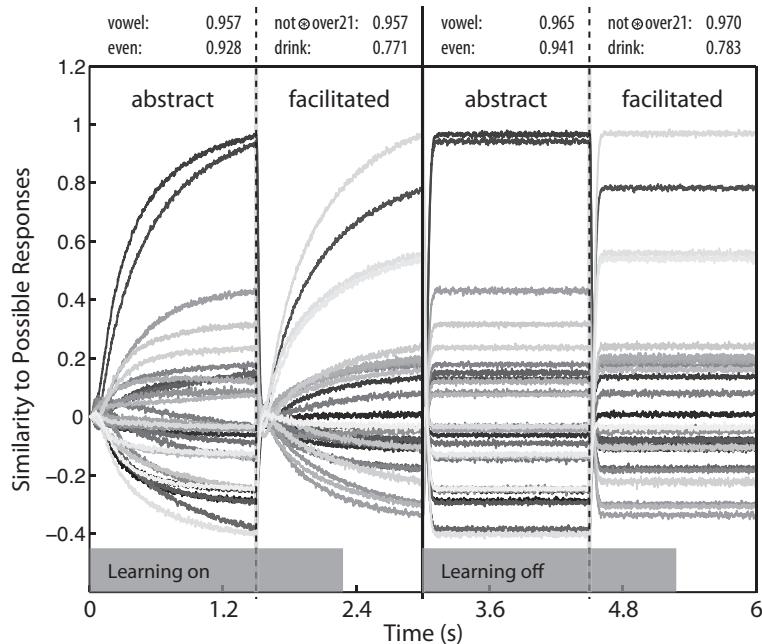


Figure 6.18: Model results for learning and applying inferences across the two contexts. The time-varying solid lines indicate the similarity of the model’s decoded spiking output from population f to all possible responses. The vector names and numerical similarity values of the top two results (over the last 100ms of that part of the task) are shown above the similarity plot. After learning, the system reliably performs the correct, context-dependent transformation, producing the appropriate response.

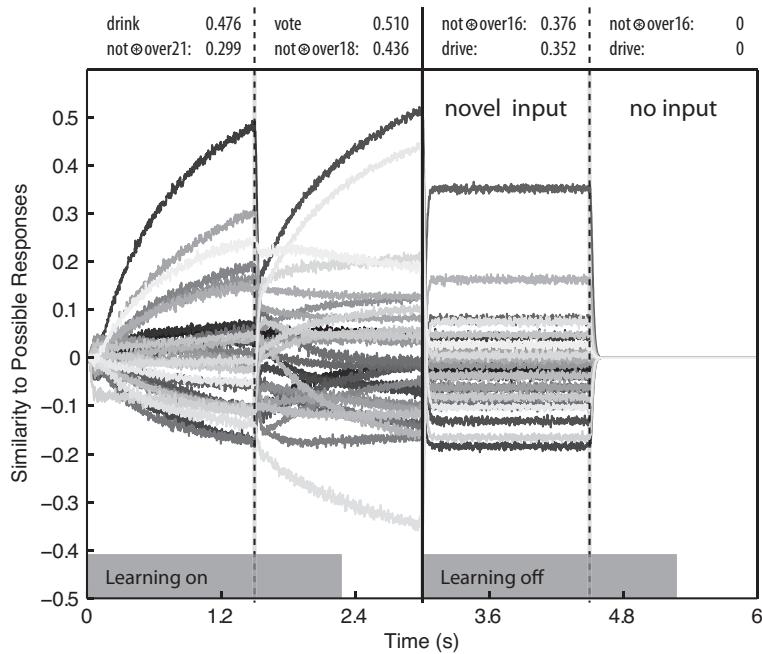


Figure 6.19: Simulation results demonstrating syntactic generalization within a context. The format is the same as described in Figure 6.18. In the first half of the simulation, two different facilitated rules are presented, and the relevant transformation is learned. In the third quarter, a novel facilitated rule is shown and the model correctly infers the appropriate response. The last quarter shows that the model does not respond without input.

For the second simulation, we can define several rules, at least three of which are of the facilitated form (e.g., “if you can vote then you are over 18”). During the learning period, two different examples of a facilitated rule are presented to the model. Learning is then turned off, and a novel rule is presented to the model. This rule has similar but unfamiliar content, and the same structure as the previous rules. The results of the simulation are shown in figure 6.19. As shown, the model is able to correctly reason about this new rule, providing the P and not-Q response. So, the model clearly has not memorized past responses, because it has never produced this response (i.e. **not** \otimes **over16** + **drive**) in the past. This is a clear case of syntactic generalization, since it is the syntax that is consistent between the examples and the new rule, not the semantics.

In essence, the semantics is still being used, because it is determining that the

context of the new rule is similar to the already encountered rules. However, the inference itself is driven by the syntactic structure of the rule. Multiple examples of the combination of semantics and syntax are needed to allow these to be appropriately separated. This is consistent with the fact that performance on this task improves with instruction that emphasizes syntax over content (Rinella et al., 2001).

Because this model is implementing a psychological theory at the neural level, there are several ways in which we can examine the model. For example, one key feature of models developed with the SPA approach is robustness to both background firing noise and neuron death. Since vector representations are distributed across a group of neurons, and since this distribution does not require a direct mapping between particular neurons and particular values in the vector, performance degrades gracefully with damage. To demonstrate this in the current model, we can randomly remove neurons from the “rule and transformation” population (population d in figure 6.16). The results shown in figure 6.20 were generated after training, but before testing on the same task as shown in figure 6.18. On average, accurate performance occurs even with an average of 1,221 (variance=600, $N=500$) neurons removed, out of 3,168.

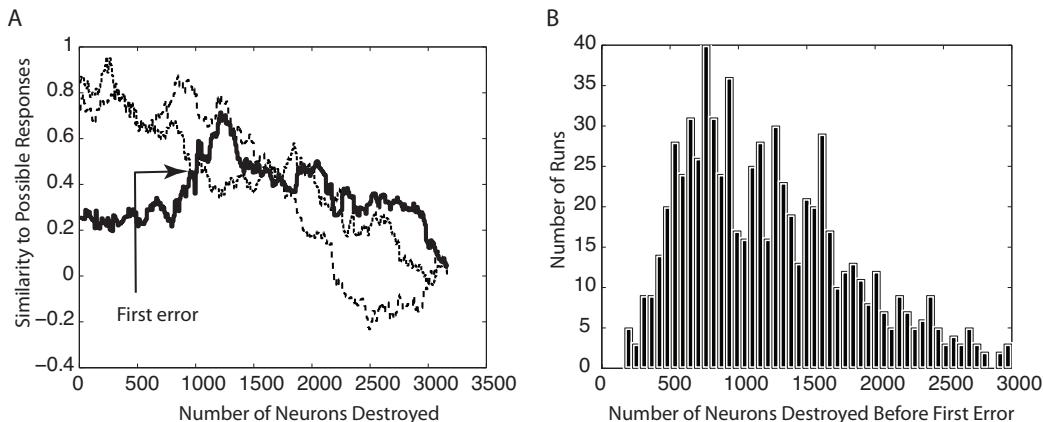


Figure 6.20: Performance of the Wason model under random destruction of neurons in population (d) in figure 6.16. A. The effect of continually running the model while random neurons are destroyed until none are left, showing the three symbols whose VSA representation most closely match the generated output. An error is made if either one of the original two top responses falls below the third. B. A histogram of the number of neurons destroyed before the first error for 500 experiments (mean=1221).

We can also explicitly test the robustness of the model to intrinsic noise, instead of neuron death. In this case, the model functions properly with up to 42% Gaussian noise used to randomly vary the connection weights.⁴ Random variation of the weights can be thought of as reflecting a combination of imprecision in weight maintenance in the synapse, as well as random jitter in incoming spikes. Eliasmith and Anderson (2003) have explored the general effects of noise in the NEF in some detail, and shown these representations to be robust (see also Conklin and Eliasmith (2005) for a demonstration of robustness in a model of rat subiculum). In short, these results demonstrate a general robustness found in SPA models, which is present largely because these models are based on the NEF, and hence deploy distributed representations. Conveniently, such biologically plausible robustness allows close comparison of NEF and SPA models to either micro-lesion (ref macneil???) or large-scale lesion data (ref???hemingelect) if it is available.

In many ways, the robustness of the model is a result of its low-level neural implementation. But, as mentioned earlier, the model is also implementing a cognitive hypothesis (i.e., that domain general mechanisms can learn context-sensitive reasoning). As a result, the model can also be examined in a very different way, relating it to more traditionally “cognitive” data. For example, we can use the model to gain some insight into how humans are expected to perform given examples of novel syntactic transformations. Qualitatively, the model is consistent with evidence of practice effects in reasoning tasks (Rinella et al., 2001). However, we can also more specifically examine performance of the model as a function of the number of examples. In table 6.1 we can see that generalization improves rapidly with the first three examples, and then improves less quickly. We can thus predict that this would translate behaviorally as more rapid and more consistent responses after three examples compared to one example.

Beyond these behavioural predictions, the model also embodies a specific prediction for cognitive neuroscience: namely, that the central difference between the abstract and content facilitated versions of the Wason task is the context signal being provided from the VMPFC, not the mechanism of reasoning itself. This is very different from the SCT suggestion that evolutionarily distinct reasoning mechanisms are necessary to account for performance differences on these two versions of the task (Cosmides, 1989). Experiments performed after the presentation of an early version of this model (Eliasmith, 2004) have highlighted the

⁴Each synaptic connection weight was independently scaled by a factor chosen from $N(1, \sigma^2)$. The model continued to function correctly up to $\sigma^2=0.42$.

Table 6.1: Improvement in generalization with learning history. Accuracy is the difference between the VSA representations of the correct answers and the largest incorrect answer, scaled by the largest incorrect answer. A number below 0 indicates that the wrong answer is produced, and larger numbers indicate greater separation between the correct and incorrect answers.

# of examples shown	Generalization accuracy
1	-0.10
2	0.05
3	0.23
4	0.26

VMPFC as the locus of cognitive processing differences on the two versions of the task (Canessa et al., 2005), supporting this suggestion.

This result also highlights why I believe that this model meets Cosmides' original challenge to provide a plausible domain general mechanism able to explain task performance while being inductive. In particular, Cosmides suggests that the inductive evidence during the normal course of human development cannot serve to explain the superior performance on the content facilitated task because the relative amount of evidence would not favor the correct response. However, the inductive mechanism provided here does not rely on the relative amount of evidence available in different contexts. Instead, it relies on a minimum threshold of available evidence (i.e., enough for the learning rule to induce the relevant transformation) within a given context. As a result, it is a domain general, inductive mechanism that accounts for context effects on the Wason task, as only VMPFC activity changes across contexts.

Finally, while closely allied to PRS, unlike PRS this model does not assume that the transformations used to solve the problem should be limited to a theoretically pre-determined group of ‘schemas’ which are the same for all subjects (e.g., the ‘permission’ schema, the ‘obligation’ schema, etc.). Instead, each individual will solve the problem using his or her own estimation of the correct transformation in the given context, as determined by his or her personal experience and learning history in that context. The contexts that can distinguish behavior are thus not restricted to clear-cut theoretical distinctions (e.g., deontic/non-deontic), but are rather driven by the similarity of context signals to past history to determine which transformations are applied. That is, the similarity space provided

by context signals will be mapped onto a transformation space. This mapping is discovered by the model based on past examples, and may not neatly align with our theoretical classifications of context. This is consistent with findings that successful performance of the task does not align neatly with intuitive theoretical distinctions (Oaksford and Chater, 1996; Almor and Sloman, 1996), and can vary with past experience (Rinella et al., 2001).

This example concludes my discussion of learning, and indeed my presentation of the SPA. However, seeing all the parts of the SPA does not capture one of its main goals; being an integrative architecture. As a result, the next chapter collects many of the pieces I presented so far together, and presents a single model that is able to incorporate a variety of the behaviors and principles we have seen to this point. My motivation is to demonstrate that the SPA examples we have seen are not task-specific, one-off models, but rather are parts of a coherent, unified picture of brain function.

6.7 Nengo: Learning

Accounting for synaptic plasticity is crucial for any characterization of brain function. In this tutorial, I demonstrate how to include synaptic plasticity in Nengo models. By default, the learning template in the toolbar in Nengo’s interface uses Trevor’s rule. Nengo supports a much wider variety of learning rules, but many of these require use of the scripting interface, which I will not consider until the next tutorial.

The implementation of learning employed by the Nengo template uses an error signal provided by a neural ensemble to modulate the connection between two other ensembles. This template can be used to learn any transformation when an error signal is present.

- Open a blank Nengo workspace and create a new network named ‘Learning’.
- Create a new ensemble named ‘pre’. Set *Number of Nodes* to 50, *Dimensions* to 1, *Radius* to 1.0, and use a normal LIF Neuron.
- Create a second ensemble named ‘post’. Use the same parameters for this ensemble as you used for the ‘pre’ ensemble.
- Drag the ‘Learning Rule’ template into the network. In the template constructor, name the error ensemble ‘error’ and give it 100 neurons. Set *Name*

of pre ensemble to ‘pre’, *Name of post ensemble* to ‘post’ , and *Learning rate* to ‘5.0e-7’. Press *OK*.

The template adds a new ensemble named ‘error’ and connects the ‘pre’, ‘post’ and ‘error’ ensembles together. The connection between the ‘pre’ and ‘post’ ensembles is initialized with random weights, but this connection will be updated with weights that correlate to small values in the ‘error’ population. Therefore, to learn a transformation, we need to establish an error signal using recurrent connections from the ensembles with plastic connections.

- Add a termination to the ‘error’ ensemble. Name the termination ‘pre’, set the input dimension of this termination to 1, and set the connection weight to 1.
- Add a termination to the ‘error’ ensemble. Name the termination ‘post’, set the input dimension of this termination to 1, and set the connection weight to -1.
- Project from the ‘pre’ ensemble to the ‘pre’ termination on the ‘error’ ensemble and likewise project from the ‘post’ ensemble to the ‘post’ termination.

The two terminations added to the ‘error’ ensemble will subtract the value of the ‘post’ ensemble from the value of the ‘pre’ ensemble, resulting in a representation of the difference being stored in ‘error’. Minimizing this difference in values will push the ‘post’ ensemble towards representing the same value as the ‘pre’ ensemble, creating a communication channel.

To test this channel, we need to add a few extra elements to the network. First, we’ll create an input function to generate data to send through the communication channel. Then we’ll add a gate to the ‘error’ network that will allow us to switch learning on and off and, finally, we’ll add a second error population that will calculate error directly (not using neural computations).

- Create a new function input. *Name the input ‘input’*, set *Output Dimensions* to 1 and click the ‘Set Functions’ button.
- From the drop-down list, select ‘Fourier Function’ and click *Set*.
- Set *Fundamental* to 0.1, *Cutoff* to 10, *RMS* to 0.5, and *Seed* to an integer.
- Finalize the function setup by clicking *OK* on all the open dialogs.

- Right-click the created function and select ‘Plot function’. Set the index to 0, start to 0, increment to 0.01, and end to 10.

The function that is graphed will provide the randomly varying input our communication channel. The function is defined in terms of its frequency components, but the details of these aren’t important as the function only needs to provide interesting and variable output. The *Seed* parameter can be changed to draw different random values.

- Close the graph of the input function and add a decoded termination to the ‘pre’ ensemble. The termination should have one dimension and a connection weight of 1.
- Project from the function input to the ‘pre’ ensemble.
- Add a gate to the network. Name it ‘gate’, set the gated ensemble to ‘error’, set the number of neurons to 100, and set the time constant to 0.01.
- Add a constant function named ‘switch’, with a default value of 0. You will need to click the ‘Set Functions’ button again to switch from using a *Fourier Function* to a *Constant Function*.
- Add a decoded termination to the ‘gate’ ensemble that accepts one-dimensional input with a connection weight of 1.
- Create a projection between the ‘switch’ and ‘gate’ ensembles.
- Create a new ensemble named ‘actual error’. Set the number of nodes, dimensions, and radius each to 1.
- Click the ‘actual error’ ensemble and, with this ensemble selected, use the drop-down menu at the center of Nengo’s top menu bar to change the simulation mode from *spiking* to *direct*.
- Add a ‘pre’ termination to the ‘actual error’ population that has a connection weight of 1 and a ‘post’ termination that has a connection weight of -1 (as before, with the ‘error’ population).
- Project from the ‘pre’ and ‘post’ populations to their respective terminations on the ‘actual error’ population.
- Open *Interactive Plots*.

The interactive plots viewer should display value plots for each of the ensembles in the network, including our single-neuron ‘actual error’ ensemble that is running in direct mode. The label for the ‘actual error’ ensemble itself isn’t shown in the default layout since this ensemble isn’t really part of the network but has been added to help analyze the system. The connection weight grid, located in the bottom-right corner of the viewer, is a visualization of connections between the ‘pre’ and ‘post’ ensembles. Red squares represent excitatory connections and blue squares represent inhibitory ones; the intensity of the colour indicates the strength of the connection. The slider control in the top-left corner can be used to toggle the gate on the error population. Since we set the value of the ‘switch’ function to zero, the gate is initially closed, no error signal will be created, and thus no learning will occur.

- Run the simulation.

The ‘actual error’ plot should show an unsurprisingly large amount of error between the two populations since no learning is occurring. The connection weights values should also not change significantly in the grid visualization.

- Reset the simulation.
- Set the value of the ‘switch’ input to 1 using the slider.
- Run the simulation.

After a short initial period of learning, the error reported in this simulation should be noticeably smaller and more stable than the error without learning. Looking at the graphs of the ‘pre’ and ‘post’ values should also reveal that the value of ‘post’ does roughly track the value of ‘pre’. There’s a good chance that you will still not be able to see the connection weights change in the grid. Most of the large weight changes occur in the first second of the simulation and the gradual changes are difficult to discern visually. To see the differences in weights, rewind the simulation to the beginning and drag the time slider quickly from the start of the simulation (after the initial connection weights are displayed) to the last recorded time. You should be able to notice some differences, although they may still be subtle.

Extending the learning network to learn functions only requires changing the error signal used to modulate the plastic connection.

- Create a new decoded origin on the ‘pre’ ensemble. Set its name to ‘square’ and the number of output dimensions to 1, then press ‘Set Functions’.

- Set the function to ‘User-defined Function’, click *Set*, and type ‘ $x0*x0$ ’ as the expression. Complete the origin with default values.
- Remove the projections from the ‘X’ origin of ‘pre’ to the two error ensembles, and replace them with projections from the ‘square’ origin.
- Run the simulation in Interactive Plots, with the ‘switch’ input set to 1.

The ‘square’ origin is computing the squared value of the variable represented by the ‘pre’ ensemble. This transformation is identical to the example we tested in the section 3.8 tutorial, but we’re using it here to generate an error signal that will direct learning between the ‘pre’ and ‘post’ ensembles. We could certainly have projected directly from the ‘pre’ ensemble to a decoded termination on the ‘post’ ensemble instead of setting up the learned connection, so this example is a bit of a ‘toy’ demonstration of the capabilities of the learning technique as opposed to a model of real biological processes. Nonetheless, you will likely encounter one interesting effect while running the simulation that is a direct result of learning with synaptic weights: namely, it may take the network a rather long time to correct its earlier training. The weights on the ‘post’ network were not re-initialized to random values before we ran the latest simulation, so they will still retain the learned weights from the communication channel. Negative values in particular are mapped to very different values under the communication channel and squaring training schemes, so expect these values to have noticeable errors until they are retrained.

Chapter 7

The Semantic Pointer Architecture (SPA)

7.1 A summary of the SPA

I have spent the last four chapters presenting the theoretical justification, and several example implementations of the central elements of the Semantic Pointer Architecture (SPA). Because that presentation has been lengthy and somewhat detailed it may seem fragmented, and hence does not provide a good sense of how the architecture yields a unified characterization of biological cognition. The purpose of this chapter is to explicitly tackle the issue of unification by presenting a single model that includes the features of the SPA presented to this point. Consequently, in this chapter I focus on demonstrating that an SPA model can perform a wide variety of cognitive (and non-cognitive) tasks without changing any aspects of the model.

First, however, let me provide a brief review of the SPA. I have attempted to graphically summarize a standard SPA subsystem in figure 7.1. I think of this ‘standard subsystem’ as a schema, whose details will depend on the specific area of the brain we are mapping it to. For example, the number of levels in the hierarchy, the nature of the transformation, the kinds of control, the flow of error information, etc. will all depend on whether we are considering vision, audition, motor control, or more abstract functions. Regardless, we can begin to get a ‘whole brain’ view if we consider linking such subsystems into a larger system, as shown in figure 7.2.

Figure 7.2 is a higher-level schema that SPA models can be mapped on to.

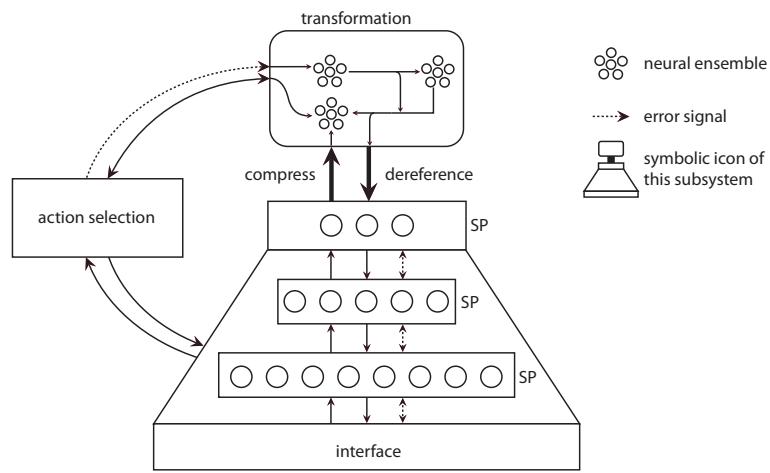


Figure 7.1: A schematic of a ‘standard subsystem’ of the Semantic Pointer Architecture. A high-dimensional representation at an interface to the environment, or other system is compressed through a hierarchical structure, generating semantic pointers. Moving up or down in this hierarchy compresses and dereferences the semantic pointer representations. Throughout the hierarchy, the generated semantic pointers can be extracted and transformed by other elements of the system. Those transformations are updated by error signals, some of which come from the action selection component. The hierarchical structure is also connected to the action selection component of the architecture. The action selection component influences routing within the system.

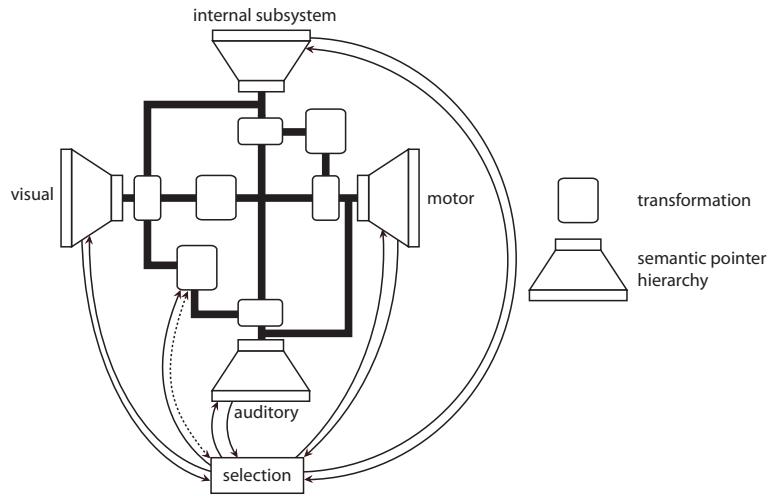


Figure 7.2: A schema of Semantic Pointer Architecture models. This figure consists of several interacting subsystems depicted in figure 7.1. Dark black lines represent projections between parts of the system, while thinner lines indicate control or error signals. An ‘internal subsystem’ may be concerned with functions like working memory, encoding a conceptual hierarchy, etc. Not all information flow is captured by this schema (e.g., many error signals are not present).

Not all models will have all of the same components, though ideally no models will have conflicting components. My intent is, after all, to provide a means of constructing unified models of biological cognition. Filling in such a schema is essentially how I think we ought to build a brain.

That being said, it should be evident that my description of the SPA does not make this a simple matter. The SPA is clearly not yet detailed enough to specify a complete (or even stable) set of subsystems and their functions. It may even be helpful to think of the SPA as providing something more like a protocol description than an architecture – it describes how neural systems can communicate and control the flow of information. However, the previous chapters also include specific commitments to particular processes occurring in identified parts of the system, which is why I have chosen to call it an architecture.

It is, however, an architecture in development (see section 10.2 for further discussion). It is thus more useful to provide example applications of the architecture than an application independent specification, as I have been doing throughout the book. And, in the next section I provide the final example in the book. Here, unlike with past examples, my purpose is to bring together the parts of the archi-

tecture I have described. As a result, I focus on integrating the various behaviors we have encountered to this point.

7.2 An SPA unified network

In the interest of brevity, I refer to this model as Spaun (SPA Unified Network). Spaun is a direct application of the SPA to eight different tasks, but in a limited domain. In particular, all of the tasks are defined over input that is natural hand-written digits plus some additional task specific symbols (see figure 7.3a). This has the advantage of being a significantly variable, real-world input, while also providing a reasonably limited domain that the model must reason about.

Spaun's output is arm motions that draw its response to the given task. Example motor output is shown in figure 7.3b. In short, we can think of Spaun as having a single, fixed eye and a single 2-joint arm. The eye does not move, but instead the experimenter changes the image falling on it by showing different inputs over time. To begin a specific task, Spaun is shown a particular letter. The subsequent input will then be interpreted by the model as being relevant to that task and processed accordingly, resulting in arm movements that provide Spaun's response to the input in the identified task.

I have chosen the tasks to cover various kinds of challenges that face a biological cognitive system. The tasks, a brief description, and a brief justification are shown in table 7.1. In short, I have attempted to choose tasks that demonstrate that integrating the previously described behaviors into a single model is possible without their interfering with one another. As well, these tasks cover simple demands faced by all biological systems (e.g., reinforcement learning and recognition), as well as tasks only known to be successfully solved by humans (e.g., rapid variable creation and fluid reasoning). They also cover behaviors that deal mainly with perceptual and motor processing (e.g., copy drawing and recognition), and tasks that depend more on the manipulation of structured and conceptual representations (e.g., counting and question answering). Together these tasks place a challenging set of demands on a biologically realistic model system.

As mentioned, the purpose of Spaun is to address all of these tasks in a single model. I take this to mean that the model cannot be externally changed during any of the tasks or between any of the tasks. As well, the only input to the model can be through its perceptual system, and the only output can be through its motor system. So, the representational repertoire, background knowledge, cognitive mechanisms, neural mechanisms, etc. must remain untouched while the system

a) Input

0	1	2	3	4	5	6	7	8	9
O	I	2	3	4	5	6	7	8	9
W	R	L	M	C	A	V	F		

[] ? P K

b) Output

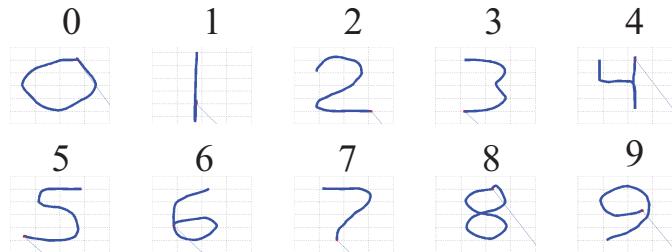


Figure 7.3: Input and output with examples for the Spaun model. a) The complete set of possible inputs with hand-drawn examples for each symbol. Numbers are used as the conceptual domain. Letters on the second row indicate which task is being tested: W for copy drawing; R for recognition; L for reinforcement learning; M for serial memory; C for counting; A for question answering; V for rapid variable creation; F for fluid reasoning. The symbols on the third row are for task control: the brackets are used to indicate grouping; the question mark is used to indicate that a response is expected; and the letters are question modifiers where P is for ‘position’ and K is for ‘kind’. b) The complete set of output with Spaun drawn examples of each kind of symbol. The thin solid lines indicate preparatory movements for which the ‘pen’ is not down. Large dots indicate the starting position of the pen-down movements.

Table 7.1: A description and justification for the eight example tasks performed by Spaun.

Name	Description	Justification
Copy drawing	The system must reproduce the visual details of the input (e.g. different 3s might look different).	Demonstrates that low-level visual information is preserved and accessible to drive the motor response.
Recognition	The system must identify the variable input as a member of a category.	Demonstrates that input variability can be accounted for. Allows the generation of stable concepts.
Reinforcement learning	The system must learn environmental reward contingencies.	Demonstrates reinforcement learning in the context of a cognitive model.
Serial memory	The system must memorize and reproduce an ordered set of items.	Demonstrates the integration of serial memory into a cognitive model. Needed for several other more complex tasks.
Counting	The system must be able to count from a presented starting position for a given number of steps.	Demonstrates flexible action selection. Demonstrate knowledge of the conceptual domain.
Question answering	The system must answer questions about the contents of structured representations in working memory.	Demonstrates the ability to construct (bind), manipulate, and extract various information from internal representations.
Rapid variable creation	The system must identify syntactic patterns, and respond to them without connection weight changes.	Demonstrates a neural architecture able to meet these cognitive demands.
Fluid reasoning	The system must solve problems analogous to those on the Raven's Progressive Matrices.	Demonstrates internal generation of a solution to a challenging fluid reasoning task.

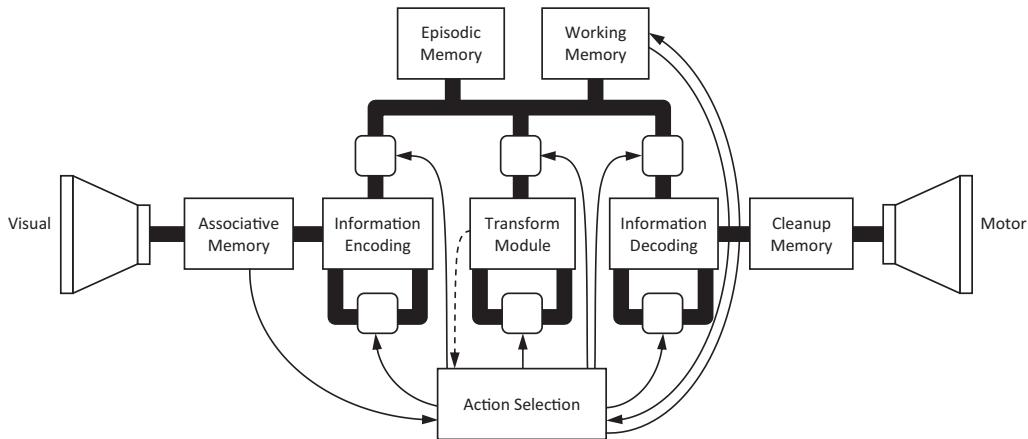


Figure 7.4: The Spaun architecture. Thick lines indicate possible information flow between elements of cortex, thin lines indicate flow of information between the action selection mechanism and cortex, and rounded boxes indicate control states that can be manipulated to control the flow information within and between subsystems. See text for details.

performs any of the tasks in any order.

A typical run would consist of showing Spaun a task letter (e.g. ‘C’) to indicate which task it is expected to perform on the upcoming input (e.g., counting). The next input provided will be a number which indicates where the counting should start from. The third input will be a number indicating how many times to count, and the final input will be a ‘?’, indicating that a response is expected. The system will then proceed to count internally in accordance with the task constraints and generate motor commands that produce a number for its response, at which point it will wait for further input. It is then up to the experimenter to provide another task letter, followed by input, at which point the system will again respond, and so on.

The architecture of Spaun is shown in figure 7.4. This is a specific instance of the SPA architecture schema introduced in figure 7.2. The architecture consists of two hierarchies, a working memory, an episodic memory, an action selection mechanism, and five subsystems. The hierarchies include a visual hierarchy, which compresses image input into a semantic pointer, and a motor hierarchy which dereferences an output semantic pointer to drive a two-degree-of-freedom arm. The working and episodic memories function as described in section 6.2 on working memory, providing stable, longer term representations of task input

and context. The action selection mechanism is the basal ganglia model described throughout chapter 5. The five subsystems, from left to right, are used to: 1) map the visual hierarchy output to a conceptual representation; 2) encode structure in the input through binding; 3) transform the input to generate a response appropriate to the task; 4) decode the result of model processing to a temporally ordered output; 5) map each output item to a semantic pointer that can be used to drive the motor hierarchy. Several of these subsystems have multiple components able to perform the identified function. For instance, the working memory subsystem includes six independently manipulable memories, each of which can store a semantic pointer. This is essential for independently tracking input, output, task state, and so on. Additional details necessary to fully re-implement the architecture are available in (choo & eliasmith, 2011, online tech report??). The model can also be downloaded from ???.

Notice that the architecture is not determined by the set of tasks being implemented, but rather captures more general information processing constraints. This helps make the model extensible to a wide variety of tasks, beyond those considered here. As a result, I believe Spaun demonstrates that the SPA provides a general method for building flexible, adaptive and biologically realistic models of the brain. Before returning to this point, let me turn to consideration of the specific tasks identified earlier.

7.3 Tasks

Each of the following sections describes one of the tasks in more detail, showing input and output examples, and providing summary performance information as appropriate.

7.3.1 Copy drawing

The copy drawing task consists of requiring Spaun to attempt to reproduce the visual features of its input using its motor system. The task consists of showing an input digit to Spaun, removing it, and having the model attempt to reproduce the digit by drawing. Spaun must generate the motor command based on the semantic pointer information, as the image itself is taken to be removed from view.

This task turns out to be quite simple, given the kind of representation generated at the highest level of the visual hierarchy (i.e., the 50D semantic pointer space), and the simplicity of the highest level of motor representation. In brief,

Spaun categorizes the digit, and then uses a map for that category to determine how semantic pointers of that category result in an ordered set of coordinates in 2D space that drive the motor system. The mapping for each category is linear, and is learned based on example digits/motor command pairs (we use 5 pairs for each digit).

- figure of examples of input/output for all numbers
- compute average similarity of standard drawing to input vs. average similarity of copy drawing to input using some measure (dot product?) and report for each number

Notably, the reason the mapping from the visual semantic pointer to the motor semantic pointer is linear is because of the sophistication of the compression and decompression algorithms built into the perceptual and motor systems. In general, it would be incredibly difficult to attempt to map directly from pixel-space to muscle-space.

Spaun's ability to perform the copy drawing task demonstrates that it has access to the low-level perceptual (i.e. deep semantic) features of the representations it employs, and that it can use them to drive behavior.

7.3.2 Recognition

The recognition task requires Spaun to classify its input, using the classification to generate a motor response that reproduces Spaun's default written digit. To perform classification, Spaun takes the 50D semantic pointer produced by the visual hierarchy, and passes it through an associative, clean-up-like structure which maps it to a canonical semantic pointer. This pointer is then routed to the motor system by the basal ganglia, which then maps it to a control signal appropriate for drawing the relevant number.

- show example successes
- show example failures (any 'i don't know' or non-responses? probably not)
- determine percent accuracy over non-trained set

Successful recognition of hand-written input demonstrates Spaun's ability to 'see through' the variability often encountered in naturalistic input. In short, it shows that variable input can be categorized, and that those categories can effectively drive action.

7.3.3 Reinforcement learning

The reinforcement task is an n -armed bandit task, analogous to that discussed in section 6.5. In this case, Spaun is simply told that it is in the reinforcement task with an ‘L’. At this point, it is shown ‘?’ and picks a number. It is then shown a ‘1’ if it receives a reward, and nothing otherwise. This process continues until another task letter is shown.

Whether or not a reward is given depends on a probability associated with each of the possibly rewarded actions, set by the experimenter. The STDP rule and its application are the same as in section 6.5. The basal ganglia in this case selects the number to be chosen as usual, and routes the visual information such that it is interpreted as a reward signal driving SNr/VTA at the appropriate times in the task.

- pick a specific n (3? or 10?), or indicate it at the beginning of the task... which?
- show choice behaviour with switches in reward & spiking neurons

Spaun’s replication of reinforcement learning behavior demonstrates that the model is adaptive, yet stable throughout the set of tasks. More generally, it shows that Spaun can learn about environmental contingencies and adjust its behavior in response.

7.3.4 Serial memory

In the serial memory task, Spaun is presented with a list of numbers that it is expected then recall in various ways. The basic implementation is similar to that described in section 6.2, although it is integrated with a fuller architecture. Notably, this is the first task that explicitly requires binding (although the recognition task uses the same working memory subsystem). As a result, it is important to be explicit about the representations being used.

In short, representations of items at a position in a list are constructed by binding the position to the item (e.g., $\mathbf{position}_i \circledast \mathbf{item}_i$), and summing over all positions. When constructing a memory trace, the recurrent nature of the encoding makes the representation slightly more complicated, as we encountered previously. As a result, the memory trace representations used by Spaun are of

the form:

$$\begin{aligned}\text{trace}_i &= \rho \mathbf{Episodic}_{i-1} + (\mathbf{position}_i \circledast \mathbf{item}_i) + \dots \\ &\quad \gamma \mathbf{WorkMem}_{i-1} + (\mathbf{position}_i \circledast \mathbf{item}_i)\end{aligned}$$

where the γ and ρ parameters are set to the same values as previously (see section 6.3).

Typical input for this task would be ‘3 2 4 2 3 <variable pause> K ?’, where the letter at the end of the list indicates if the list should be reproduced backwards (‘K’) or forwards (‘P’).

- figure showing the input and reproduction for a list with/without errors
- figure with primacy/recency curve fwd/bckwd, effects of having a really long pause?

The generation of human-like memory response curves in Spaun show that its working memory performance maps well to known behavior. Because working memory is central to several subsequent tasks, this shows that Spaun can perform these tasks with appropriate limitations.

7.3.5 Counting

The counting task demonstrates simple learned sequences of action, similar to the examples in section 5.4 and 5.6. One interesting difference is that the internal representations themselves contain more structure. Specifically, the representation for numbers are constructed through recursive binding. This proceeds as follows:

1. Choose a random unitary vector¹, call that **one**
2. Construct the next number in the sequence through self-binding, i.e. **two** = **one** \circledast **one**
3. Repeat

This process will construct distinct but related vectors for numbers. Crucially, this method of constructing number vectors encodes the sequence information associated with numerical concepts. The relationship of those conceptual properties to visual and motor representations is captured by the clean-up-like associative

¹A unitary vector is one which maintains a length of 1 when convolved with itself.

memory introduced earlier that maps these vectors onto their visual and motor counter-parts (and vice versa). As a result, visual semantic pointers are associated with conceptual semantic pointers, which are conceptually inter-related through recursive binding. This representation serves to capture both the conceptual semantic space and the visual semantic space, and link them appropriately. I should note that these are the representations of numbers used in all previous tasks as well.

The input presented to the model for the counting task consists of a starting number and a number of positions to count, e.g. ‘2 3 ?’. The model then generates its response, e.g. ‘5’. This kind of silent counting could also be thought of as an ‘adding’ task. In any case, given the representations described above, counting is a matter of recognizing the input and then having the basal ganglia bind the base unitary vector with that input the given number of times to generate the final number in the sequence. Given a functioning recognition system, working memory, and action selection mechanism, the task is relatively straight forward.

- Figure showing input and counting (errors?)
- Summary figure...? error rates at different starting points; output at different starting points? not sure what... showing the time it takes to count as a function of the number of positions?

Successful counting demonstrates the flexible action selection that is a central part of Spaun’s behavior. It also shows that the model has an understanding of order relations over numbers, and can exploit that knowledge to produce appropriate responses.

7.3.6 Question answering

The question answering task demonstrates that Spaun has access to information that it encodes in its internal representations. To allow the input to be more flexible, I introduce four special characters. These are the ‘[’ and ‘]’ brackets for grouping the input, and the letters ‘K’ (kind) and ‘P’ (position) for indicating what kind of question is being asked. For example, we might present ‘[1 1 2 3] P [2]’, which would be asking: What position is a two at? The answer in this case should be ‘3’. In contrast, ‘[1 1 2 3] K [2]’ would be asking: What kind of object is at position two? In which case, the answer should be ‘1’.

This task again combines many of Spaun’s abilities, with a particular emphasis on unbinding and clean-up. The recognition system is now fully exploited, as all

of the possible inputs must be recognizable in this task context. After recognition, the items are sent to the serial working memory for encoding into an internal representation. Basal ganglia controls the processing, based on currently available visual input, and so knows that input after ‘]’ indicates the kind of question to be asked. This results in setting up the appropriate routing within cortex for applying the next number input to the memorized representation appropriately.

- couple example runs. (with errors?)
- percent correct for each kind of question as the length of the input varies?
- Perhaps also as a function of how many of one kind there are?

The ability of Spaun to answer questions about structured input shows that it has various kinds of access to the information it encodes.

7.3.7 Rapid variable creation

In a recent paper, Bob Hadley argues forcefully that variable binding in connectionist approaches remains an unsolved problem (Hadley, 2009). His arguments crystallize a traditional concern expressed previously by Jackendoff (2002) and Marcus (2001). While Hadley acknowledges that many solutions have been proposed, he highlights that *rapid* variable creation and deployment, in particular, is inconsistent with all current proposals that are not simply neural implementations of a classical solution. However, rapid variable deployment is deftly performed by humans. Consider a central example from Hadley:

Training Set

Input: *Biffle biffle rose zarple*. Output: *rose zarple*.
Input: *Biffle biffle frog zarple*. Output: *frog zarple*.
Input: *Biffle biffle dog zarple*. Output: *dog zarple*.

Test Case

Input: *Biffle biffle quoggie zarple*. Output: ?

Hadley suggests that this task requires rapid variable creation because the second last item in the list can take on any form, but human cognizers can nevertheless

identify the overall syntactic structure and identify ‘*quoggie zarple*’ as the appropriate response. So it seems that a variable has been created, which can receive any particular content, and that will not disrupt generalization performance.

Hadley argues that learning rules cannot be used to solve this problem, given the known constraints on the speed of biological learning. Weight changes do not happen quickly enough to explain how these brief inputs can be successfully generalized within seconds. He also argues that approaches such as neural black-board architectures (NBAs; see section 9.1.3) would need to pre-wire the appropriate syntactic structures. Finally, he suggests as well that VSAs cannot solve this problem. This is because, as he correctly notes, the binding operation in VSAs will not provide a syntactic generalization. However, the SPA has additional resources, most importantly it has inductive mechanisms employed both in the Raven’s example (see section 4.6), and the Wason example (section 6.6). The Raven’s example, in particular, did not rely on changing neural connection weights to perform inductive inference. As a result, we can use this same mechanism to perform the rapid variable creation task.

To test rapid variable creation, we can employ the same representations used in the previous task. For example, an input might be ‘[1 1 2 3] [2 3] [1 1 5 3] [5 3] [1 1 6 3] [6 3] [1 1 1 3] ?’. The expected response in this case is ‘1 3’. The parallel to the previous example with words should be clear. The first item is a constant repeated twice, the third item is a variable, and the last item is a constant. Spaun is able to solve this task by performing induction on the presented input/output pairs to infer the transformation rule that is being applied to the input, just as in the Raven’s task.

- figures showing responses to various inputs; including tasks that are more difficult than those considered by Badley, i.e. multiple variables.
- summary figure of quality of inference on different kinds of strings? two variables, etc?

Spaun’s ability to successfully perform the rapid variable creation task demonstrates that the non-classical assumptions behind the model do not prevent it from reproducing this important cognitive behavior. In other words, it shows that biologically realistic, connectionist-like approaches can solve this outstanding problem.

7.3.8 Fluid reasoning

The final task I will consider is fluid reasoning. The purpose of including this task is to demonstrate the ability of the model to perform an unequivocally cognitive task that has been behaviorally well-characterized (see section 4.6). In many ways, there are no new aspects of the architecture to introduce at this point. The main difference between this example and the fluid reasoning task considered earlier, is that this version is fully controlled by the basal ganglia model.

The input to the system is in the form of a typical Raven's matrix, though it is shown serially. For example, a simple case is '[1] [1 1] [1 1 1] [2] [2 2] [2 2 2] [3] [3 3] ?'. Where the expected output is '3 3 3'. The main difference between this task and that in the previous section is the structure of the input. Specifically, induction is only driven by pairwise comparisons within rows, so the middle input sequence is employed twice: once as post-transformation and once as pre-transformation.

- Several examples showing different complexities of input matrices on these kinds of counting tasks...
- Percentage correct over some set of input matrices?

Successfully inducing implicit rules in a fluid reasoning task shows that Spaun captures central features of human cognition and intelligence.

7.3.9 Discussion

While the above set of tasks are defined over limited input and a simple semantic space, I believe they demonstrate the unique potential of the SPA to provide a general method for understanding biological cognition. Notably, it is straightforward to add other tasks into the Spaun architecture that can be defined over structured representations in the same semantic space. For instance, including additional forms of recall, backwards counting, *n*-back tasks, pattern induction, and so on would result from minor additions to the state-action mappings in the basal ganglia.

More importantly, the *methods* used to construct this simple model are not restricted to this particular semantic or conceptual space, and hence increases in the complexity of the input structure, conceptual structure, and output modality are direct extensions of the same principles used in this simple case. Overall, the purpose of Spaun is to show how general purpose SPA models can be generated using the principles described throughout the book.

In many ways, looking at the specific performance of the model on various distinct tasks, as I have done to this point, does not highlight what I take to be the main lessons we can learn from this model. For one, Spaun is *unified*. Thus the performance on each task is less important than the fact that all tasks were performed by the exact same model. No parameters were changed or parts of the model ‘rewired’ as the tasks varied. The representations used throughout the model are the same, as are the underlying computational principles, and methods of mapping to neural spikes. As a result, this single model has neuron responses in visual areas (e.g. V1 and IT) that match known visual responses, *as well as* neuron responses and circuitry in basal ganglia that match known responses and anatomical properties of basal ganglia, *as well as* behaviorally accurate working memory limitations, *as well as*, the ability to perform human like induction, and so on. Spaun is thus both physically and conceptually unified.

Spaun is also reasonably flexible. It can adapt to environmental rewards. It will reason differently given different input. It can learn new rules it did not previously know. It can use the same input ‘channel’ to identify tasks, identify input, and identify queries.

Spaun is also robust. There is a lot of intrinsic noise in the model because of the use of spiking neurons. We can destroy many of the cells and observe a graceful degradation in performance (analogous to the Wason model in section 6.6). We can change the timing of the input, or show unexpected input strings, and the system continues to function (see figure ???). It is robust to many variations in the hand-writing used to drive the system. ???Robustness of arm to perturbations while writing???. It continues to function when it makes errors in recall, induction, recognition and so on.

- figure showing the effects of asking for answers at weird times, interrupting the task and going on to a new one, etc.?

Of course, Spaun is also a meagre beginning. There are many more ways it could be unified, flexible, and robust. I discuss several of these in section 10, which discusses challenges for the SPA. Nevertheless, it is a reasonable beginning and, as I explore in subsequent chapters, I believe it has several advantages over past approaches. However, one of its advantages is somewhat subtle, and does not fit naturally into discussions of current approaches to cognitive modelling. So, I consider it next.

7.4 A unified view: Symbols and probabilities

As outlined in chapter 1, one traditional means of distinguishing symbolicist and connectionist approaches is that the former is taken to be most appropriate for describing high-level planning and rule following, and the latter is taken to be most appropriate for capturing lower-level pattern recognition and statistical kinds of behavior. These strengths are typically attributed to the reliance on symbols and symbolic structures by classical approaches, and the reliance on learned distributed representations by connectionist approaches. In this section, I want to consider how semantic pointers can be simultaneously interpreted as symbols and probabilities.

To help make this point, it is useful to contrast the SPA with a more symbolic approach. As I will discuss in some detail in section 9.1.1, the ACT-R cognitive architecture is perhaps the most successful and influential cognitive modeling architecture available. Many aspects of the SPA have been inspired by comparisons and contrasts with ACT-R. For instance, both ACT-R and the SPA characterize the basal ganglia as the locus of action selection, employ multiple cortical working memories, and so on. The most obvious difference between ACT-R and the SPA is that all representations in the SPA are distributed vector representations, whereas most representations in ACT-R are symbolic. So, while I have often characterized the SPA as manipulating symbol-like representations for ease of exposition, a probabilistic view also provides a consistent – and ultimately more computationally powerful – interpretation of the SPA. In particular, instead of thinking of semantic pointers as symbol-like, we can think of them as being probability distribution-like.

To give a sense of why probability distributions can be more computationally powerful, notice that a distribution more completely describes the possible states of the world than a symbol. Typically, a symbol is taken to have no special internal structure. Its semantics are determined by relations to other symbols (e.g., Harman, 1982), or in some other fashion (e.g., Fodor, 1998). In contrast, probability distributions, like any vector, can be treated in ways that draw on their internal structure. Let us consider an example.

Suppose we have the symbol ‘dog’, and we also have a probability distribution that represents the class of ‘dog’, $\rho(g)$. We can define a rule like “If something is a dog, then it is furry”. In a symbolic system, like ACT-R, the presence of the ‘dog’ symbol will result in the presence of the ‘furry’ symbol given this rule. There is only one ‘dog’ symbol and it leads to only one ‘furry’ symbol (unless we add rules). In contrast, in a probabilistically driven system, an inference rule will

relate *distributions*, which can take on many values. For example:

$$\rho(f) = \int_g \rho(f|g)\rho(g) dg$$

provides a means of determining the probability that something is furry, $\rho(f)$, knowing the probability it is a dog and how those distributions relate, $\rho(f|g)$. The internal structure of the resulting ‘furry’ distribution will depend on the internal structure of the ‘dog’ distribution. In fact, only in the special case where $\rho(f|g)$ is equal to a kind of identity function, will this rule be equivalent to the symbolic one. Consequently, the variety of rules defined over probability distributions is much greater than that defined over symbols; for probability distributions, rules may or may not depend on the details of the structure of that distribution.²

In fact, this increase in computational power has been recognized, and explored by the ACT-R community. They have worried about ‘partial matching’ of productions (if-then rules) to input symbols in the past (Lebiere and Anderson, 1993). However, the vast majority of ACT-R architectures, including the most recent official one (i.e., ACT-R 6) does not employ this technique. The SPA, in contrast, includes partial matching as a central kind of computation.

Recall that in the SPA there is a mapping from cortex to striatum, which I have discussed as determining the ‘if’ part of an ‘if-then’ rule (see section 5.4). This mapping is in fact a partial matching operation. As mentioned in my earlier description, if we think of the map from cortex to striatum as a matrix \mathbf{M}_b , and the cortical state as a vector \mathbf{x} , then the result of $\mathbf{M}_b\mathbf{x}$ is a vector in which each element is a value that represents how well \mathbf{x} matches the ‘if’ part of each rule represented in \mathbf{M}_b . That is, the ‘degree of match’ is exactly the result of the projection of cortex into striatum in the SPA. Continuing through the basal ganglia, that result is then used, via a winner-take-all-like operation, to determine the appropriate action, which is then mapped to a new cortical state \mathbf{x}' . This process is depicted in figure 7.5a.

So, under a symbolic view of the SPA, we might say that the semantic pointer (qua symbol) represented in cortex is used to match a rule in the basal ganglia, resulting in the selection of the action associated with that rule. Under a more

²I should point out that strictly speaking this is not a mathematical fact, but rather a conceptual one. That is, it depends on what we take to be basic cognitive states more than on our chosen formalism. If someone decides to suggest that each dimension in a semantic pointer is a many-stated symbol, then we could say the SPA is a symbolic architecture. In such a case, there is no interesting difference in the set of computations we can describe using either formalism. However, such a claim does violence to the typical use of ‘symbolic’ in the behavioral sciences.

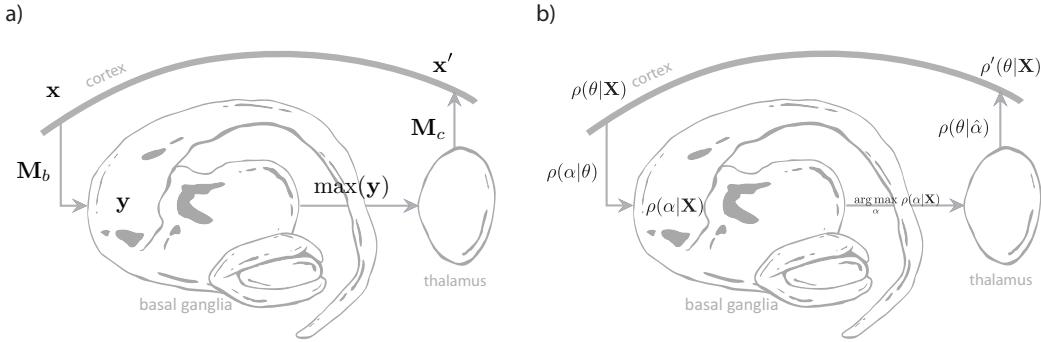


Figure 7.5: Two SPA interpretations of the cortical-basal ganglia-thalamus loop. a) The matrix-vector characterization of basal ganglia, that I have previously discussed in terms of implementing ‘if-then’ rules (see section 5.6). b) An equivalent probabilistic characterization of basal ganglia function. This characterization helps identify how the SPA goes beyond traditional production systems, and hence unifies symbolic and connectionist approaches to cognitive modelling. See text for details.

vector-based view of the SPA, we would say that the semantic pointer (qua vector) is measured against familiar states in the basal ganglia, resulting in the selection of a new cortical state most appropriate under the given measure.

Interestingly, because the SPA employs high-dimensional vectors as its central representations, this exact same process has a very natural statistical interpretation, as shown in figure 7.5b. Mathematically, all objects (scalars, functions, fields, etc.) can be treated as vectors. So, we could just simply stipulate that semantic pointer vectors are probability distributions. But, this does not provide us with an actual interpretation, mapping specific distributions onto brain states. Consequently, we must be careful about our probabilistic interpretation, ensuring the already identified neural computations are still employed, and that the distributions themselves make biological sense.

Perhaps surprisingly, the interpretation provided here lets us understand the cortical-basal ganglia-thalamus loop as implementing a standard kind of Bayesian inference algorithm known as Empirical Bayes. Specifically, this algorithm relates perceptual observations (X), parameters of a statistical model of that input (θ), and so-called “hyperparameters” (α) that help pick out an appropriate set of model parameters. Essentially, identification of the hyperparameters allow the brain to determine a posterior distribution on the model parameters based on the observation. We can think of this distribution as acting like a prior when relating

observations and model parameters.

The basic idea behind this interpretation is that cortex takes sensory measurements and attempts to determine a model (or “explanation”) for those measurements. In short, it infers θ given \mathbf{X} , using past experience that helps determine the form of various priors. However, this inference is approximate and difficult, so it employs the basal ganglia to iteratively improve the estimate of that distribution using learned relations between environmental regularities. The resulting distribution can then be used by the motor system to generate samples that are consistent with the model, or it can be used to predict future states, or help interpret new sensory measurements.

To be specific, let me go through the steps depicted in figure 7.5b. Before determining the initial distribution in the diagram, $\rho(\theta|\mathbf{X})$, we begin with a likelihood over model parameters, $\rho(\mathbf{X}|\theta)$, where \mathbf{X} are the observations, and cortex has learned how to generate an initial guess of the probability distribution for those observations given various values of the parameters θ .³ Cortex initially estimates a distribution of how likely various parameter settings are given the observations. Typically, this would be done using Bayes’ rule:⁴

$$\rho(\theta|\mathbf{X}) \propto \rho(\mathbf{X}|\theta)\rho(\theta).$$

However, the difficulty of computing this distribution exactly is precisely why we need to employ the basal ganglia. Consequently, the first pass through cortex will only provide some rough estimate of this distribution. This provides a starting point analogous to default routing being used to generate the initial cortical state \mathbf{x} in the matrix-vector characterization.

The mapping into the striatum from cortex is defined by $\rho(\alpha|\theta)$ which encodes the learned relationship between the hyperparameters and the parameter values. In virtue of mapping through this distribution, the result in striatum relates the observations directly to these hyperparameters:⁵

$$\rho(\alpha|\mathbf{X}) = \int_{\theta} \rho(\alpha|\theta)\rho(\theta|\mathbf{X}) d\theta,$$

which is analogous to \mathbf{y} in the matrix-vector interpretation.

³To interpret a likelihood, you can imagine that the observations \mathbf{X} are fixed and that the probability varies as the parameters, θ , are changed.

⁴As is common, I am ignoring the denominator on the right hand side, $\rho(\mathbf{X})$, both because it is a constant, and because it is very difficult to compute. In short, it does not change the answer we compute except by a scaling factor, which is typically not important.

⁵I am assuming α and \mathbf{X} are conditionally independent given θ .

Now, as before, a maximization operation is assumed to be performed by the rest of the basal ganglia circuit. In inference, this is known as maximum a posteriori (MAP) inference, and is used to determine the specific value of α (i.e., $\hat{\alpha}$) that maximizes the original distribution $\rho(\alpha|\mathbf{X})$.

This result is then sent through thalamus, and back to cortex, where the thalamic-cortical mapping is used to determine a new distribution of the parameters θ given the inferred value of $\hat{\alpha}$, i.e., $\rho(\theta|\hat{\alpha})$. Once back in cortex, this distribution can be used to update the estimate of the distribution of the parameters given the data:

$$\rho'(\theta|\mathbf{X}) \propto \rho(\mathbf{X}|\theta)\rho(\theta|\hat{\alpha}),$$

which completes the loop.

Typically in such inference algorithms, the loop is performed several times to improve the quality of the estimate of $\rho(\theta|\mathbf{X})$, as it is improved on each iteration. This need not be the case, however, if time is of the essence. Furthermore, this distribution is very useful because it can be used to support many further inferences. For example, it can be used to determine how likely a new observation x_{new} is given past observations:

$$\rho(x_{new}|\mathbf{X}) = \int_{\theta} \rho(x_{new}|\theta)\rho(\theta|\mathbf{X}) d\theta.$$

Or, it can be used to drive motor action:

$$\rho(c|\mathbf{X}) = \int_{\theta} \rho(c|\theta)\rho(\theta|\mathbf{X}) d\theta$$

where c is the motor command to execute, and so $\rho(c|\theta)$ would encode the previously learned mapping between a model of the world and motor commands. This characterization preserves a natural role for the basal ganglia in action selection. More generally, the probabilistic interpretation of the basal ganglia processing loop as an iterative inference process is consistent with all aspects of my previous discussions of the basal ganglia.⁶

⁶Three brief technical points for those who might be concerned that a probabilistic characterization assumes very different cortical transformations than before. First, the convolution of two probability distributions is the distribution of the sum of the original variables, so the previous binding operation is well characterized probabilistically. Second, the superposition and binding operations do not significantly alter the length of the processed variables. Specifically, the expected length of two unit length bound variables is one, and the superposition is well-normalized by the saturation of neurons. Third, the multiplication and integration operation I have used several times is identical to matrix-vector multiplication.

Let me step away from the formalism for a moment to reiterate what I take to be the point of this exercise in re-interpretation of the cortex-basal ganglia-thalamus loop. Recall that a central distinction between symbolicism and connectionism is that the former has strengths in language-like rule processing and the latter has strengths in probabilistic inference. I have spent most of the book showing how semantic pointers can be understood as implementing language-like rule processing, and am here reiterating that they can also be understood as implementing statistical inference (I had suggested this only for perception and motor control earlier, in chapter 3). As a consequence, I believe that these two consistent and simultaneously applicable interpretations of the SPA highlight a fundamental unification of symbolic and connectionist approaches in this architecture. And, it begins to suggest that the SPA might be in a position to combine the strengths of past approaches. Furthermore, this kind of unification is not available to approaches, like ACT-R, which do not preserve the internal structure of the representations employed.

This, however, is only one preliminary comparison of the SPA to past work. In the second part of the book, I provide a more thorough comparison.

7.5 Nengo: Large-scale modeling

In the tutorial from section 5.9, we built a network with thousands of spiking neurons that incorporates structured representations to describe statements and questions, transformations to manipulate these semantic pointer and control information routing, and dynamics to build an integrator that performed the function of working memory. Despite the breadth of this network, it is still a small and specialized subsystem in comparison to many of the models presented in this book which are themselves currently restricted to narrow domains and pale in comparison to biological brains. This tutorial will not introduce a more complicated network – the building blocks for developing integrated models have already been introduced – but it will introduce new methods for efficiently constructing models.

The ability to generate networks in Nengo using scripts is vital for large projects. The scripting interface can be used to automate every interaction with Nengo that could be executed with the graphical interface. It can also be used to extend Nengo’s capabilities to work with third-party software or your own libraries. This tutorial will cover the scripting library used to build large semantic pointer architecture models. For Nengo scripting tutorials that cover lower-level utilities such as creating individual ensembles with customized tuning curves or integrat-

ing Nengo with Matlab, please see the online tutorials at <http://compneuro.uwaterloo.ca/cnrglab/?q=node/2>.

- Open a blank Nengo workspace and press **Ctrl+P** to open the scripting console.
- Type ‘run demo/spa_sequence.py’.
- Open the network in the *Interactive Plots* viewer.
- Run the simulation.

The ‘Utility’ visualization shows the similarity measure resulting from a comparison of the input to the basal ganglia with the vocabulary vectors. The ‘Rules’ visualization shows which output component of the basal ganglia is active at any given time. The network in this demo script cycles through the semantic pointers ‘A’ through ‘E’ in alphabetical order before looping back to ‘A’. This sequential rehearsal network may be familiar from section 5.4, where it was first introduced. Recall that basal ganglia activation indicates which rule’s input condition most closely matches the inputs at a given time. The thalamus then implements the actions specified by the active rule.

- Find and open Nengo’s install directory. This is the directory where you unzipped Nengo’s files in the first tutorial.
- Open the ‘demo’ folder, located in Nengo’s install directory.

The files located in the demo folder are all Python scripts. They can be opened with any text editor, although specialized editors are also freely available on the internet.

- Open the file ‘spa_sequence.py’ in an editor.

The file you’ve just opened contains the instructions required to build the ‘Sequence’ network. These instructions should match the listing given below.

```

1 from spa import *
2 D=16
3
4 class Rules:
5     def A(state='A'):
6         set(state='B')

```

```

7     def B(state='B'):
8         set(state='C')
9     def C(state='C'):
10        set(state='D')
11    def D(state='D'):
12        set(state='E')
13    def E(state='E'):
14        set(state='A')
15
16 class Sequence(SPA):
17     dimensions=16
18     state=Buffer()
19     BG=BasalGanglia(Rules())
20     thal=Thalamus(BG)
21     input=Input(0.1, state='D')
22
23 seq=Sequence()

```

The first line of the file ties the demo script to other script files that handle interactions with Nengo and enable basic scripting functions. The second line uses the variable *D* to store the number of dimensions we'd like each ensemble in the network to represent. The next block of code defines the ‘Rules’ class which sets the rules that the basal ganglia and thalamus will instantiate. The rule definitions all follow the following format:

```

def RuleName(inputState='InputSP'):
    set(outputState='OutputSP')

```

RuleName sets the name of the rule that will appear in the interactive graphs, *InputSP* is the name of a semantic pointer in the vocabulary that an input from the ensemble named *inputState* must match for the rule to fire, and *OutputSP* is the name of a semantic pointer in the vocabulary that is sent to the ensemble *outputState* when the rule fires. Note that in the spa_sequence.py demo, both *inputState* and *outputState* are set to the same ensemble, which is simply named ‘state’. The rules given in the demo specify a chain of rules that loop through the states; when ‘A’ is the dominant state, ‘B’ is sent as an input, when ‘B’ is the dominant state, ‘C’ is sent, and so on.

The ‘Sequence’ class which is defined below the ‘Rules’ class declares four objects: a buffer, a basal ganglia component, a thalamus component, and an input. The buffer is an integrator that serves as a generic memory component and it

is assigned to a variable named ‘state’. The basal ganglia and thalamus act as previously described, but note that the class ‘Rules’ is invoked within the brackets of the basal ganglia constructor on line 19 of the listing and the name of the basal ganglia variable is given on the following line when the thalamus is created. These steps are required to create basal ganglia and thalamus components that conform to the specified rules. The last object is an input function. The syntax for creating an input function is as follows:

```
Input( duration , target='SP' )
```

The first parameter, *duration*, specifies how many seconds the input will be presented for, the parameter *target* gives the population that will receive input from the function, and *SP* is the name of the semantic pointer that will be given as input. The four objects created in the ‘Sequence’ class are assigned to variables named ‘state’, ‘BG’, ‘thal’, and ‘input’ and these variable names are used to name the items created in the Nengo workspace. The final line of the script runs the ‘Sequence’ class, which generates all the network objects described above and connects them together to form the ‘Sequence’ network.

Since we have a script that will generate the network automatically, it is easy to make adjustments to the network and test them quickly. For example, we can replicate the result shown in figure 5.7 of section 5.4 in which the sequence is interrupted by constant input.

- Save the ‘spa_sequence.py’ file as ‘spa_tutorial.py’. This is to avoid over-writing the demo script accidentally.
- Edit the line declaring the input object to read ‘input=Input(10, state ='A’)’. This is line 20 of the program listing above.
- Save the ‘spa_tutorial.py’ file.
- Return to a blank Nengo workspace and open the scripting console.
- Type ‘run demo/spa_tutorial.py’ and press enter.
- Open the network with *Interactive Plots* and run the simulation.

The input to the state buffer is now presented for ten seconds instead of the previous 0.1 seconds. As reported in chapter 5, this prevents the activation of the second rule in the sequence despite the high utility of the rule because the input value drives the network back to its initial state. The solution described in the

earlier chapter was to introduce routing to control the flow of information. We can make this change by adding three lines to the script.

- Open the file ‘spa_sequencerouted.py’ located in the demo folder.

The ‘spa_sequencerouted.py’ file is identical to the ‘spa_sequence.py’ file with the addition of the changes required to introduce routing. The first change that was made the introduction of a new rule named ‘start’. Unlike the other rules, the ‘start’ rule doesn’t assign a fixed semantic pointer to the ‘state’ ensemble when the rule fires; it uses the value stored in the ‘vision’ ensemble instead. The ‘vision’ ensemble is itself a new addition, but it’s just a regular ensemble. The only other change is to set the initial input to contain the ‘LETTER’ semantic pointer, which will cause the start rule to copy the input from the ‘vision’ ensemble to the ‘state’ ensemble. The utility of the ‘start’ rule is lower than the utility of the sequence rules, so it will only transfer information from the visual area when no other action applies.

- Return to a blank Nengo workspace and open the scripting console.
- Type ‘run demo/spa_sequencerouted.py’ and press enter.
- Open the ‘Routing’ network in the network viewer and double click the ‘thal’ network to view the thalamus network.

The thalamus network in this scripting framework is similar to the thalamus network that we created with the drag-and-drop interface in the section 5.9 tutorial with a few modifications. The scripted thalamus includes the gating components required to implement routing as part of the network and also includes a recurrent connection on the ‘Rules’ ensemble which causes rules to mutually inhibit each other, reducing the likelihood of having multiple rules fire simultaneously.

- Close the ‘thal’ network and run the ‘Routing’ network in the *Interactive Plots* simulator.

With routing, the network ignores inputs from the visual area after the initial ‘start’ rule is completed. We’ve seen networks involving basal ganglia control several times throughout this book, including once in a previous tutorial, so this is likely not a stunning result. What’s new about this network is that it is described within a couple dozen short lines in a script file that can easily be extended to more complex models. Adding more rules and buffers, these scripting techniques can be used to implement the Tower of Hanoi model and they lay a foundation that can be used in comprehensive models such as Spaun as well. This concludes the Nengo tutorials for this book. The next step is to build your own models!

Part II

Is that how you build a brain?

Chapter 8

Evaluating cognitive theories

8.1 Introduction

To this point, I have completed my description of the functional and methodological ideas behind the Semantic Pointer Architecture. Along the way, I have mentioned several unique or particularly crucial features of the SPA and provided a variety of examples to highlight these attributes. However, I have not had much opportunity to contextualize the SPA in the terrain of cognitive theories more generally. This is the task I turn to in the next three chapters.

As you may recall from chapter 1, cognitive science has been dominated by three main views of cognitive function: symbolism, connectionism, and dynamicism. In describing and comparing these views, I argued for the surprising finding that despite occasionally harsh exchanges between proponents of these views, there is a core consensus on criteria for identifying cognitive systems. I dubbed this core the Quintessential Cognitive Criteria (QCC; see table 8.1), and briefly enumerated them at the end of section 1.3. At the time, I made no attempt to justify this particular choice of criteria, so I could turn directly to the presentation of the SPA.

In this chapter, I return to the QCC to make it clear why each is crucial to evaluating cognitive theories. In the next chapter, I briefly describe several other architectures for constructing cognitive models to both introduce the state-of-the-art, and to provide an appropriate context for evaluating the SPA. This allows me to highlight similarities and differences between the SPA and past approaches. In the final chapter, I briefly address several ramifications of the SPA regarding central cognitive notions including ‘representation’, ‘dynamics’, ‘inference’, and

Table 8.1: A reproduction of table 1.1 of the Quintessential Cognitive Criteria (QCC) for theories of cognition.

1. Representational structure
a. Systematicity
b. Compositionality
c. Productivity (the problem of variables)
d. The massive binding problem (the problem of two)
2. Performance concerns
a. Syntactic generalization
b. Robustness
c. Adaptability
d. Memory
3. Scientific merit
a. Triangulation (Contact with more sources of data)
b. Compactness

‘concept’.

8.2 Quintessential cognitive criteria (QCC)

The QCC are ‘quintessential’ in the sense that they are my attempt to distill the most typical sorts of criteria that researchers employ in deciding whether a system is cognitive. However, I also use them to evaluate cognitive architectures. That is, if a proposed architecture is likely to produce and explain systems that can satisfy these criteria, I consider it a good architecture. While I have said little about what it takes to be a cognitive architecture *per se*, I believe I have cast the net broadly enough to satisfy reasonable definitions from proponents of any of the three standard views (for discussions and history of the notion of a ‘cognitive architecture’ see Thagard (2011) or Anderson (2007)).

Note that in the remainder of this section, I rely heavily on the terminology introduced in chapter 1. Consequently, it may prove helpful to review sections 1.2 and 1.3, to recall the distinguishing features, metaphors, and theoretical commitments of each of symbolism, connectionism, and dynamicism. For ease of reference, the QCC themselves are summarized in table 8.1.

8.2.1 Representational structure

Historically, theories of cognition have been evaluated using criteria that focus on representational structure. Perhaps this is because representational commitments can be used to easily distinguish between dynamicism, connectionism, and symbolism. As a consequence, the following discussion is closely related to the ongoing debates amongst these views.

8.2.1.1 *Systematicity*

The fact that cognition is systematic – that there is a necessary connection between some thoughts and others – has been recognized in several different ways. Gareth Evans (1982), for instance, identified what he called the Generality Constraint. In short, this is the idea that if we can ascribe a property to an object, then we can ascribe that same property to other objects, and other properties to the original object (e.g., anything ‘left of’ something can also be ‘right of’ something). For Fodor and Pylyshyn (1988a), the same observation is captured by their systematicity argument. This argument is, in brief, that any representational capacities we ascribe to a cognitive agent must be able to explain why our thoughts are systematic. They must explain, in other words, why if we can think the thought “dogs chase cats”, then we can necessarily also think the thought that “cats chase dogs”. They argue that a syntactically and semantically combinatorial language is able to satisfy this constraint. A combinatorial language is one which constructs sentences like “dogs chase cats” by combining atomic symbols like “dogs”, “chase”, and “cats”. As a result, if one of those symbols is removed from the language, a whole range of sentence-level representations are systematically affected. For instance, if we remove the symbol “cats” then we can think neither the thought “dogs chase cats” nor the thought that “cats chase dogs”.

While many non-classicist researchers disagree with the assumption that only a combinatorial language with atomic symbols can explain human systematicity, the observation that our thoughts are systematic is widely accepted. This is likely because systematicity is so clearly evident in human natural language. As a result, whatever representational commitments our cognitive architecture has, it must be able to describe systems that are appropriately systematic.

8.2.1.2 *Compositionality*

If a representation is compositional, then the meaning of that representation is determined by its structure and the meaning of its constituents. Most formal languages respect this constraint. So, if I have a word like “fish” which takes a set of objects in the world, and I have a word like “pet” which determines a relation between an owner and an animal, then a combination like “pet fish” would indicate the “fish” objects that lie in the “pet” relation to an owner. So, the meaning of the combination is a simple function of the two constituents. At first glance, this may seem to reflect semantics of natural languages.

However, there is good evidence that concepts in natural language often do not combine compositionally. To return to the previous example, our notion of a “pet fish” does not, in fact, seem to be a simple function of our notions of a “pet” and a “fish.” For instance, even though a prototypical pet is a dog, and a prototypical fish is a bass, a prototypical pet fish turns out to be a goldfish – which has no obvious semantic connection to either a dog or a bass (Osherson and Smith, 1981). One might want to argue that “logically speaking” the locution “pet fish” simply identifies the class of fish that are pets. However, the goal of cognitive systems research is to understand the best examples of cognitive systems that we have, not to pre-specify the way we think such systems *should* work.

Specifying semantic constraints on cognitive systems in advance ignores the often contingent nature of natural language semantics – a semantics that forms the basis of the flexibility of our behavior. One simple example of this contingency can be found in the dictionary definitions of “ravel” and “unravel.” They turn out to be the same. This, of course, is surprising since the prefix “un-” usually reverses the meaning of a root. In short, the mapping between our combinatorial syntactic languages and semantics seems to be much more complex and contingent than is allowed for by simple compositionality. The way in which semantic information is combined, or even if it is combined, in the face of different syntactic structures remains largely mysterious. As a consequence, most cognitive systems researchers reject natural language as being compositional, in anything like the way a formal language is.

So, understanding how words are composed – semantically as well as syntactically – to create complex representations is crucial to understanding cognitive systems. It is just that simple compositionality, as originally proposed by Fodor and Pylyshyn, will not always do. As a result, explaining observed compositionality effects remains an important criteria for a cognitive theory to address. Real

cognitive systems sometimes draw on a wealth of experience about how the world works when interpreting compositional structures. Although we have much to learn about human semantic processing, what we do know suggests that such processing can involve most of cortex (Aziz-Zadeh and Damasio, 2008), and can rely on information about an object's typical real-world spatial, temporal, and relational properties (Barsalou, 2009).

I want to be clear about what I am arguing here: my contention is simply that ideal compositionality is not always satisfied by cognitive systems. There are, of course, many cases where the semantics of a locution, e.g. "brown cow", is best explained by the simple, idealized notion of compositionality suggested by Fodor and Pylyshyn. The problem is that the idealization misses a lot of data (such as the examples provided). Just how many examples are missed can be debated, but in the end an understanding of compositionality that misses fewer should clearly be preferred.

Thus, determining how well an architecture can define systems that meet the (non-idealized) compositionality criteria will be closely related to determining how the architecture describes the processing of novel, complex representations. We should be impressed by an architecture which defines systems that can provide appropriate interpretations that draw on sophisticated models of how the world works – but only when necessary. That same theory needs to capture the simple cases as well. No doubt, this is one of the more challenging, yet important, criteria for any architecture to address.

In sum, compositionality is clearly important, but it is not only the simple compositionality attributed to formal languages that we must capture with our cognitive models. Instead, the subtle, complex compositionality displayed by real cognitive systems must be accounted for. The complexity of compositionality thus comes in degrees, none of which should be idealized away. An architecture that provides a unified description spanning the observed degrees of compositional complexity will satisfy this criteria best.

8.2.1.3 *Productivity*

Ideally, productivity is the ability of a language to generate an infinite variety of valid sentences with a finite set of words and a finite grammar. In many ways, it was precisely this property of language that Chomsky (1959) relied on so effectively to critique behaviorism – a critique largely seen as a major turning point in the cognitive revolution. Fodor and Pylyshyn identify productivity as a

third central feature of cognition. Similarly, Jackendoff (2002) identifies the third challenge for cognitive theories as one of explaining the “problem of variables”. This is the problem of having grammatical templates, in which any of a variety of words can play a valid role. Typically, these variables are constrained to certain classes of words (e.g. noun phrases, verbs, etc.). Despite such constraints, there remains a huge, possibly infinite, combination of possibly valid fillers in some such templates. As a result, Jackendoff sees the existence of these variables as giving rise to the observed productivity of natural language.

The claim that productivity is central for characterizing cognition has received widespread support. Productivity is, however, clearly an idealization of the performance of any actual cognitive system. No real cognitive system can truly realize an infinite variety of sentences. Typically, the productivity of the formalism used to describe a grammar outstrips the actual productivity of a realized system. For instance, there are sentences which are “grammatically” valid that are neither produced nor understood by speakers of natural language. Examples of such sentences can be generated using a grammatical construction called “recursive center embedding”. An easily understood sentence with one such embedding is “The dog the girl nuzzled chased the boy.” A difficult to understand example is “The dog the girl the boy bit nuzzled chased the boy”, which has two embeddings. As we continue to increase the number of embeddings, all meaning becomes lost to natural language speakers. The reason, of course, is that real systems have finite resources.

The two most obvious resource limitations are time and memory. These are, in fact, intimately linked. As we try to cram more things into memory, two things happen: we run out of space; and what is in memory is forgotten more quickly.¹ So, when we are trying to process a complex sentence in “real-time”, as we keep adding items to memory (such as embedded clauses), we eventually run out of resources and are not able to make sense of the input. Cognitive systems are limited in their capacity to produce and comprehend input.

In short, real-world cognitive systems have a *limited* form of productivity: one which allows for a high degree of representational flexibility, and supports the generation and comprehension of fairly complex syntactic structures. While cognitive systems are representationally powerful, perfect productivity is not a reasonable expectation, and hence is not a true constraint on cognitive architectures. It is,

¹I’m thinking here mainly of working memory, which is most relevant for processing information on the time scale of seconds. However, similar resource constraints exist for both briefer and more extended forms of memory.

as with compositionality, a matter of degree. Natural measures of the degree of productivity of a system include the depth, length and number of variables of a structure that can be successfully manipulated. Such measures capture both the sense that there is a limit on productivity, and acknowledges that a great deal of representational flexibility is provided by cognitive representations. A good cognitive architecture should provide resources that match both the power and limits of real productivity.

8.2.1.4 *The massive binding problem*

Jackendoff (2002) identifies as his first challenge for cognitive modelling “the massiveness of the binding problem”. He argues that the binding problem, well-known from the literature on perceptual binding, is much more severe in the case of cognitive representations. This is because to construct a complex syntactic structure, many “parts” must be combined to produce the “whole”. Jackendoff provides simple examples which demand the real-time generation of several types of structure, including phonological, syntactic, and semantic structure. He also notes that each of these structural parts must be inter-related, compounding the amount of binding that must be done. It is clear to Jackendoff that traditional accounts of perceptual binding (e.g. synchrony) have not been designed for this degree of structural complexity (even if, as he acknowledges, such complexity could be found in a complex visual scene).

Given that there is a massive amount of binding in cognitive systems, is not surprising that in some circumstances the same item may be bound more than once. Jackendoff identifies this as a separate challenge for connectionist implementations of cognition: “the problem of two”. Jackendoff’s example of a representation that highlights the problem of two is “The little star’s beside a big star” (p.5). Identifying this as a problem is inspired by the nature of past connectionist suggestions for how to represent language that are familiar to Jackendoff. Such suggestions typically ascribe the representation of a particular word, or concept, to a group of neurons. Thus, reasoned Jackendoff, if the same concept appears twice in a single sentence, it will not be possible to distinguish between those two occurrences, since both will be trying to activate the same group of neurons.

While the problem of two poses certain challenges to understanding cognition as implemented in neurons, it is not obviously separate from the more general considerations of systematicity and binding. After all, if an approach can bind a representation for “star” to two different sentential roles, it should solve this

problem. Consequently, I have placed the problem of two under this criterion, as it relates to “binding” more generally.

To summarize, Jackendoff sees the scalability of proposed solutions to binding as being essential to their plausibility. The degree to which a proposed architecture is cognitive is directly related to how well it can generate appropriately large structures, in the same length of time and with the same kind of complexity as observed in real cognitive systems. So, we can see the “massiveness of binding” challenge as emphasizing the practical difficulties involved in constructing a suitably productive system. This problem thus straddles the distinction I have made between representational concerns and performance concerns. Regardless of where it is placed on the list of cognitive criteria, it highlights that a good cognitive architecture needs to identify a binding mechanism that scales well.

8.2.2 Performance concerns

While criteria related to representational structure focus on the theoretical commitments that underwrite cognitive architectures, criteria related to performance concerns are directed towards actually implementing a cognitive system.

8.2.2.1 *Syntactic generalization*

Like the massive binding problem, syntactic generalization straddles representational and performance concerns. In short, this criterion identifies one of the most crucial kinds of performance we should expect from a system that satisfies the representational criteria. Namely, the ability to generalize based on the syntactic structure of representations. We have already encountered this notion in earlier example models from sections 4.6, 6.6, and 7.2.

Let us consider another example. Suppose you are told that you must figure out what “triggle” implies given the following information:

- “The square being red triggles the square being big” implies that the square is big and the square is not red; also
- “The dog being fuzzy triggles the dog being loud” implies that the dog is loud and the dog is not fuzzy.

Now I ask the question: What does “the chair being soft triggles the chair being flat” imply? If you figured out that this implies that the chair is flat and not soft,

you have just performed syntactic generalization. That is, based on the syntactic structure presented in the two examples, you determined what the appropriate transformation to that structure is in order to answer the question. Evidently, the problem of rapid variable creation (see section 7.3.7) is just a simpler form of syntactic generalization as well. While the emphasis in that case is on speed, the problem being solved is essentially one of identifying syntactic structure, and using it in a novel circumstance.

There are, in fact, even simpler examples of syntactic generalization, such as determining when contractions are appropriate (Maratsos and Kuczaj, 1976). However, the example I provided above shows the true power of syntactic generalization. That is, it demonstrates how reasoning can be sensitive to the structure, not the meaning, of the terms in a sentence. Taken to an extreme, this is the basis of modern logic, and it is also why computers can be used to perform reasonably sophisticated reasoning.

It should not be surprising, then, that if we adopt the representational commitments of computers, as symbolism does, syntactic generalization becomes straightforward. However, if we opt for different representational commitments, as do both dynamicism and connectionism, we must still be able to explain the kind of syntactic generalization observed in human behavior. In short, syntactic generalization acts as a test that your representational commitments do not lead you astray in your attempt to explain cognitive performance.

I should note here, as I did for several of the criteria in the previous section, that human behavior does not always conform to ideal syntactic generalization. As described in section 6.6, there are well-known effects of content on the reasoning strategies employed by people (Wason, 1966). In the preceding example, this would mean that people might reason differently in the “dog” case than in the “chair” case, despite the fact that they are syntactically identical. So again, we must be careful to evaluate theories with respect to this criterion insofar as it captures human performance, not with respect to the idealized characterization of the criterion. In short, if people syntactically generalize in a certain circumstance, a good cognitive architecture must be able to capture that behavior. If people do not syntactically generalize, the architecture must be able to capture that as well.

8.2.2.2 *Robustness*

Syntactic generalization deals with performance on a specific, though ubiqui-

tous sort of cognitive task. Robustness, in contrast, deals with changes of performance across many cognitive tasks. “Robustness” is a notion which most naturally finds a home in engineering. This is because when we build something, we want to design it such that it will continue to perform the desired function regardless of unforeseen circumstances or problems. Such problems might be changes in the nature of the material we used to construct the object (for instance the fatiguing of metal). Or, these problems might arise from the fact that the environment in which the object is built to perform is somewhat unpredictable. In either case, a robust system is one which can continue to function properly in spite of these kinds of problems, and without interference from the engineer.

In some respects, the hardware of modern digital computers is extremely robust. It has been specifically built such that each of the millions of transistors on a chip continues to perform properly after many millions of uses. This robustness stems partially from the fact that the voltage states in a transistor are interpreted to be only on or off, despite the fact that the voltage varies between ± 5 V. As a result, if an aging transistor no longer reaches 5 V, but is above zero, it will be acting properly. The cost of this robustness, however, is that these machines use a lot of power: the human brain consumes about 20 W of power; much less impressive digital computers use tens or hundreds of times more power.

Nevertheless, computers are well-designed to resist problems of the first kind – degradation of the system itself. In fact, these same design features help resist some of the second kind of problem – environmental variability. Specifically, electromagnetic fluctuation, or heat fluctuations, can be partially accommodated in virtue of the interpretation of the transistor states. However, “interesting” environmental variability is not accounted for by the hardware design at all. That is, the hardware cannot run a poorly written program. It cannot use past experience to place a “reasonable” interpretation on noisy or ambiguous input. This limitation is not solely the fault of the hardware designers. It is as much a consequence of how software languages have been designed and mapped onto the hardware states.

All of this is relevant to cognitive science because symbolism adopted an approach which shared these central design features used to construct and program digital computers. But, it seems that biology has made a different kind of trade-off than human engineers in building functional, flexible devices. Brains use components which, unlike transistors, are highly unreliable, often completely breaking down, but which use very little power. Consequently, the kinds of robustness we see in biological systems is not like that of computers. So, when connectionists adopted a more brain-like structure in their theoretical characteri-

zation of cognition, much was made of the improved robustness of their models over their symbolicist competitors. Indeed, robustness was not of obvious concern to a symbolicist, since the hardware and software on which their simulations ran (and whose theoretical assumptions they had taken on board) essentially hid such concerns from view.

However, robustness concerns could not be ignored for long when connectionists began explaining certain kinds of phenomena, such as the “graceful degradation” of function after damage to the brain, that seemed beyond the reach of symbolism (Plaut and Shallice, 1994). As well, adopting more brain-like architectures made it possible for connectionists to explain the performance of cognitive systems on tasks such as pattern completion, recognition of noisy and ambiguous input, and other kinds of statistical (as opposed to logical) inference problems.

But, connectionist models continued to make “un-biological” assumptions about the nature of the implementation hardware. Most obvious, perhaps, was that connectionist nodes were still largely noise free. Nevertheless, early connectionist models made the important point that matching the observed robustness of cognitive systems can be closely tied to the specific implementation architecture that they were built on. More generally, these models established that the robustness of cognitive systems could be used as a means to justify aspects of an underlying cognitive theory. In short, they established that robustness was a relevant criterion for characterizing cognitive architectures.

In sum, as a criterion for a good cognitive architecture, robustness demands that an architecture supports models that continue function appropriately given a variety of sources of variability, such as noisy or damaged component parts, imprecision in input, and unpredictability in the environment.

8.2.2.3 *Adaptability*

Adaptability has long been one of the most admired features of cognitive systems. It is, in many ways, what sets apart systems we consider “cognitive” from those that we do not. Simply put, adaptability is exhibited by a system when it can update its future performance on the basis of past experience. This, of course, sounds a lot like learning, and indeed the two terms can often be used interchangeably. However, as discussed earlier, in cognitive science (especially connectionism) learning often refers specifically to the changing of parameters in a model in response to input. Adaptability, however, goes beyond this definition. There are, for instance, nonlinear dynamical systems which can exhibit adaptable behavior

without changing any of the parameters in the system (the fluid intelligence model in section 4.6, is one example). Indeed, many dynamicist models rely on precisely this property.

Notice also that adaptability can be provided through sophisticated representational structures. A prototypical symbolicist model can respond to input it has never seen before based solely on the syntactic structure of that input, often producing reasonable results. As long as the rules it has been programmed with, or has learned, can use structure to generalize, it will be able to exhibit adaptability: one example, of course, is syntactic generalization.

So, adaptability has been identified and explored by all past approaches to cognition. While the kinds of adaptability considered are often quite different – chosen to demonstrate the strengths of the approach – all of these types of adaptability are exhibited by real cognitive systems to varying degrees. That is, exploitation of nonlinear dynamics, tuning of the system parameters to reflect statistical regularities, and generalization over syntactic structure are all evident in cognitive systems. Consequently, a theory which accounts for adaptability in as many of its various guises as possible will do well on this criteria – the more the better.

8.2.2.4 *Memory*

In order for a system to learn from its experience, it must have some kind of memory. Despite the obviousness of this observation, it is by no means obvious what the precise nature of memory itself is. In the behavioral sciences, many varieties of memory have been identified, including long-term, short-term, working, declarative, procedural and so on. Not only are there many different kinds of memory, but most of these different types of memory have been identified primarily through behavioral markers. Consequently, it is unclear what specific components a architecture needs, or how such components might internally function, in order to explain the varieties of memory.

Past approaches have treated the issue of memory in very different ways: as symbolic databases; as connection weights between nodes; as slowly varying model parameters; etc. Ultimately, the adequacy of a cognitive architecture on this criterion is going to be determined by its ability to address the wide variety of data that relates to memory. There are a vast number of possible phenomena to account for when considering memory, so allow me to simply focus my discussion on aspects of long-term and working memory, which seem crucial for cognitive behaviors.

Cognition often necessitates the manipulation of complex, compositional structures. Such manipulation will demand significant memory resources from both long-term and working memory. This is because the structures themselves must be stored in long-term memory, so as to be available for application to a cognitive task. As well, each element and relation in such a structure must also be stored (be it in the structure itself, or separately) in long term memory. Furthermore, when such structures are manipulated, intermediate steps and final results, as well as the original structure itself, may need to be stored in working memory.

Interestingly, there is a trade-off between manipulating complex structure, the kinds of representation in the structure, and working memory demands. If each element in a complex structure encoded all experientially relevant information, combinations of such elements would quickly become unmanageable. Such structures would be unmanageable in the sense that all of the information pertaining to all of the elements of the structure would have to be manipulated if the structure itself was manipulated, and hence working memory demands would be enormous. Of course, symbols solve precisely this problem. In natural language, manipulating the symbol “dog”, after all, does not mean constantly dealing with all of the information we have about dogs. Often, it means we can move around the three letter string “d-o-g” in place of all the connected semantic and structural information, and still perform useful manipulations. Consequently, we can perform sophisticated reasoning using symbols, while having a working memory that is limited to only about four items (Cowan, 2001).

However, we must be careful that the representations in working memory do not become orphaned from the massive amounts of past experience that give them their content. It is virtue of this content, after all, that a cognitive system can successfully deal with many of the problems it confronts. The utility of symbols is a double-edged sword: low working memory resource demands means that semantics are more computationally distant. Indeed, deep semantic processing seems to engage much of cortex (see chapter 3), presumably in an effort to access long-term information regarding the appropriate use of representations in working memory.

So, in this woefully incomplete discussion of memory, I have suggested that there are at least two important functions for memory in cognition. One is to address the problem of ensuring that symbol-like representations are appropriately connected to experiences related to their semantics; i.e., to address the symbol grounding problem. This is an important role for long-term memory. A second is to support the “moving around” and processing of structures that underwrite occurrent cognition. This is an important role for working memory. Thus, a

good cognitive architecture must account for the functions of, and relationships between, working and long-term memory, so as to not overload one, and not underestimate the depth of the other. These considerations, perhaps, are among the many reasons why researchers typically include “memory” in the list of important capacities to consider when building cognitive models.

8.2.2.5 Scalability

As I noted previously in section 1.3, all three approaches to cognitive theorizing have a large and complex system as their target of explanation. As a result, many of the simple cognitive models proposed in contemporary research must be “scaled up” to truly become the kinds of cognitive explanations we would like. The reason that scalability is a criterion for cognitive architectures is because of the many difficulties hidden in the innocuous sounding “scaling up”.

The reason we need to take scalability seriously is that it is notoriously difficult to predict the consequences of scaling. This is tragically illustrated by the case of Tusko the elephant. In 1962, Louis West and his colleagues decided to determine the effects of LSD on a male elephant, to see if it would explain the phenomenon of “musth”, a condition in which elephants become uncontrollably violent (West et al., 1962). Because no one had injected an elephant with LSD before, they were faced with the problem of determining an appropriately sized dosage. The experimenters decided to scale the dosage to Tusko by body weight, based on the known effects of LSD on cats and monkeys.

Unfortunately, five minutes after the injection of LSD the elephant trumpeted, collapsed and went into a state resembling a seizure. He died about an hour and a half later. Evidently, the chosen means of scaling the dosage had the effects of a massive overdose that killed the 7000 pound animal. If the dose had been scaled based on a human baseline, it would have been 30 times lower. If it had been scaled based on metabolic rate, it would have been about 60 times lower. If it had been scaled based on brain size, it would have been about 300 times lower. Clearly, the dimension along which you characterize an expected scaling is crucial to determining expected effects. The lesson here for cognitive theories is that scaling can seldom be accurately characterized as “more of the same”, since we may not know which “same” is most relevant until we scale.

A second reason to take scaling seriously, which is more specific to functional architectures, is the result of considering the complexity of potential interactions in large systems. As Bechtel and Richardson have forcefully argued, decomposition and simplification is an essential strategy for explaining complex systems (Bechtel and Richardson, 1993). It is not surprising, then, that most cognitive

theorists take exactly this kind of approach. They attempt to identify a few basic functional components, or principles of operation, that are hypothesized to characterize “full-blown” cognition. Because we are not in a position to build models of “full-blown” cognition, the principles are applied in a much more limited manner. Limitations are typically imposed by abstracting away parts of the system, and/or highly simplifying the task of interest. When faced with a different task to explain, a new model employing related principles and methodologies is often constructed and a new comparison is made. Sometimes, different models employing the same architecture will use few or no overlapping components, sometimes there will be significant overlap.

In either case, such a strategy is problematic because it skirts the main challenge of building complex systems. As any engineer of a complex real-world system will tell you, many of the challenges involved in building large systems come from characterizing the *interactions*, not the internal functions, of components of the system. As has been well established by disciplines such as chaos theory and dynamical systems theory, the interactions of even simple components can give rise to complex overall behavior. In order to ensure that the hypothesized cognitive principles and components can truly underwrite a general purpose, cognitive architecture, simple, task specific models must be integrated, and hence scaled up, to simultaneously account for a wide variety of tasks.

In the context of constructing cognitive architectures, these two considerations are closely related: the first suggests it is difficult to know in advance how to predict the effects of scaling, and the second suggests that the true challenges of scaling are found in the interactions of parts of large systems. Both considerations point to the importance of actually constructing large, scaled-up models to test a proposed architecture. Noticing this relationship can help us to apply the principle of scaling.

Since scalability is difficult to predict from simpler instances, the weakest form of scalability is *scalability in principle*. Scalability in principle amounts to demonstrating that there is nothing in your assumed characterization of cognition that makes it unlikely that you could scale the theory to account for full-blown cognition. This form of scalability is weak because it could well be the case that a crucial dimension for predicting scalability has been missed in such an analysis.

A far more significant form of scalability is *scalability in practice*. That is, actually building large models of large portions of the brain that are able to account for behaviors on many tasks, without intervention or task-specific tuning. Such scaling can be extremely demanding, both from a design standpoint and computationally. However, this simply means that being able to construct such models

makes it that much more convincing that the relevant architecture is appropriate for characterizing real cognitive systems.

8.2.3 Scientific merit

The final set of criteria I consider are those which relate to good scientific theories, regardless of their domain. While most introductions to the philosophy of science discuss between about five to eight different properties that make for a good theory (Quine and Ullian, 1970; McKay, 1999), here I consider only two: triangulation and compactness. I have picked these because they have proven to be the most challenging for cognitive theories (Taatgen and Anderson, 2010; Anderson, 2007). And, given the very large number of tunable parameters typical of current cognitive models, these two criteria are especially critical (Roberts and Pashler, 2000).

8.2.3.1 *Triangulation (contact with more sources of data)*

The behavioral sciences are replete with a variety of methods. Some methods characterize the opening and closing of a single channel on a neural membrane, while others characterize activity of an entire functioning brain. In some sense, all of these methods are telling us about the functioning of the brain. So, if we take our cognitive theory to be a theory of brain function, then information from all such methods should be consistent with and relatable to our theory.

Unfortunately, this ideal seems to not only be largely un-realized, but seldom even treated as a central goal by cognitive theorists. Perhaps this is because cognitive science researchers have generally been placed into traditional academic disciplines like psychology, neuroscience, computer science, and so on. As a result, the conventional methods of one discipline becomes dominant for a given researcher and so his or her work becomes framed with respect to that specific discipline. In some sense, the identification of “cognitive science” as a multidisciplinary but unified enterprise is an attempt to overcome such a tendency. Nevertheless, cognitive theories seem to often have a “home” in only one or perhaps two of the sub-disciplines of cognitive science.

A truly unified theory of cognitive function, in contrast, would have clear relationships to the many disciplines of relevance for understanding brain function. Most obviously this would be evident from such a theory being able to predict

the results of experiments in any relevant discipline. In short, the more kinds of data that can be used to constrain and test a theory, the better. We should, after all, be impressed by a model that not only predicts a behavioral result, but also tells us which neurons are active during that task, what kind of blood flow we should expect in the relevant brain areas, what sort of electrical activity we will record during the task, how we might disrupt or improve performance on the task by manipulating neurotransmitters within the system, and so on. While the interdisciplinary nature of the behavioral sciences can prove to be one of the most daunting aspects of performing research in the area, it also provides one of the most stringent tests of the quality of a purported theory of cognition.

8.2.3.2 *Compactness*

While I have called this criteria “compactness”, it often goes by the name of “simplicity”. In fact, I think the former better indicates what is important about theories that are deemed good. It is not the fact that they are “easy” or “simple” theories that makes them good – they may indeed be quite difficult to understand. Instead, it is that they can be stated comprehensively, in a highly succinct manner, that makes them good. Often, this kind of succinctness is possible because the theory employs mathematics, a language whose terms and relationships are well-defined, and which can succinctly express complex structure. While mathematics itself does not supply theories of the world (since the terms in mathematical expressions must be mapped onto the world), it *is* good for clearly expressing the relationships between those terms.

One reason compact expressions of a theory are highly valued is because they make it difficult to introduce unnoticed or arbitrary changes in a theory when employing it in different contexts. If a cognitive theory changes when moving from simple cognitive tasks to more complex ones, there is little sense to be made of it being a single, compact theory. For instance, if we must introduce a parameter in our theory for no reason other than to fit new data, the additional complexity introduced by that parameter is poorly motivated from a theoretical standpoint. Similarly, if we must change our mapping between our theory and the world depending on the circumstances (e.g., a “time step” in a model is not always the same amount of real time), the statement of our theory should include a description of how to determine what the appropriate mapping is, making the theory less compact. In short, any unprincipled “fitting” of a model to data pays a price in

compactness. Consequently, a theory will do well on this criterion if it can be stated succinctly, and if that statement remains consistent across all of its applications.

Notably, the two criteria related to scientific merit combine to provide opposing constraints. The result is a demand for theories that explain a wealth of data, but do so compactly. This, not surprisingly, is an ideal that is typical for scientific theories.

8.3 Conclusion

As I have said before, this list of criteria is no doubt incomplete. However, I have attempted to identify criteria that are unlikely to be removed from any such list. In other words, I suspect that any theory that had nothing to say about one of these criteria would be deemed worse than one that did, all else being equal. I have also attempted, no doubt unsuccessfully, to identify these criteria in a non-theory-laden way. For example, even if we don't think there are representations, we still ought to think that the system has compositional behavior, because we have operationalized such behavior, and hence can measure it. In general, most of these criteria relate directly to measurable properties of cognitive systems, so hopefully satisfying them comes with reasonably few pre-theoretic constraints.

Recall that these QCC are largely inspired by the questions behavioral scientists have been asking about cognition over the last 50 years (see section 1.3). Undoubtedly, such questions have been driving decisions about the kinds of data to collect, and to do so they must be far more specific than the QCC I have identified. One result of this specificity seems to be that much of our knowledge about cognitive systems is highly empirical – we know lots about how people perform on various kinds of memory tasks, but we do not know why. That is, we can *describe* regularities in behavior, but we have not characterized the underlying principles or mechanisms well enough to really understand the genesis of that behavior. As a result we do not really understand how the system would function under a much wider variety of circumstances than we have explicitly tested. Using the QCC to evaluate cognitive approaches can help emphasize the importance of also striving for systematic, theoretical characterizations of cognition, to complement our empirical understanding.

Unfortunately, as I argue in the next chapter, current approaches do not generally do very well on more than a few of these criteria (see table 9.1 for a summary). Pessimistically, we might think that the last 50 years of research in the behavioral sciences has taught us that a satisfactory cognitive theory is missing. Optimisti-

cally, we might think that we are at least in a better position to evaluate cognitive theories than we were 50 years ago. In either case, the last 50 years should make us humble.

Chapter 9

Theories of cognition

9.1 The state of the art

In chapter 1, I recounted a history of cognitive science in which there are three main contenders: symbolism, connectionism, and dynamicism. From that discussion, it may seem natural to conclude that each paradigm is likely to be equally well represented by the state-of-the-art cognitive architectures. However, once we begin to examine implemented architectures, it becomes clear that this is not the case. By far the dominant paradigm underwriting contemporary, functioning cognitive architectures is symbolism.

In fact, it is somewhat of an embarrassment of riches when it comes to symbolic cognitive architectures. In a number of recent surveys, which list over 20 different architectures, all but a small handful are symbolic (Pew and Mavor, 1998; Ritter et al., 2001; Taatgen and Anderson, 2010).¹ To be fair, many of these architectures include methods for integrating some connectionist-type components into their models (e.g. Cogent, (Cooper, 2002)), but typically as something of an afterthought. However, there are some architectures that consider themselves to be explicitly hybrid architectures (e.g., Clarion, (Sun, 2006)). Inclusion of dynamicist components in cognitive architectures is even more rare.

Arguably, the most successful and widely applied cognitive architecture to date is the ACT-R architecture (Anderson, 2007), which is symbolist. The current 6.0 version is one in a long line of architectures that can trace their methodological assumptions back to what many consider the very first cognitive archi-

¹See http://en.wikipedia.org/wiki/Cognitive_architecture for an accessible but highly incomplete list.

tecture, the General Problem Solver (GPS; Newell and Simon, 1976). Consequently, ACT-R shares central representational and processing commitments with most symbolic architectures, and thus is a natural choice as a representative of the paradigm. While ACT-R contains some “neurally inspired” components, making it something of a hybrid approach, the core representational assumptions are symbolicist. Interestingly, it has recently been used to explore the biological plausibility of some of its central assumptions, and is unique amongst symbolic approaches in its ability to map to both behavioral and neural data. Because I am centrally interested in *biological cognition*, these features of ACT-R make it an important point of comparison for the SPA. In many ways, the ACT-R architecture embodies the best of the state-of-the-art in the field.

Nevertheless, there are several other architectures which are more biologically inspired, connectionist approaches to cognitive modelling, and hence are also natural to compare to the SPA. These include architectures that use various mechanisms for constructing structured representations in a connectionist network, such as Neural Blackboard Architectures (van der Velde and de Kamps, 2006), which use local integrators, and SHRDLU (Shastri and Ajjanagadde, 1993) and LISA (Hummel and Holyoak, 2003), which use synchrony. Another influential connectionist-based approach to structure representations focuses less on specific implementations, and more on proposing a broad theoretical characterization of how to capture symbolic (specifically linguistic) processing with distributed representations and operations (Smolensky and Legendre, 2006a,b). The associated architecture is known as ICS, and shares central commitments with the SPA regarding structured representations. Other connectionist approaches focus less on representational problems, and deal more with issues of control and learning in cognitive tasks. Leabra is an excellent example of such an approach, which has had much success mapping to reasonably detailed neural data (O’Reilly and Munakata, 2000). In addition, there has been a recent effort to combine Leabra with ACT-R (Jilk et al., 2008), as a means of simultaneously exploiting the strengths of each. This, again, provides an excellent comparison to the SPA.

Finally, there are some dynamicist approaches to cognitive modeling that have been gaining prominence in recent years (Schöner, 2008). Perhaps the best known amongst these is Dynamic Field Theory (DFT), which employs a combination of dynamicist and neurally-inspired methods to model cognitive behaviors (e.g., Schöner and Thelen, 2006). The focus of DFT on time, continuity, and neural modeling provides a useful and unique comparison for the SPA.

In the remainder of this chapter, I consider each of these approaches in more detail, describing their strengths and some challenges each faces. This discussion

will provide background for a subsequent evaluation of these theories with respect to the QCC. And, it sets the stage for an explicit comparison between this past work and the SPA.

Before proceeding, it is worth explicitly noting two things. First, there is important work in cognitive science that has been very influential on the architectures described here, including the SPA, which I do not discuss in detail for lack of space and/or lack of a full-fledged architecture specification (e.g., Newell, 1990; Rogers and McClelland, 2004; Barsalou, 2003). Second, because I am considering six architectures, each receives a relatively short examination. This means that my discussion is somewhat superficial, though hopefully not inaccurate. As a consequence the reader is encouraged to follow up with the provided citations for a more detailed and nuanced understanding of each of the architectures under consideration.

9.1.1 ACT-R

Adaptive control of thought-rational, or ACT-R, is perhaps the best developed cognitive architecture. It boasts a venerable history with the first expression of the architecture as early as 1976 (Anderson, 1976). It has been very broadly applied to characterizing everything from child language development (Taatgen and Anderson, 2002) to driving (Salvucci, 2006), and from the learning of algebra (Anderson et al., 1995b) to categorization (Anderson and Betz, 2001).

The many successes and freely available tools to build ACT-R models have resulted in a large user community developing around this architecture (see <http://act-r.psy.cmu.edu/>). This in turn has had the effect of there being many “sub-versions” of ACT-R being developed, some of which include central components from originally competing architectures. For example, a system called ACT-R/PM includes “perceptual-motor” modules taken from EPIC (Kieras and Meyer, 1997). This has lead to proponents claiming that the ACT-R framework allows the creation of “embodied” cognitive models (Anderson, 2007, p. 42). Since a central goal of ACT-R is to provide descriptions of “end-to-end behavior” (Anderson, 2007, p. 22), this is an important extension of the methods.

The basic ACT-R architecture consists in a central procedural module, which implements a production system, and is bidirectionally connected to seven other modules (i.e., the goal, declarative, aural, vocal, manual, visual, and imaginal buffers). Each of these modules has been mapped to an area of the brain, with the procedural module being the basal ganglia, and hence responsible for controlling communication between buffers and selecting appropriate rules to apply based on

the contents of the buffers. A central constraint in ACT-R is that only a single production rule can be executed at a time, and that it takes about 50 ms for a production rule to fire (Anderson, 2007, p. 54).

Because of its commitment to a symbolic specification of representations and rules, most researchers take ACT-R to be a largely symbolicist cognitive architecture. However, its main proponent, John Anderson, feels no particular affinity for this label. Indeed, he notes that it “stuck in [his] craw” when he was awarded the prestigious Rumelhart prize in 2005 as the “leading proponent of the symbolic modeling framework” (Anderson, 2007, p. 30). In his view, ACT-R is equally committed to the importance of subsymbolic computation in virtue of at least two central computational commitments of the architecture. Specifically, “utilities” and “activations” are continuously varying quantities in the architecture which account for some central properties of its processing. For example, continuous-valued utilities are associated with production rules to determine their likelihood of being applied. As well, utilities slowly change over time with learning, determining the speed of acquisition of new productions. Anderson takes such subsymbolic commitments of the architecture to be at least as important as its symbolic representational commitments.

As well, these subsymbolic properties are important for the recent push to map ACT-R models to fMRI data, because they play a crucial role in determining the timing of processing in the architecture. In his most recent description of ACT-R, John Anderson provides several examples of the mapping of model responses to the bold signal in fMRI (Anderson, 2007, pp. 74-89, 119-121, 151). This mapping is based on the amount of time each module is active during one brief fMRI scan (usually about 2s). The idea is that an increase in the length of time a module is active results in increased metabolic demand, and hence an increased BOLD signal, picked up by the scanner. While the quality of the match between the model and data vary widely, there are some very good matches for certain tasks. More importantly, the inclusion of neural constraints, even if at a very high-level, significantly improves the plausibility of the architecture as a characterization of the functions being computed by the brain.

Several critiques have been levelled against the ACT-R architecture as a theory of cognition. One is that the architecture is so general, that it does not really provide practical constraints on specific models of specific tasks (Schultheis, 2009). Before the explicit mappings to brain areas of the modules, this criticism was more severe. Now that there is a more explicit implementational story, however, there are at least some constraints provided regarding which areas, and hence which modules, must be active during a given task. However, these constraints

are not especially stringent. As Anderson notes, the mapping to brain areas is both coarse, and largely a reiteration of standard views regarding what the various areas do (Anderson, 2007, p. 29).

For instance, the ACT-R mapping of the declarative module to prefrontal areas is neither surprising, nor likely to be generally accurate. The prefrontal cortex is extremely large, and is differentially involved in a huge variety of tasks sometimes in ways not suggestive of providing declarative memories (Quirk and Beer, 2006). So, it is not likely informative to map it to only one (i.e. declarative memory) of many functions ascribed to the area. As well, it would be surprising if all retrieval and storage of declarative memory could be associated with prefrontal areas, given the prominent role of other structures (such as hippocampus) in such functions (Squire, 1992).

Similarly, the visual module is mapped to the fusiform gyrus, despite the acknowledgement that there are many more areas of the brain involved in vision. While fusiform no doubt plays a role in some visual tasks, it is somewhat misleading to associate its activation with the “visual module” given the amount of visual processing that does not activate fusiform.

In addition, the mapping between an ACT-R model and fMRI data often have many degrees of freedom. In at least some instances, there are several parameters for such a mapping, each of which are tuned differently for different areas, and differently for different versions of the task to realize the reported fits (Anderson et al., 2005). This degree of tuning makes it less likely that there is a straightforward correspondence between the model’s processing time and the activity of the brain regions as measured by BOLD. Thus, claims that ACT-R is well-constrained by neural data should be taken with some skepticism. In fairness, Anderson (2007) does note that these mappings are preliminary, acknowledges some of the concerns voiced here, and clearly sees the fMRI work as only the beginning of a more informative mapping (p. 77, 248).

Another criticism of the ACT-R approach is its largely non-embodied approach to cognition. For instance, the claim that ACT-R/PM provides for “embodied” models seems to overreach the actual application of this version of ACT-R. The perceptual and motor areas of this architecture do not attempt to capture the processing of natural images, or the control of high-degree of freedom limbs. Instead, they provide lengths of time that such processing might take, and capture some high-level timing data related to processing such as attentional shifts. Applications of such models are to tasks such as the effect of icon placement and design on reaction times. Visual representations in such models are specified as symbolic lists of features: e.g., “gray circle” (Fleetwood and Byrne, 2006). It seems disin-

genuine to suggest such a model is “embodied” in the sense typically intended by theorists of embodied cognition (Haugeland, 1993; Clark, 1997). Consequently, ACT-R has not yet achieved its laudable goal of accounting for end-to-end behavior.

Nevertheless, ACT-R continues to develop. And, there are continuing attempts to ground the architecture in biological structure. Of particular note is the ongoing attempt to integrate ACT-R with an architecture I consider shortly, Leabra. I will discuss this combination in more detail in my discussion of Leabra. However, it is helpful to see what kinds of challenges for ACT-R this integration is expected to meet. For example, this integration is intended to answer questions such as “how can ‘partial matching’ of production rules that ‘softens’ traditional production conditions operate,” “how can the learning of utility parameters that control production selection be grounded in plausible assumptions about the nature of feedback available,” and “how can context influence production matching and selection beyond the explicit specification of precise conditions” (Jilk et al., 2008, p. 211). What seems to tie these considerations together are ways of making symbols less rigid (see also my discussion in section 7.4). In the end, then, the commitment of ACT-R to symbolic representations seems to be hampering its mapping to neural structure.

In conclusion, the many successes of ACT-R are undeniable, and the desire to constrain the architecture further with detailed neural considerations is understandable and to be commended. However, it is not yet clear if the commitment of ACT-R to symbolic representations, production rules, and a production system will, in the end, allow it to develop in the way its proponents foresee.

9.1.2 Synchrony-based approaches

When faced with the problem of implementing structured representations, like those used in ACT-R, in a neural substrate, one of the basic decisions that must be made is how to implement the “binding” of objects within relations. In the next three sections, we encounter three different suggestions for how this might be done in a cognitive architecture.

One of the earliest suggestions for how this kind of binding might occur was taken from a suggestion in visual neuroscience that binding might occur through the synchronization of spiking activity across neurons representing different elements of the overall representation (von der Malsburg, 1981). This suggestion has been imported into cognitive modeling by Shastri and Ajjanagadde (1993) in their SHRDLU architecture. More recently it has seen perhaps its most so-

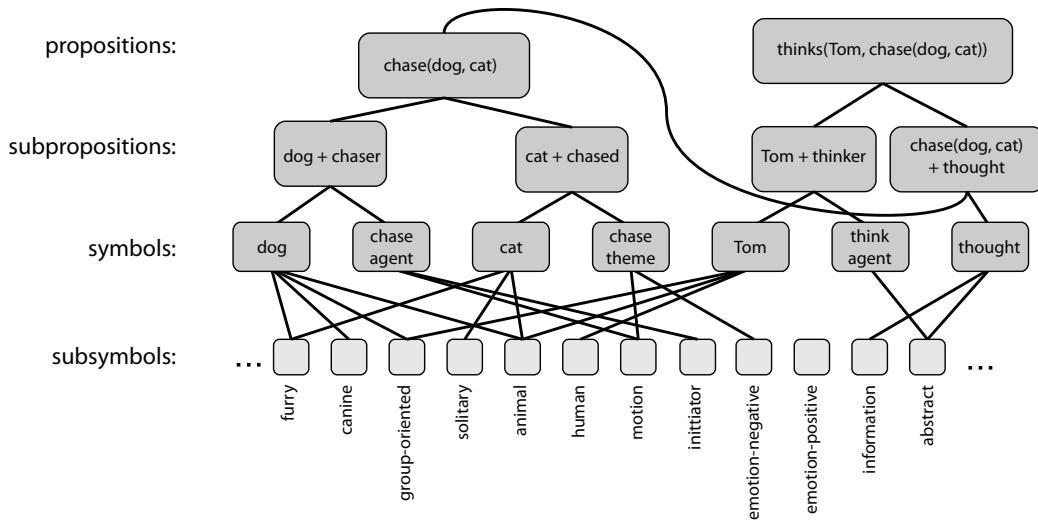


Figure 9.1: The LISA architecture. Boxes represent neural groups and lines represent synaptic connections. Shown are just those neural groups needed to represent “dogs chase cats” and “Tom thinks that dogs chase cats”. Based on Hummel & Holyoak (2003), figure 1.

phisticated neural expression in the recent LISA architecture from Hummel and Holyoak (2003).

In LISA, synchrony is used as a direct method for neurally representing structured relations (Hummel et al., 1994; Hummel and Holyoak, 1997, 2003). In this architecture, a structured representation is constructed out of four levels of distributed and localist representations. The first level consists of localist “subsymbols” (e.g. mammal, furry, male, etc.). The second level consists of localist units connected to a distributed network of subsymbols relevant to defining the semantics of the second level term (e.g., dog is connected to furry, mammal, etc.). The third level consists of localist “subproposition” nodes that bind roles to objects (e.g., dog+verb-agent, or dog+verb-theme, etc.). The fourth and final level consists of localist proposition nodes that bind subpropositions to form whole propositions (e.g., dog+chase-agent and cat+chase-theme). LISA is an interesting case because it is clearly a connectionist network, but it implements a classical representational scheme since all elements of the structures are explicitly tokened whenever a structure is tokened. In this sense, it is precisely the kind of implementation that could underwrite the symbolic representations of ACT-R.

Hummel and Holyoak demonstrate how LISA can perform relational inference

and generalization in a cognitive architecture that they consider “psychologically and neurally realistic” (p. 220). They demonstrate LISA on several examples that show how the architecture can solve inference problems and learn new schemas from past examples to solve future problems. Their work focuses on the role of analogy, and hence a central feature of the architecture is its ability to “map” one structured representation to another. For example, given the prior knowledge that Bill has a Jeep, Bill wanted to go to the beach, and thus Bill drove his Jeep to the beach, LISA can infer in a new instance in which John has a Civic and John wants to go to the airport, that John will drive his Civic to the airport (p. 236). Hummel and Holyoak proceed to show how this basic ability can explain a myriad of psychology experiments, and argue that LISA “provides an existence proof that [modeling human relational inference] can be met using a symbolic-connectionist architecture in which neural synchrony dynamically codes variable bindings in [working memory]” (p. 245).

It is clear that their architecture can capture some features of human reasoning. It is less clear that the model uses “neurally realistic” mechanisms and assumptions. Hummel and Holyoak are careful to note that each of their localist units is intended to be a population of neurons: “we assume that the localist units are realized neurally as small populations of neurons (as opposed to single neurons), in which the members of each population code very selectively for a single entity” (p. 223). Because of this selectivity, each such population only represents one object, subproposition, or proposition. They note that this kind of representation is essential to the proper functioning of their model. But unfortunately, this limits the ability of the model to scale to realistic representational requirements. Since a separate neural group is needed to represent each possible proposition, the total neural requirements of this system grow exponentially. To represent any simple proposition of the form *relation(agent, theme)*, assuming 2000 different relations and 4000 different nouns, $2000 \times 4000 \times 4000 = 32,000,000,000$ (32 billion) neural *populations* are needed. This is unrealistic, given that the brain consists of about 100 billion *neurons*, can handle a much larger vocabulary (of approximately 60,000 words; see appendix B.5 for details), and needs to do much more than represent linguistic structures. So, while their representational assumptions may be appropriate for the simple problems considered in their example simulations, such assumptions do not allow for proper scaling to truly cognitive tasks.

A separate aspect of LISA that is neurobiologically problematic is its use of neural synchrony for performing binding. There have been many criticisms of the suggestion that synchrony can be effectively used for binding. For example, after considering binding, O'Reilly and Munakata (2000) conclude that “the available

evidence does not establish that the observed synchrony of firing is actually used for binding, instead of being an epiphenomenon” (p. 221). Similarly, from a more biological perspective, Shadlen and Movshon (1999) present several reasons why synchrony has not been established as a binding mechanism, and why it may not serve the functional role assigned by proponents.

Furthermore, synchrony binding in LISA is accomplished by increasing real-valued spike rates together at the same time. Synchrony in LISA is thus very clean, and easy to read off of the co-activation of nodes. However, in a biological system, communication is performed using noisy, individual spikes, leading to *messy synchronization*, with detailed spectral analyses necessary to extract evidence of synchronization in neural data (Quyen et al., 2001). Thus it is far from clear that using more neurally realistic, spiking, noisy nodes would allow LISA to exploit the mechanism it assumes.

However, there is an even deeper problem for LISA. The kind of synchronization exploited in LISA is nothing like that that has been hypothesized to exist in biological brains. In LISA, synchronization occurs because there are inhibitory nodes connected to each subproposition that cause oscillatory behavior when the subproposition is given a constant input. That oscillation is then reflected in all units that are excitatorily connected to these subpropositions (i.e., propositions and themes/verbs/agents). Therefore, in LISA binding is established by constructing appropriately excitatorily connected nodes, and the oscillations serve to highlight one such set of nodes at a time. So, the representation of the binding *results in* synchronization patterns: this essentially puts the bound cart before the synchronous horse. Synchronization in the neurobiological literature is supposed to *result in* binding (see, e.g., Engel et al., 2001). For instance, if “red” and “circle” are co-occurring features in the world, they are expected to be synchronously represented in the brain, allowing subsequent areas to treat these features as bound. Thus the synchronization patterns result in binding, not the other way around. Consequently, the neural plausibility of LISA is not supported by current work on synchronization in the neurosciences.

9.1.3 Neural blackboard architecture (NBA)

LISA suffers from severe scaling problems because it relies on dedicated resources for each possible combination of atomic symbols. As a result, attempts have been made to avoid this problem by positing more flexible, temporary structures that can underwrite symbolic binding. One recent approach is that by van der Velde and de Kamps (2006) who propose a binding mechanism that they refer to

as the Neural Blackboard Architecture (NBA).

Blackboard architectures have long been suggested as a possible cognitive architecture (Hayes-Roth and Hayes-Roth, 1979; Baars, 1988), though they tend to find their greatest application in the AI community. Blackboard architectures consist in a central representational resource (the blackboard) that can be accessed by many, often independent processes that specialize in different kinds of functions. The blackboard, along with a controller, provides a means of coordinating these many processes, allowing simpler functional components to work together to solve difficult problems.

Given their AI roots, implementations of these architectures tend to be symbolic. However, the growing interest in trying to relate cognitive architectures to brain function, coupled with concerns about scalability of some neural architectures, has lead van der Velde and de Kamps to propose their methods for implementing blackboard-like representations in a connectionist network. Like proponents of ACT-R and LISA, they are interested in demonstrating that some aspects of the symbolic representational assumptions underlying the architecture can be given a neurally plausible characterization.

In their presentation of the NBA, van der Velde and de Kamps explicitly describe it as a means of addressing Jackendoff's challenges (also captured by the representational QCC; section 8.2.1). To avoid the possibility of an exponential growth in the number of neurons needed for structure representation, the NBA uses a smaller set of “neural assemblies” that can be temporarily associated with particular basic symbols. Larger structures are then built by binding these assemblies together using a highly intricate system of neural gates. This not only allows for representation of basic propositional logic, but also of full sentential structures.

In order for word assemblies to be bound to structure assemblies, there is a connection structure that links the assemblies, and is able to have sustained, slowly decaying activation. It is the activation of that connection structure (a “memory circuit”) which determines the binding. A separate “gating circuit” determines which role (e.g., *agent* or *theme*) a particular word assembly is connected to in the structure (see figure 9.2).

The use of neural assemblies that can be temporarily bound to a particular symbol greatly reduces the number of neurons required for this method. As with LISA, each of the localist units is actually intended to be a group of neurons, meaning that the architecture ignores the details of neural spiking activity, and instead deals with an abstract, real-valued neural “activity”. If we assume that these neural groups consist of an average of 100 neurons, and that the vocabulary is limited as in the LISA example (6000 terms), the NBA can represent any struc-

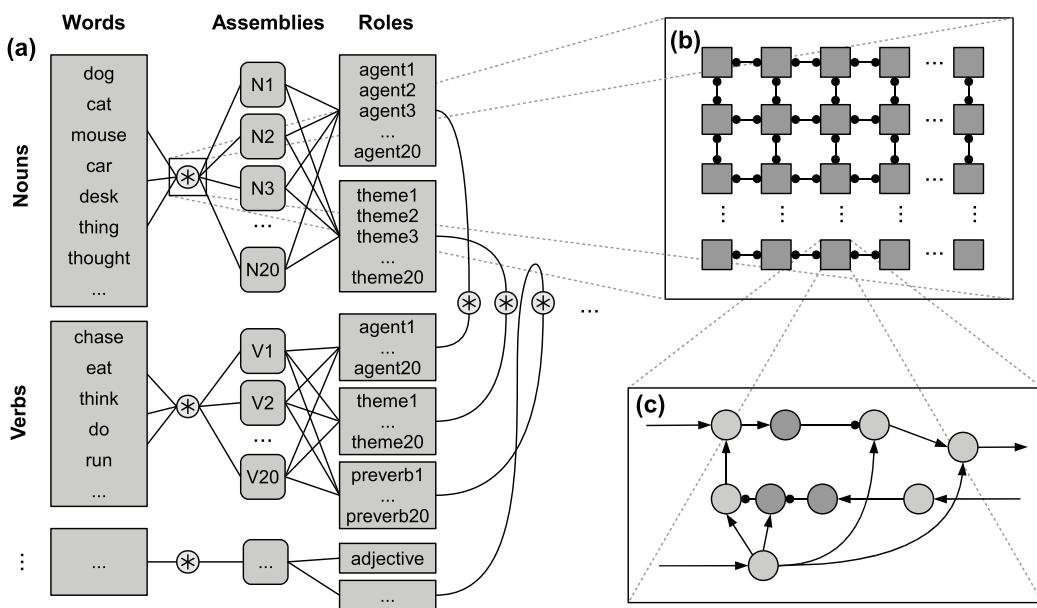


Figure 9.2: The Neural Blackboard Architecture. Neural groups representing words in (a) are connected by mesh grids (b) built using the neural circuit (c). Excitatory connections end with arrows, and inhibitory connections with circles. (See van der Velde & de Kamps (2006) for more details.)

tured proposition by employing approximately 480,000,000 neurons. For a more realistic, adult-sized vocabulary of 60,000 symbols, around 4,800,000,000 neurons are needed, or about 480cm² of cortex (see appendix B.5 for details). This is approximately one-quarter of the total area of cortex, which is much larger than the size of known language areas.

While this is a slightly more reasonable number of neurons than in LISA, it does not suggest good scaling for the NBA. This is because these resources are required only for the representation of structure, not the processing, transformation, updating, etc. of these structures. The focus of the NBA on representation is a reasonable starting point, but it means there is little provision for characterizing a general cognitive architecture. To fully specify a blackboard architecture, you need to also specify the controller, and the other processes interacting through the blackboard – each of these will require more neural resources. As it stands, the NBA is something of a blackboard with no one to write on it.

There are other reasons to be concerned with the proposed representational structure beyond the problems with scaling and function. For instance, the gating structures (see figure 9.2b, c) are highly complex and carefully organized, such that the loss or mis-wiring of one connection could lead to catastrophic failure of the system. There is no evidence that microlesions should be expected to have these sorts of effects. On the contrary, lesions underlying syntactic deficits are quite large, typically larger than for other deficits (Hier et al., 1994). Furthermore, connecting all nouns to all noun assemblies and all noun assemblies to all verb assemblies requires long distance and highly complete connectivity (within the entire 480 cm² area), while most real cortical connections are much more local and sparse (Song et al., 2005). As well, the evidence cited in the original paper to support the necessary specific structures merely demonstrates that some individual inhibitory cells in visual cortex synapse on other inhibitory cells in the same layer (Gonchar and Burkhalter, 1999). This does not render the NBA architecture plausible in its details.

Notably, like LISA, the NBA is also best seen as an implementation of classical symbolic representation. Each word, structural element, and binding is explicitly represented in a specific location in the brain. As such, it is another attempt that could address ACT-R’s problem of relating symbolic representations to neural implementation. However, the proposed implementation makes unrealistic assumptions about the connectivity and robustness of the neural substrate. Finally, even with those assumptions, it seems unlikely that the architecture will scale to the level of human cognition with the resources available in the brain.

9.1.4 The integrated connectionist/symbolicist architecture (ICS)

A third approach to characterizing symbolic binding in a neurally-inspired architecture is that employed in the SPA: conjunctive coding. As mentioned in my earlier discussion, the VSA approach I adopted has its conceptual forerunner in the tensor product representations suggested by Smolensky (1990). In an excellent and ambitious recent book, Smolensky and Legendre (2006a) describe how tensor products can be used to underwrite language processing, and suggest what they call the Integrated Connectionist/Symbolicist Cognitive Architecture (ICS). Coupled with their representational considerations are Optimality Theory (OT) and Harmonic Grammar (HG), which provide functional characterizations of linguistic processing as soft-constraint satisfaction. As a result, the ICS is largely characterized in the context of linguistics, especially phonology.

Specifically, Harmonic Grammar is used to demonstrate that tensor products can have sufficiently complex embedded structure to do linguistic processing. As such, it provides a link between the purely symbolic Optimality Theory, and the implementation of such a theory in a connectionist network. It should be noted, that both HG and OT are used in the context of localist networks. While Smolensky and Legendre do describe how the localist HG network could be systematically mapped to distributed representations, the actual models are not run or optimized with distributed representations (see, e.g., Smolensky and Legendre (2006a) chapter 11, where the relationship is best described), but rather solved using localist networks.

Regardless, the work has been widely appreciated for its comprehensive and detailed account of the relations between language processing, parallel constraint satisfaction, and connectionism. The ICS provides a clear, quantitative description of how specific constraints on language processing can be used to explain a wide variety linguistic phenomena including phonology, grammaticality judgments, and semantic/syntactic interactions. The ICS is clearly a significant achievement, and it provides by far the most comprehensive connectionist account of language currently available. Furthermore, the methods are systematic and rigorous, meaning their application to new phenomena is clear and the predictions are testable. In short, the ICS goes far beyond the SPA or any other architecture in its description of the functions need to explain linguistic behavior.

That being said, however, there are some concerns about considering the ICS as a general account of biological cognition. For one, the biological plausibility is untested and unclear as it stands. As mentioned, simulations are typically localist, and it is not evident how populations of neurons could perform the same functions

without being “wired up” differently for each optimization (e.g. to parse each sentence). Relatedly, there is no suggestion for what kind of control structures could “load” and “manipulate” language as quickly as it happens in the human system. So despite the isomorphic mapping between localist and distributed networks, it remains a major challenge for the ICS to show how a distributed network can actually perform all the steps needed to process language in a biologically relevant way.

As with past approaches, biological plausibility is also related to scaling issues, a known challenge for the ICS’s preferred method of binding: the tensor product. As mentioned earlier, tensor products scale badly because the product of two N dimensional vectors results in an $N \times N$ length vector. So, embedded structures become exponentially large and unwieldy. For example, to represent structures like “Bill believes John loves Mary” using 500-dimensional vectors (as in the SPA) would require 12 billion neurons, or about half the area of available cortex (see appendix B.5) – as with the NBA, this is only for representation. In their book, Smolensky and Legendre explicitly note the utility of a compression (or “contraction”) function (like that employed by all other VSAs) to address this scaling issue: “the experimental evidence claimed as support for these models suggests that contracted tensor product representations of some kind may well be on the right track as accounts of how people actually represent structured information” (p. 263). However, they do not subsequently use such representations.

Staying with the assumption of standard tensor products seems to be motivated by Smolensky and Legendre’s explicit concern to connect a description of the brain as a numerical computer with the description of mind as a symbolic computer. As a result, their approach has been praised by Anderson (2007) for its focus on how to do complex processing with the proposed representation: “The above criticism is not a criticism of connectionist modeling per se, but ... of modeling efforts that ignore the overall architecture... in the Smolensky and Legendre case, [their approach] reflects a conscious decision not to ignore function” (p. 14-15). The approach appeals to Anderson because Smolensky and Legendre hold a view of cognitive levels that is compatible with Anderson’s suggestion that symbolism often provides the “best level of abstraction” for understanding central aspects of cognition (2007, p. 38). In their discussion of the ICS, Smolensky and Legendre explicitly note their commitment to an isomorphism between their representational characterization in terms of vectors and a characterization in terms of symbols ?, p. 515. If you use compressed representations, however, the isomorphism no longer holds. Compressed representations lose information about the symbolic structures they are taken to encode, so they can quickly diverge from

an “equivalent” symbolic representation.

This commitment to isomorphism is not a throw away aspect of the ICS: “The formal heart of these [ICS] principles is a mathematical equivalence between the fundamental objects and operations of symbolic computation and those of specially structured connectionist networks” (p. 147). Ironically, perhaps, this is the same equivalence that ACT-R researchers have given up in their concern to match behavioral data, and why Anderson does not consider ACT-R to be a (purely) symbolic approach. In any case, if this equivalence could be realized, it would provide ACT-R with a direct means of relating its symbolic representations to neuron-like units. This, of course, is the goal that has been shared by all three connectionist approaches we have seen to this point.

But, all three have run into the same two problems: scaling and control. Control relates to how the system itself can construct, manipulate, decode, etc. the representations employed. The scaling issue relates to difficulties in mapping standard symbolic representations into neural ones. Perhaps the control problem could be solved with time, but I suspect that the scaling issue is symptomatic of a deeper problem.

I would like to suggest that the representational issues found in LISA, the NBA, and the ICS are instructive: after all, all three are approaches for implementing a classical form of symbolic representation, and all three are inconsistent with known neural constraints. I suspect that this failure is a fundamental problem with attempting to neurally implement a classical representational scheme: if we require each atom within a structure to appear in the representation of the overall structure (as per Fodor & Pylyshyn, 1988), then we will either have an exponential explosion in the number of neurons required (as in LISA and the ICS), or a complex and brittle system for temporarily binding and rebinding terms which is incompatible with known neural constraints (as in the NBA). In other words, implementing a classical (i.e. explicitly compositional) representational scheme directly seems to make for unscalable systems. So perhaps the problem lies in the “top-down” representational constraints assumed by symbolic architectures to begin with. Perhaps it is a mistake to think that a good characterization of biological cognition will come from an implementation of a largely symbolic system.

9.1.5 Leabra

The last connectionist cognitive architecture I will consider has taken a very different approach to characterizing cognition. Rather than focusing on representation, Leabra focuses on control and learning (O’Reilly and Munakata, 2000). The

Leabra (Local, Error-driven and Associative, Biologically Realistic Algorithm) system is a method that is intended to be used to learn central elements of a cognitive architecture. The algorithm is able account for results traditionally explained by either back-propagation or Hebbian learning, as it combines these two kinds of learning within its algorithm.

Proponents of Leabra have employed it widely, and have convincingly argued that it has helped develop our understanding of gating of prefrontal representations, as well as reward, motivation, and goal-related processing in cognitive control (O'Reilly et al., 2010). One recent application of Leabra is to the PVLV (Primary Value and Learned Value) model of learning (O'Reilly et al., 2007), which simulates behavioral and neural data on Pavlovian conditioning and the midbrain dopaminergic neurons that fire in proportion to unexpected rewards (it is proposed as an alternative to standard temporal-difference methods).

This learning model has been integrated into a model of the brain that includes several areas central to cognitive function. The result is called the PBWM (prefrontal-cortex, basal-ganglia working memory) model, which employs PVLV to allow a computational model of the prefrontal cortex to control both itself and other brain areas in a task-sensitive manner. The learning is centered around subcortical structures in the midbrain, basal ganglia and amygdala, which together form an actor/critic architecture. The authors demonstrate the model on several working memory tasks, perhaps the most challenging being the 1-2-AX task (O'Reilly and Frank, 2006; Hazy et al., 2007). This task consists of subjects being shown a series of numbers and letters, and their having to respond positively to two possible target patterns. The first is a 1 followed by AX, and the second is a 2 followed by BY. There may be intervening letters between the number and its associated letter string. There may also be non-target stimuli, to which no response is given. This is a difficult task because the subject must keep in mind what the last number was, as well as what the the associated next target and current stimulus pattern is, all while ignoring irrelevant stimuli. Nevertheless the model successfully performs the task.

The general structure (e.g., basal ganglia gating of pre-frontal cortex) of the PBWM model is supported by various sources of empirical evidence. For example, Frank and O'Reilly (2006) administered low doses of D2 (dopamine type-2 receptor) agents cabergoline (an agonist) and haloperidol (an antagonist) to normal subjects. They found that the Go and No-go learning required by the PBWM model on the 1-2-AX tasks was affected as expected. Specifically, cabergoline impaired, while haloperidol enhanced, Go learning from positive reinforcement, consistent with presynaptic drug effects. And, cabergoline also caused an overall

bias toward Go responding, consistent with postsynaptic action.

It is clear from such examples of past success that Leabra has taken a very different approach to modeling cognitive behavior than the other approaches I have considered. While the representations in the tasks employed are not as complex as those used by other connectionist architectures, the mapping onto brain structures, the ability to address control, and the inclusion of learning mechanisms are significant strengths of this work. It is perhaps not surprising that Leabra-based approaches are often lauded for their improved level of biological realism.

Nevertheless, there remain central features of Leabra which cast some doubt on how well the proposed mechanisms capture what we know about low-level brain function. For example, these methods again use rate neurons, which makes it difficult to compare the results to single cell data actually recorded from the brain structures of interest. This is especially important when it comes to detailed timing effects, such as those considered in section 5.7.

Perhaps more importantly, there are basic computational elements of Leabra, which are of dubious plausibility. The most evident is that it directly applies a k -Winner-Takes-All (kWTA) algorithm, which is acknowledged as biologically implausible: “although the kWTA function is somewhat biologically implausible in its implementation (e.g., requiring global information about activation states and using sorting mechanisms), it provides a computationally effective approximation to biologically plausible inhibitory dynamics” (<http://grey.colorado.edu/emergent/index.php/Leabra>; see also O'Reilly and Munakata (2000), pp. 94-105 for further discussion). In other words, the actual dynamics of the system are replaced by an approximation that is computationally cheaper. The problem is that the approximation is much cleaner and faster than the actual dynamics, and hence will not allow precise prediction of timing information, and likely changes the system's function in various circumstances. For example, the basal ganglia is often characterized as a kWTA computation, but in our simulations (see section 5.7), the timing of behavior depends crucially on parameters like the amount of value difference between items, the complexity of the actions, and so on. While approximations can be practically useful, it is essential to clearly demonstrate that the approximation is accurate in situations in which it is employed. This is generally not the case for Leabra models, making its connection to neural dynamics, in particular, unclear.

Similar concerns arise when we consider the Leabra algorithm itself (pseudo-code is available at <http://grey.colorado.edu/emergent/index.php/Leabra>). Much is currently known about plausible learning mechanisms in the brain (see section 6.4). However, in the Leabra algorithm, there are several steps that are

not obvious how to implement in an unsupervised network, i.e. a network that does not have a “supervisor” able to control the flow of information through the network. For example, the algorithm includes combining two different kinds of learning at each step, one of which takes some amount of time to settle and another which does not. If there can be different numbers of learning steps within one simulation step, it seems likely that there would be unexpected dynamic interactions in an unsupervised implementation. As well, units are clamped and unclamped by the supervisor, so again there is concern that these changes of state are likely cause unexpected interactions if not externally controlled.

I should note that these concerns with biological plausibility are much different than those for past approaches. Here, the concerns are related to specific neural mechanisms more than to representational assumptions. This is largely because Leabra models typically do not have very sophisticated representations. Proponents do give some consideration to binding, though it is mostly in the context of simple vision tasks, and has not been extended to complex structured representations. Combining this limitation of Leabra with the complementary strengths and limitations of ACT-R has resulted in a recent push from proponents of both to combine their approaches.

The idea is that since they share an overall architectural commitment (basal ganglia as a controller, cortex as working memory, etc.) ACT-R can provide more sophisticated representations and Leabra can provide more realistic biological mechanisms (Jilk et al., 2008). In section 9.1.1, I outlined some of the concerns about ACT-R that Leabra is expected to help address. Similarly, ACT-R is expected to help Leabra by answering questions such as “how can new procedural learning piggyback on prior learning, rather than using trial and error for each new behaviour” and “how can verbal instructions be integrated and processed” (p. 211). In short, Leabra needs language-like representations and ACT-R needs biological plausibility.

Unfortunately, there has not been a lot of progress in combining the approaches. I believe this is because they have not effectively integrated symbolic and neuron-based representations. Instead, they associate different kinds of representations with different parts of the brain: “we associate the active buffers, procedural production system, and symbolic representations from ACT-R with the prefrontal cortex and basal ganglia, while the graded distributed representations and powerful learning mechanisms from Leabra are associated with the posterior cortex” Jilk et al. (2008, p. 213). While they explicitly worry about the “interface” between these kinds of representations, they do not have a solution for easily allowing these areas to communicate within a unified representational substrate. As a result, the

methods have currently not been integrated, but rather “attached.” Consequently, it is difficult to argue that the resulting models solve the kinds of problems they were attempting to address. I believe it is fair to say that the desired unification of biological realism and symbolic processing has so far remained elusive.

9.1.6 Dynamic field theory (DFT)

My examination of the state-of-the-art to this point has addressed only symbolic and connectionist approaches. However, since the mid 90s there has been a continuous and concerted effort to construct neurally-inspired cognitive models using a more dynamicist approach (Schöner, 2008). The methods behind the approach are known as Dynamic Field Theory (DFT), and are based on earlier work by Shun-ichi Amari on what he called “neural field equations” (Amari, 1975). These equations treat the cortical sheet as a continuous sheet of rate neurons that are coupled by inhibition and excitation functions. They thus model a “metric dimension” (e.g. spatial position, luminance, etc.) over which neural activity is defined. Much early work focused on the kinds of stable patterns (attractors) that could be formed on these sheets, and under what conditions different patterns formed. In essence, this was one of the earliest integrator network models of neural stability.

In the hands of modern DFT researchers, these networks of stable activity patterns are connected in various configurations to map onto more sophisticated behavior, such as control of eye movements, reach perseveration, infant habituation, and mapping spatial perception to spatial language categories. Interestingly, talk of “representations” is not uncommon in this work, despite a general resistance in dynamicist approaches to discussion representation (van Gelder, 1995; Port and van Gelder, 1995). For instance, Schöner (2008) comments that “localized peaks of activation are units of representation” (p. 109), and Mark Blumberg speaks of “representation-in-the-moment” in his introduction to DFT <http://www.uiowa.edu/delta-center/research/dft/index.html>. Nevertheless, there is also a strong emphasis on embodiment of the models, with many being implemented on robotic platforms, and an uncompromising description of the dynamics of the models.

Some of the best known applications of DFT have been to infant development. For instance, Schöner and Thelen (2006) present a DFT model of infant visual habituation. The model consists in two interacting and coupled neural fields. The first represents the activation that drives “looking,” and the second, the inhibition that leads to “looking away,” or habituation. The model is presented with simulated visual input of varying strengths, distances, and durations and is able to

simulate the known features of habituation, including familiarity and novelty effects, stimulus intensity effects, and age and individual differences. Other DFT models capture the development of perseverative reaching in infants (Thelen et al., 2001), or the improvement in working memory through development (Schutte et al., 2003).

More recent work has addressed more cognitive tasks, such as object recognition (Faubel and Sch, 2010), spatial memory mapping to spatial language (Lipinski et al., 2006), and speech motor planning (Brady, 2009). The speech motor task model, for example, is constructed out of two neural fields by mapping the input/output space to a first sheet of neural activity, and then having a second-order field learn how to map the current first-order state into a desired next first order state. This is a nice example of a simple hierarchical dynamical system, which is able to capture how a static control signals can aid the mapping between appropriate dynamic states.

We can see in such examples that, like Leabra, the DFT approach has used working memory models as an important component of building up more cognitive models. Unlike Leabra, DFT is focused on ensuring that the low-level dynamics are not compromised by introducing algorithmic methods into their simulations. Unlike Leabra, DFT is often not mapped to specific anatomical structures, and types of neurons.

As with Leabra, there are reasons to be concerned with the neural plausibility of DFT. DFT again relies on non-spiking single cell models. Spikes can introduce important fluctuations into the dynamics of recurrent networks, and so ignoring them may provide misleading results regarding network dynamics. As well, the assumption in DFT that all neurons are physiologically the same is not borne out by neural data. As described earlier, neurons possess a wide variety of responses, and are generally heterogeneous along a variety of dimensions. Because the DFT models are homogeneous, they are difficult to compare in their details to the kinds of data available from single neuron recordings.

But again, as with Leabra, there are more important concerns. Perhaps the most crucial drawback is that the kinds of representations, and hence the complexity of the tasks DFT models address, is very limited. Specifically, there are no methods for introducing complex structured representations into the approach. Hence, most cognitive applications address fairly minimal cognitive tasks, not the kind of tasks addressed by more traditional cognitive models, such as ACT-R. It is somewhat difficult to argue that these last two approaches are general cognitive architectures given their lack of ability to address high-level cognitive function. While these approaches have not made unscalable assumptions in trying to incor-

porate symbolic representations, they have also not provided an alternative that will allow their methods to be applied to the kinds of tasks for which symbolic representations are needed.

9.2 An evaluation

Now that I have provided a brief description of several past approaches to understanding biological cognition, describing some strengths and limitations of each, I now return to the QCC and consider how each criterion applies to the current state-of-the-art. I want to be clear that this evaluation is in no way intended to underestimate the important contributions of this past work, but rather it is intended to highlight points of difference, and to see the relative strengths and weaknesses of the various approaches that have been taken to understanding biological cognition to date.

9.2.1 Representational structure

a. Systematicity Because symbolic approaches to cognitive representations are systematic, so are the approaches that attempt to implement a symbolic representational system directly. Of the approaches I considered, this includes ACT-R, LISA, the NBA, and the ICS. The other two approaches, Leabra and DFT, do not traffic in symbolic representations, and hence do not have an obvious kind of representational systematicity. If, however, we take systematicity to also identify behavioral regularities, then it seems clear that both of these approaches have systematic processes and hence some systematic behavior. For instance, if Leabra can react appropriately to “1 appears before AX” then it can also react appropriately to “AX appears before 1”. Similarly for DFT: if it can identify that “the square is left of the circle”, then it can also identify that “the circle is left of the square”.

I take it, however, that most cognitive theorists understand systematicity more in the classical sense, and hence approaches that are able to employ representations which account for the processing of language-like structure do better with respect to this criteria.

b. Compositionality Like systematicity, compositionality seems equally well captured by ACT-R, LISA, the NBA, and the ICS. However, in this case, it is not

clear that they capture the “right kind” of compositionality. Recall from my discussion of this criteria earlier, that compositionality seems to be an idealization of the way in which meaning actually maps to linguistic structure. None of these approaches provides an especially detailed account of the semantics of the symbols which they employ. However, they all equally account for a purely compositional semantics, given their ability to implement a classical representation architecture.

It may seem that the ICS, because it employs highly distributed representations, may have more to say about the semantics of the employed representations. Smolensky and Legendre (2006a) do consider some cases of more grounded spatial representations, but in the end they note that the “semantic problem... will not be directly addressed in this book” (p. 163). The NBA and ACT-R do not discuss semantics in much detail. Though LISA does, the semantics typically consist only in sharing a few subsymbolic features which are chosen explicitly by the modeler (e.g. the “Bob” and “John” symbol nodes are both linked to a “male” subsymbol node and hence are semantically similar).

Again, Leabra and DFT do not capture classical compositional semantics because they do not implement structured representations. Nevertheless, proponents of Leabra consider linguistic processing (O’Reilly and Munakata, 2000). However, none of these linguistic models employ Leabra-based architectures. Instead, they are largely demonstrations that the underlying software can implement past connectionist models of language. DFT models that directly address semantics are not common, though some characterize the mapping between simple spatial words and perceptual representations (Lipinski et al., 2006). More generally, however, the DFT commitment to embodiment brings with it the suggestion that representational states in these models will have a more grounded semantics than are found in other approaches to cognition. The claim that language is highly influenced by the embodied nature of the language user is a common one in some approaches to linguistics (Gibbs Jr. (2006); though see Weiskopf (2010) for reservations).

Overall, current approaches seem to either account for classical compositional semantics, or to have the potential to ground semantics without relating this grounding to language-like representations. Of the past approaches, the ICS seems to have the representational resources to do both, but the grounding of its distributed representations is not considered. Clearly, much remains to be done to integrate simple classical compositional semantics with more sophisticated semantics grounded in detailed perceptual and motor representations.

c. Productivity (the problem of variables) Again, systems which implement classical representational schemes are able to be productive, at least in principle. In my previous discussion of productivity, I emphasized that doing well on this constraint meant matching the limits of real productivity. In this respect, ACT-R and LISA do better than the NBA and the ICS. LISA, for instance, places an explicit constraint on the number of items that can be in working memory while an analogy is being considered. This constraint plays no small role in its ability to match a variety of psychological results. As well, some ACT-R models manipulate a decay parameter that determines how quickly the activation of a production in declarative memory goes to zero. This can do an effective job of explaining working memory limitations in many tasks.

Because Leabra and DFT do not use language-like representations, they do not have much to say about productivity. If instead we take Jackendoff's more general "problem of variables" seriously, it becomes clear that much work remains to be done by both of these approaches. Most of the applications of DST and Leabra avoid describing how anything like a structured representational template can be employed by their approaches. While Leabra's behavior has been characterized in terms of such templates (e.g. in the 1-2-AX task), such templates are learned anew each time the model encounters a new scenario. Thus it is not clear how the learned relations can be manipulated as representational structures in themselves, that is, as more than just behavioral regularities. Jackendoff, of course, argues that in order to explain the flexibility of human cognition such structures are essential.

In many ways, Jackendoff's characterization of the productivity problem explains some of the scaling problems we find in LISA, the NBA and the ICS. In all three cases, it is the wide variety of items that can be mapped into the syntactic variables that result in unwieldy scaling when considering productivity at the level of human cognition. LISA, for example, requires nodes dedicated to all the possible bindings between variables and values. As described earlier, this results in an exponential explosion of nodes which soon outstrips the resources available in the brain. The NBA needs to hypothesize complex and fragile binding circuits. The ICS has very poor scaling when such structures get even marginally large.

So, while implementations of classical representations can explain productivity in principle, the representational assumptions they make in order to do so result in implausible resource demands. This is true despite the fact that some provide useful accounts of the limits of productivity. The approaches which do not employ language-like representations do not seem to provide a solution.

d. The massive binding problem (the problem of two) The “massive binding problem” is obviously a scaled up version of the “binding problem”. All of the implementations of classical representation that I have discussed have provided solutions to the binding problem. In each case, I considered their ability to scale in the context of linguistic representations. And, in each case, I demonstrated that the proposed representational and binding assumptions were not likely to be implemented in a biological system (see appendix B.5 for details).

ACT-R, in contrast, is silent on the question of what the neural mechanism for binding is. While this allows it to address this particular criterion well (as binding is essentially “free”), it will do poorly on subsequent criteria because of this agnosticism.

Again, DFT and Leabra have little to say about this particular issue because they do not describe methods for performing language-like binding in any detail.

So, while all of the neural implementations of classical representation are able to solve the problem of two (because they all posit a viable mechanism for small-scale binding), none of them scale up to satisfactorily address the massive binding problem. The approaches to biological cognition that do not account for binding also do poorly with respect to this criterion. Only ACT-R successfully addresses it, with the qualification that it does not do so for perceptual binding.

9.2.2 Performance concerns

a. Syntactic generalization As I mentioned in my earlier discussion of syntactic generalization (section 8.2.2.1), a commitment to a classical representational structure will automatically provide syntactic generalization in principle. However, there are two important caveats to this observation. First, being able to syntactically generalize in principle does not mean that a proposed classical representational system will be able to meet additional criteria, such as the speed at which such generalization must proceed. As I mentioned earlier (section 7.2) current approaches to structured representations are not able to meet known constraints on available neural mechanisms and the observed speed of new generalizations. The second caveat is that there are clear content effects in human performance of syntactic generalization (see section 6.6).

With this more pragmatic view towards syntactic generalization in mind, the four proposed approaches that adopt a classical representational method do not seem to fair well. ACT-R, for instance, suffers from being able to only provide an ad hoc solution to the content effects that are observed. Because there are no implicit semantic relationships between the symbols employed in ACT-R (i.e.,

we cannot compare “dog” to “cat” by performing a simple operation, such as a dot product, on those two items), any content effects have to be explained in virtue of there being separate, specific rules for different symbols. In view of such concerns, there has been work employing LSA-based semantic spaces in ACT-R (Budiu and Anderson, 2004), but the semantic information is generated outside of the ACT-R framework. Thus, if a new word is encountered by the model, it cannot be incorporated into its semantic representational space without re-running a costly LSA analysis.

LISA, the NBA, and the ICS are greatly affected by the temporal constraints. As mentioned in section 7.3.7, Hadley (2009) considers these constraints in detail, and concludes that such approaches do not have the ability to solve the problem of rapid variable creation. Essentially, LISA does not have a mechanism for adding new nodes to its localist network. Similarly, the NBA does not have a method for introducing and wiring up a new word or structure into its fairly complex architecture. The ICS also does not provide a characterization of how the system can itself generate representations that can be used to perform syntactic generalization on-the-fly. I would add that it is also not obvious how ACT-R would handle a completely novel symbol, and be able to generate a new production to handle its interpretation.

Perhaps unsurprisingly given their representational commitments, neither Leabra or DFT have been shown to perform syntactic generalization.

In sum, while ideal syntactic generalization can be performed by any of the approaches that employ classical representational schemes, the practical examples of such generalization in human behavior are not well accounted for by any of the current approaches.

b. Robustness I concluded my earlier discussion of robustness with the observation that this criterion demands that the models resulting from a cognitive theory should continue to function regardless of changes to their component parts, or variability in their inputs and environment (section 8.2.2.2). I also described how, historically, the more connectionist approaches had a better claim to robustness, often being able to “gracefully degrade” under unexpected perturbations. One question that arises, then, is whether or not the connectionist implementations of a classical representational scheme preserve this kind of robustness.

From my earlier descriptions of LISA and the NBA, it is evident that neither of these approaches is likely to remain robust in the face of neural damage. For instance, even minor changes to the complex wiring diagrams posited by the NBA

will rapidly reduce its ability to bind large structures. Similarly, binding in LISA will be highly sensitive to the removal of nodes, since many individual nodes are associated with specific bindings. As well, for LISA, a reliance on synchrony is subject to serious concerns about the effects of noise on the ability to detect or maintain synchronous states (section 9.1.2). Noting that each of the nodes of these models is intended to map to many actual neurons, will not solve this problem. There is little evidence that cortex is highly sensitive to the destruction of hundreds or even several thousand neurons. Noticeable deficits usually occur only when there are very large lesions, taking out several square centimeters of cortex (one square centimeter is about 10 million neurons). If such considerations are not persuasive to proponents of these models, the best way to demonstrate that the models are robust is to run them while randomly destroying parts of the network, and introducing reasonable levels of noise.

ACT-R's commitment to symbolic representations suggests that it too will not be especially robust. Of course, it is difficult to evaluate this claim by considering how the destruction of neurons will affect representation in ACT-R. But there are other kinds of robustness to consider as well. As I have mentioned, it is a concern of proponents of the approach that they do not support partial matching (section 9.1.1). This means, essentially, that there cannot be subtle differences in representations – the kinds of differences introduced by noise for instance – which are fixed by the production matching process. Or, put another way, there cannot be “close” production matches which could make up for noise or uncertainty in the system. So, ACT-R is robust in neither its processes nor its representations.

The robustness of the ICS is difficult to judge. If the system were consistently implementable in a fully distributed manner, it should be quite robust. But, because control structures are not provided, and most of the actual processing in current models is done in localist networks, it is not clear how robust an ICS architecture would be. So, while the distributed representations used in tensor products should be highly robust to noise, the effects of noise on the ability of the system to solve its optimization problems in distributed networks has not been adequately considered. As a consequence, we might expect that an ICS system could be quite robust, but little to no evidence is available that supports that expectation, especially in the context of a reconfigurable, distributed system.

We can be more certain about the robustness of DFT. Attractor networks are well known to be highly robust to noise, and reasonably insensitive to the disruption of individual nodes participating in the network (Conklin and Eliasmith, 2005)???add Macneil ref???. It is likely, then, that the dynamics intrinsic to most DFT models will help improve the robustness of the system overall. One concern

with robustness in DFT models, is that it is well known that controlling dynamical systems can be difficult. Because there are no “control principles” available for DFT, it would be premature to claim that any DFT model will be robust and stable. So, while the representations seem likely to be robust, it is difficult to determine, in general, if the processes will be. Essentially the architecture is not well-specified enough to draw general conclusions.

Similarly, for Leabra, the reliance on attractor-like working memories will help address robustness issues. In many Leabra models, however, the representations of particular states are localized. Because the models are seldom tested with noise and node removal, it is again difficult to be certain either way about the robustness of a scaled-up version of these models. The current theoretical commitments, at least, do not preclude the possibility that such models could be robust.

Overall, then, state-of-the-art approaches to biological cognition do not do especially well on this criterion. It seems that systems are either highly unlikely to be robust, or there is little positive evidence that generated models would continue to function under reasonable disturbances due to neuronal death, input variability, or other sources of noise.

c. Adaptability As mentioned in my previous discussion, adaptability comes in many different guises (section 8.2.2.3). Syntactic generalization, for example, is one kind of adaptability, in which syntactic representations are used to reason in a new circumstance. As discussed earlier, “in principle” syntactic generalization is captured by all of the classical representational approaches. However, rapid generalization, exemplified in the problem of rapid variable creation (see section 7.3.7), is not well-handled by any of these approaches.

Another obvious form of adaptability is learning. Many of the state-of-the-art approaches consider learning of one kind or another. ACT-R models, for example, are able to generate new productions based on the use of past productions in order to speed processing. This is done through a kind of reinforcement learning (RL) rule. Leabra, too, has much to say about reinforcement learning, and hence is quite adaptable in this sense. Because DFT has focused on models of child development, it too has dealt explicitly with the adaptability of the system it is modeling. And, finally, LISA includes a kind of learning in order to extract schemas from past experience. Only the ICS and the NBA have little to say about adaptability in this sense.

Of course, as indicated in my discussion in section 6.4, there are many differ-

ent kinds of learning to be considered when understanding biological cognition. While many of these approaches have something to say about learning, they often address only one specific kind of learning. (e.g., RL in ACT-R, Hebbian learning in DFT, and schema extraction in LISA). Ideally, many of these different kinds of learning should be integrated within one approach. On this count, Leabra is notable for its inclusion of Hebbian, RL, and backprop-like forms of learning.

One other adaptability challenge that is especially salient for LISA, the NBA, the ICS, and DFT is that of task switching. Because these approaches are silent regarding the operation of a control system that can construct, ignore, and transform representations, they have little to say about the evident ability of cognitive systems to move seamlessly between very different tasks. Each of these approaches tends to construct different models for different kinds cognitive phenomena, or different tasks. Hence, it is unclear to what degree such approaches will support this kind of observed adaptability. This may also be a concern for Leabra and ACT-R. Even though control is given serious consideration in Leabra, it is often limited to a highly specific task. Similarly, ACT-R models are usually reasonably focused. Of course, this concern comes in a matter of degrees. The highly complex tasks of some ACT-R models (e.g. driving), may involve completing many sub-tasks.

In conclusion, I take it that state-of-the-art models have a lot to say about adaptability. However, none of the approaches captures both the variety of timescales, and mechanisms that have been observed in biological cognition to be responsible for adaptation. One adaptation problem that provides a significant challenge for all considered approaches is the problem of rapid variable creation.

d. Memory Out of the many kinds of memory, I highlighted long-term and working memory as important contributors to cognition. I noted that one role of long-term memory is to support symbol grounding, and hence semantic relations, and that one role of working memory is to support occurrent cognitive processing.

The ACT-R explicitly includes each of these kinds of memory in the distinction between its “buffers” and “declarative memory” systems. Indeed, ACT-R has been used to describe many different kinds of memory phenomena. Consequently, it does well on this criterion. However, one central drawback that remains in ACT-R’s account of long-term memory lies in its inability to relate complex perceptual long-term memories to the system’s understanding of the world. The declarative memory that ACT-R relies on includes many facts about the world that can be stated symbolically. These are typically determined by the programmer of the

model. However, what this misses is how the symbols used in those descriptions relate to the perceived and manipulated world of the system. That is, there is no representational substrata which describes, for example, a model of visual properties that capture the variability and subtlety of perceptual experience. In other words, the account of long-term memory provided by ACT-R is partial and semantically superficial.

LISA and the NBA both have working memory-like mechanisms. In the NBA these mechanisms are used to support temporary binding of symbols. In LISA, they account for the capacity limitations of working memory observed in people. However, both have poor accounts of long-term memory. LISA's minimal semantic networks could be considered a kind of long-term memory. However, they do not capture either retrievable facts about the world, or more subtle perceptual representations like those missing from ACT-R. The NBA does not discuss long-term memory at all. The ICS discusses neither.

Leabra and DST are both centrally concerned with working memory phenomena. As a result, they have good characterizations of working memory in a biologically-inspired substrate. Both have little to say about long-term memory. This, perhaps, should be qualified by the observation that Leabra is able to learn procedures for manipulating its representations, and these may be considered long-term memories. As well, DFT considers the development of various kinds of behavior over time that would presumably persist into the future. In both cases, however, these are clearly very limited forms of long-term memory when compared to the declarative system of ACT-R, or to the kinds of perceptual and motor semantic representations that are crucial for central aspects of biological cognition.

e. Scalability Geoff Hinton recently wrote: “In the Hitchhiker’s Guide to the Galaxy, a fearsome intergalactic battle fleet is accidentally eaten by a small dog due to a terrible miscalculation of scale. I think that a similar fate awaits most of the models proposed by Cognitive Scientists” (Hinton, 2010, p. 7). In short, I agree.

I highlighted several challenges relating to scalability in section 8.2.2.5. These included knowing the dimension along which to evaluate scaling, knowing how to account for interactions in functional systems, and being able to establish scalability in practice, rather than only in principle. I have obliquely addressed some of these issues in my discussion of previous criteria. For instance, the massive binding problem clearly relates to the ability to scale binding assumptions up to

complex representations. Similarly, my concerns about the ability of many approaches to be able to address multiple tasks in a single model is closely related to issues of accounting for interactions in a complex system. In both cases, I was somewhat pessimistic about the probable success of current approaches.

In the context of the previous analysis of scalability, this pessimism is reinforced by the inability of several approaches to even meet “in principle” scalability constraints. For example, each of LISA, the NBA, and the ICS seem unlikely to be able to support representations of minimal complexity over a human-sized vocabulary of about 60,000 words without overtaxing available resources and the human brain (see appendix B.5). DFT and Leabra do not address binding issues in any detail, and hence it is difficult to be optimistic about their abilities to scale as well. While ACT-R can claim to be able to encode many complex representational structures, the fact that attempts to relate those representational assumptions to a neural architecture have failed, do not bode well for its scalability in the context of the physical resources of the brain.

As discussed in the section on adaptability, it is also difficult to be optimistic about the scalability of the functional architectures associated with most of the approaches. The ICS, LISA, the NBA, DFT, and Leabra do not specify the majority of the control processes necessary for implementing a functional system that is able to perform many cognitive tasks in a single model. Again, on this count ACT-R is in a much better position. This is not only because ACT-R has been applied to more complicated tasks but also because most other approaches do not provide principles that specify how they *could* be applied to complex tasks.

Turning to consideration of “in practice” scalability makes it clear how far we have to go in our understanding of biological cognition. As I’ve already discussed, many of these approaches construct different models for each different task to which they are applied. Constructing systems which can perform many different tasks using the exact same model is largely unheard of. And, as the story of Tusko the elephant will remind us, thinking that “more of the same” will solve this problem is a mistake.

This is perhaps the performance criterion with respect to which the most progress remains to be made.

9.2.3 Scientific merit

Criteria related to scientific merit are less often discussed in cognitive science than might be expected from a scientific discipline.² Perhaps this is because it is all too obvious that cognitive theories should respect such criteria. Or perhaps it is deemed too difficult to effectively address such criteria early in the discipline's development. Or perhaps both. In any case, it should be clear that, as Taatgen and Anderson (2010) put it: "Eventually, cognitive models have to live up to the expectations of strong scientific theories" (p. 703).

a. Triangulation (contact with more sources of data) Proponents of ACT-R have been explicitly concerned with trying to improve the ability of their methods to contact more sources of data. Their most successful efforts have been aimed at relating various models to fMRI data, as I described earlier. However, this same motivation lies behind recent attempts to integrate ACT-R and Leabra, and the general interest in understanding how ACT-R relates to the details of the neural substrate.

It is clear that ACT-R is well constrained by behavioral data such as reaction times, choice curves, eye movements, and so on. But, as I discussed earlier (section 9.1.1), despite the efforts undertaken to relate ACT-R to neural data, the quality of that relationship is questionable. For instance, by far the majority of brain areas are not related to elements of the ACT-R architecture. As well, the number of free parameters used to fit fMRI data is often very high. And, the mapping is not one based on a physiological property of the model (e.g., the activation level of a module), but rather a temporal one (amount of time spent using a module). Essentially, brain areas are treated as being on or off.

Perhaps we can dismiss these as minor concerns that will be addressed by future work on ACT-R. None of these are, after all, fundamental flaws. But, a deeper problem lies in the fact that it is unclear how *other* neural data can be related to ACT-R models. With a commitment to symbolic representation, and no theory on how such representations relate to single neuron activity, most fine spatial and temporal scale methods cannot be related to ACT-R models. Given the increasing power and sophistication of such methods this does not bode well for the future of ACT-R. Perhaps this explains the urgency with which proponents are pursuing methods of overcoming this major conceptual hurdle.

²Although those who agree with Searle's (1984) quip – "A good rule of thumb to keep in mind is that anything that calls itself 'science' probably isn't" (p. 11) – may be less surprised.

While ACT-R proponents have not embraced a particular implementational story, many such stories have been offered: for example by LISA, the NBA, and the ICS. However, the neural implausibility of each of these approaches makes them unlikely to be able to ultimately provide a connection between detailed neural methods and a classical representational system like ACT-R. All suffer from over-taxing available neural resources. As well, to the best of my knowledge, none have made explanatory or predictive connections to detailed neural data. The NBA, for example, has not been connected to high-resolution MEG, fMRI, LFP or neural array experiments that might provide insight into their proposed connectivity. LISA has not predicted or explained expected patterns of synchrony during various tasks. The ICS has not been correlated with MEG, fMRI or EEG experiments run during language processing. And, none have been tested against the plethora of data available from experiments on related mechanisms in non-human animals.

Many of these same criticisms can be levelled against DFT. While the models are run on abstract “neural sheets”, and plots of the dynamics of neural activity can be provided, these are generally not compared directly to available neural data (although see Jancke et al. (1999)). For example, there have been many single cell experiments performed during working memory tasks in various mammals. Such experiments do not provide evidence for the idealized homogeneous responses and highly stable dynamics found in DFT models. Indeed, a recent theme in some of this work is to explain why the dynamics in working memory areas are so *unstable* and why neural responses are so *heterogeneous* (Romo et al., 1999b; Singh and Eliasmith, 2006). Similarly, many of the developmental processes addressed by DFT models are observed in non-human animals as well, so relevant single cell data is often available for such models. Nevertheless, it is uncommon to find an explicit connection between these models and the neural data.

By far the most biologically driven of the approaches I have considered is Leabra. This is nowhere more evident than in the consideration of its contact with neural data. As I described (section 9.1.5), Leabra has been used to predict the effects of different kinds of neurotransmitter agonists and antagonists. It has also been used to predict the effects of aging and disease on cognitive function due to variations in underlying neural mechanisms. It has also been employed to explain higher-level neural data like fMRI. And, unlike ACT-R, Leabra neural activity is used to match to fMRI activation (Herd et al., 2006).

That being said, Leabra is not able to match detailed single cell data because it models “average” neurons, and so is missing the observed heterogeneity. This is crucial because it is becoming increasingly evident that the precise distribution

of neural responses can tell us much about the function of a population of cells (Eliasmith and Anderson, 2003; Machens et al., 2010, chp. 7). As well, Leabra uses rate neurons, and so is unable to provide spike timing data. Relatedly, concerns I voiced earlier about the use of kWTA in place of actual neural dynamics make it less able to address a variety of dynamical neural phenomena in any detail. Nevertheless, Leabra is clearly more directly related to neural methods than other approaches, save, perhaps, DFT.

For both DFT and Leabra, their stronger connection to low-level neural data has been achieved at the price of being less able to explain higher-level behavior. This can be seen in the desire of proponents of Leabra to connect their work to ACT-R. The enviable breadth of application of ACT-R is not achievable within Leabra because it does not provide theoretical principles that determine how the low-level activity patterns in Leabra relate to the symbol manipulation exploited in ACT-R. This is true despite the explicit attempts to marry these two approaches I mentioned previously. Similarly, in DFT the “most cognitive” behaviors addressed are identification of simple spatial relations from visual stimuli, and the tracking of occurrent objects. These are not comparable to the “most cognitive” behaviors addressed by the likes of ACT-R.

In sum, there are no state-of-the-art approaches that convincingly connect to most of the relevant kinds of data regarding brain function. Instead, most approaches have a preferred kind of data and, despite occasional appearances to the contrary, are not well-constrained by other kinds of data. Not surprisingly, since I am considering cognitive approaches, behavioral data is by far the most common kind of data considered. However, Leabra and DFT stand out as being better connected to neural data than the other approaches. Unfortunately, this seems to have resulted in their being less well-connected to the high-level behavioral data.

b. Compactness Perhaps all of the considered approaches should be praised for their compactness since they are each stated rigorously enough to be simulated on computers. This is not a common characteristic of much theorizing about brain function (though it is, thankfully, becoming more so). Nevertheless, there are clearly differing degrees to which the considered approaches are rigorous, systematically applied, and non-arbitrarily mapped to the brain.

I believe ACT-R is the most compact of current available theories. It is quantitatively stated, it has some central, unvarying parameters (e.g., 50ms per production), and the mapping of the mathematical description to the brain is clearly stated. In addition, its description of control structures and straightforward rep-

resentational commitments mean that different applications retain central features of the architecture. Perhaps the greatest source of arbitrariness stems from the fact that the declarative memory, which contains the relevant rules and facts, is specified by the modeller. It has been suggested that this essentially provides the architecture with unlimited flexibility (Schultheis, 2009). However, that is only a problem if other commitments of the architecture do not limit this flexibility, but they do. The 50ms assumption provides some limitations on what can be done within a given time frame, assuming the complexity of productions are kept within reason. As well, if we insist on the specification of models that are able to perform many tasks without changing the model, interactions between components will force some degree of consistency amongst specified rules. Unfortunately, different ACT-R models are seldom integrated in this manner. Nevertheless, the architecture provides the resources necessary to do so.

Approaches such as LISA, the NBA, the ICS, and DFT, which do not specify an internal control structure, are in a difficult position with respect to compactness. These methods currently need the modeller to specify a wide variety of elements prior to modelling a particular task. LISA requires a specification of the wiring to support the relevant bindings and processing, as well as the representations and semantics needed for a task. The NBA and the ICS require comparable specifications. Similarly, DFT requires specification of the number and interaction of the attractor networks, and their relation to input and output stimuli. Of course, specifying things is not the problem. The problem is that these are often different from model to model because the architecture does not provide general principles for doing so. Indeed, some of such specifications are more principled than others. For instance, mapping from a harmony network to a distributed equivalent is mathematically defined in the ICS. In contrast, the particular choice of subsymbolic nodes to specify semantics, and the particular wiring to perform a task in LISA and the NBA is left to the modeller. Of course, these criticisms could be addressed if there was a single, canonical model that captured performance on many tasks as once. Then, one specification of these aspects of the architecture would suffice for a broad range of tasks, and the charge of "arbitrariness" would be weak. As a result, such concerns are clearly related to the scalability issues I discussed in more detail earlier.

Leabra has something of a more principled control structure, though at a very high level of abstraction. Nevertheless, the coupling of Leabra based models, such as PVLV and PBWM is to be commended for being systematically used across a variety of models. Attempts are clearly made to keep the many parameters of these models constant, and basic structural assumptions generally remain consis-

tent. I have suggested that ACT-R is more compact than Leabra largely because judgements of compactness depend also on the breadth of application of a theory. Because Leabra does not address a wide variety of cognitive behaviors accessible to ACT-R, but ACT-R does heavily overlap with Leabra in its account of reinforcement learning (though Leabra's account is more thorough), ACT-R seems to be more compact overall. In addition, ACT-R is more systematic in its mapping to temporal phenomena.

When turning to consideration of temporal phenomena, in general, most of the considered approaches are disconcertingly arbitrary. That is, perhaps the greatest degree of arbitrariness in all of the current accounts is found in their mappings to time. There are no obvious temporal parameters at all in some accounts, such as the ICS and the NBA. Other approaches, like LISA and Leabra are prone to use "epochs" in describing temporal phenomena. Unfortunately, different models often map an "epoch" onto different lengths of "real" time, depending on the context. As well, Leabra relies on basic computational mechanisms that avoid time all together (e.g., kWTA). In contrast, DFT is fundamentally committed to the importance of time. The difficulty comes when we again consider how the time parameters in DFT models and real time are related. As with LISA and Leabra, the DFT does not specify principles for time-constant parameters are related to real time, and so different models are free to map model time to real time in different ways. Finally, ACT-R is less temporally arbitrary because of its insistence on always using 50ms to be how long it takes to fire a production. However, as I hinted above, the complexity of productions may vary. This means that very simple or very complex rules may be executed in the same length of time. Without a further specification of some constraints on the complexity of productions, the ACT-R mapping to time is not as systematic as it could be.

To conclude, while all of these approaches are to be commended for rigorously specifying their models, ACT-R and Leabra are the most compact. This stems from greater evidence of their providing a systematic mapping between components of the theory and the system they are modelling. Nevertheless, all approaches are somewhat arbitrary in their relation to real time.

9.2.4 Summary

Evaluating past approaches with respect to the QCC helps make clear the kinds of trade-offs different approaches have made. It also helps highlight general strengths and weaknesses that there appear to be in the field as a whole. I have attempted to summarize my previous discussion in table 9.1, though much subtlety

Table 9.1: Comparison of past approaches with respect to the QCC, rated on a scale from 0 to 4.

	ACT-R	LISA	NBA	ICS	DFT	Leabra
Systematicity	++++	++++	++++	++++	++	++
Compositionality	++	++	++	++	-	-
Productivity	++++	+++	+++	+++	-	-
Massive binding	+++	+	+	+	-	-
Syntactic Generalization	++	++	++	++	-	-
Robustness	-	+	+	++	++	++
Adaptability	++	+	-	-	+	++
Memory	+++	++	+	-	++	++
Scalability	+	-	-	-	-	-
Triangulation	+	+	+	+	++	++
Compactness	++	+	+	+	+	++

is lost.

Nevertheless, it is possible to see some general trends from this table. It is clear for instance, that symbol-like representation and biological detail remain somewhat orthogonal strengths, as they have been historically. It is also evident that each approach would greatly benefit from pursuing larger-scale models, in order to more effectively address several criteria including not only scalability, but also adaptability, massive binding, and compactness. These same large-scale models could serve to demonstrate the robustness of the approaches, another criterion on which most approaches do poorly.

Finally, this analysis also suggests that the general criteria for good scientific theories could be much better addressed by current approaches. This is an observation that, as I noted earlier, has been made by other theorists in the field. This observation is also consistent with a sentiment sometimes encountered among non-modelers, who suggest that there are “too many free parameters” in most brain models. The more compact and data driven theories of cognition become, the more unreasonable such off-the-cuff dismissals will be.

One of my main motivations for providing this analysis, and the QCC in general, is to help situate the SPA in the context of past work. While presenting the SPA, I have attempted to highlight some similarities and differences between the SPA and other approaches. In the next two sections, I have gathered and related these observations to the QCC in an attempt to give a fuller picture of how the

SPA fits into the landscape of current cognitive approaches.

9.3 The same...

The approaches I have been considering for the last several pages span the range of paradigms on offer over the last 50 years of cognitive science. While I have picked the more biologically relevant approaches from each of symbolism, connectionism, and dynamicism, the main theoretical commitments of all three approaches are well-represented in what I have surveyed. As I now turn to a consideration of the approach on offer here, it may already be obvious that the SPA is all and none of these past approaches. Indeed, my earlier plea to “move beyond metaphors” (section 1.2) intentionally opened the door the possibility that no one paradigm would turn out to be the right one. In this section, I survey the many similarities between each of these past approaches and the methods and commitments of the SPA. This demonstrates that the SPA bears important similarities to much past work, making clear that it is an extension of much of what has gone on before. In the next section, I emphasize the differences.

With LISA, the NBA, and the ICS, the SPA shares a commitment to providing a neurally plausible account that captures symbol-like representations. In all four cases, there is an emphasis on the importance of binding to construct structured representations that can then support a variety of syntactic and semantic manipulations. In fact, the preferred mechanism for binding in the SPA comes from the same family as that employed in the ICS. As a result, many of the mathematical results available regarding the ICS are quite informative about representational and transformational resources available to the SPA. Relatedly, the ICS offers a sophisticated story regarding how such representations can be related to a wide variety of linguistic processing. The representational commitments of the SPA should be largely consistent with that characterization.

The SPA shares a central commitment to dynamics with DFT. It is no surprise, then, that attractor networks form a basic computational component that is re-used throughout both architectures. This concern for dynamics connects directly to interest in perception-action loops, and accounting for the integration between these aspects of behavior, and more traditionally cognitive aspects. There is little in the SPA that could not be reformulated using terms more familiar to

dynamic systems theorists. Control theory and dynamic systems theory use many of the same mathematical tools, concepts, and methods, after all. The common use of temporally-laden terminology belies this deeply shared commitment to understanding biological cognition through time.

Turning to Leabra, the terminology becomes drawn heavily from neuroscience. This central interest in the biological basis of cognition is shared with the SPA. The similarities between Leabra and the SPA go further, however. There is a general agreement about some basic architectural features. Both consider the cortex-basal ganglia-thalamus loop to play a central role in controlling cognitive function. Both consider the biological details of this architecture when discussing learning of control, as well as past learned behaviors. Not surprisingly, then, both have similar relationships to neural data regarding the dopamine system. It is clear that Leabra has much to recommend it, and the SPA has been heavily influenced by this, and related approaches to basal ganglia function (e.g., Gurney et al., 2001). Overall, the SPA and Leabra share a concern with the biological implementation of adaptive control in cognition.

ACT-R and the SPA share a basic interest in high-level cognitive function. Both propose ways that rule-like behavior, resulting from symbol-like representation, can occur. They share, with Leabra, a commitment to the cortex-basal ganglia-thalamus loop as being the central control structure in explaining such behavior. All three see a similar place for reinforcement learning, and hence relate to similar learning data in some respects. As well, both the SPA and ACT-R have something like rule matching at the heart of that adaptive control structure. A related interest of ACT-R and the SPA is found in their focus on the timing of behaviors. Specifically, behavioral-level reaction times are relevant to the functioning of both architectures. Finally, both approaches are very concerned with understanding the mapping of the cognitive architecture to the underlying neural structures. Hence, both are interested in relating their approaches to the biological basis of cognition.

Given these similarities, there are many ways in which the SPA can be considered a combination of commitments of the ICS, DFT, ACT-R, and Leabra. While ACT-R and Leabra alone do not seem to have been broadly integrated, despite attempts to do so, adding the ICS to the attempt provides an important extra ingredient to that integration. The challenge, after all, was to relate the representations of ACT-R to those of Leabra. The ICS provides a rigorous and general mapping between the two. The basic control structure is similar between ACT-R and Leabra, and the ICS and DFT are silent on that issue, so there are no direct conceptual conflicts. The SPA can thus be seen as an attempt to build the neces-

sary bridges between these approaches in a constructive and neurally responsible way, while incorporating some conceptual insights from DFT regarding dynamics and embodiment.

In the first chapter, I suggested that successfully breaking from the dominant metaphors in cognitive science may lead to a new kind of theory, as happened in the wave-particle debate in physics. The intent of such an integration is to not only adopt the strengths of each approach, but to overcome the weaknesses. In the next section, I describe how I believe the SPA accomplishes this, by focusing on the differences between the SPA and past approaches. My intent is to suggest that the result is more than the sum of its parts.

9.4 ... but different

While the SPA shares an interest in understanding symbol-like behavior in a neural substrate, it takes a notably different approach from each of LISA, the NBA, and the ICS. It is, as I mentioned, most similar to the ICS, but the difference lies in the fact that the multiplicative binding is always compressed in the SPA. As I discussed in section 9.1.4, this is a crucial difference between the SPA and the ICS. In fact, this is a difference that helps distinguish the SPA from past attempts at implementing symbolic architectures in general. This is because compressive binding guarantees that the SPA does not implement a classical symbolic representational system (see section 4.2.1).

Because of the compression, there is a loss of information in the encoded representation. This means that there is no isomorphism between the original symbolic structure, and the elements of the compressed semantic pointer. Without that isomorphism, classic compositionality does not hold, and there is no clean implementational relationship between the SPA representations and a classical symbolic representation. In short, I have been careful to note that they SPA traffics in “symbol-like” representations, not symbols. They are like symbols in their ability to encode structure, but unlike symbols in their compositional characteristics. It is exactly this approximation that allows the SPA to scale much more plausibly than any of the other approaches (see sections 4.2.2 and 4.5, and appendix B.5). The price, of course, is a decrease in accuracy of reconstructing what was encoded. This puts important constraints on the depth of structure that can be encoded. But, as mentioned in my description of the QCC, those same kinds of constraints are relevant for all of the representational criteria: productivity is limited, compositionality is partial, and systematicity is restricted.

This lack of an isomorphism between the neural implementation and the symbolic-

level description means that the ICS and the SPA embody different conceptualizations of the relationship between levels of description in cognitive science. I have described both conceptions in some detail (sections 9.1.4 and 2.4 respectively), and the difference essentially lies in the “messy” mapping between levels allowed by the SPA. This messiness means that clean isomorphisms are unlikely, and that neural details will matter for certain kinds of behavioural accounts (e.g., recall the example of serial working memory). Nevertheless, higher-level descriptions can be helpful, if approximate, for answering certain kinds of questions about cognitive function (e.g., what rule was induced in the Raven’s task?). Of course, the neural level description is an approximation to a more detailed model in a similar manner. The pragmatic aspects of “descriptive pragmatism” about levels (see section 2.4) means that the questions we are asking of a model will help determine what the appropriate level of analysis and simulation of the model is. Like the ICS, the SPA supports descriptions across a range of levels, but not in virtue of an isomorphism. And, I should add, the SPA covers a wider range of levels than does the ICS.

While the non-classicism of the SPA approach helps distinguish it from the NBA and LISA, there is perhaps a deeper difference. This is the reliance of the SPA on the “moving around” of representations within the architecture. In the NBA and LISA, the representation of specific items is fixed to a specific part of the network. In a localist network, this describes the standard method of having the “dog” representation be a single node. In the distributed case, this would be some specific subset of nodes being the “dog” representation. In the SPA, the representation of “dog” can be in many different networks at the same time, or at different times. This is because that representation can (often, though not necessarily exclusively) be found in the activity patterns of a population of cells. Populations of cells represent vector spaces, and many different items can be represented in a given vector space. At any point in time, the specific representation in a population depends on transient activity states.

In a way, the SPA characterizes neural representations as being somewhere between standard computer representations, and standard implementations of neural network representations. Standard computers are powerful partly because they do not commit a particular part of their hardware to specific representations. Thus, the same hardware can process representations about many different things. It is the *function* of the hardware, not its representations, that are often taken to define its contribution to the overall system. If we adopt the NBA or LISA style methods of representation, then specific populations or nodes are committed to particular representations. The SPA commits hardware to a specific vector space, but not a

specific point in a specific labelling of a vector space. This helps it realize some of the key flexibility of traditional computers in being able to represent many things with the same hardware (and represent the same thing with many different bits of hardware). Consequently, the same neural population can compute functions of a wide variety of representations. This is a subtle yet important difference between the SPA and many past connectionist approaches. It is because of this flexible use of neural populations that the SPA can support control and routing of representations. This is an aspect of cognitive function conspicuously absent from LISA and the NBA.

In the previous section I highlighted the shared emphasis on dynamics and embodiment between DFT and the SPA. This is a conceptual similarity, but the practical applications are quite different. The SPA is regularly compared to, and constrained by, detailed single cell spiking dynamics. As well, the dynamic components of the SPA are intended to be actual neurons, mapping well to the observed heterogeneity, tuning, and response properties of these elements of the real system. As such, the parameters of the nodes are measurable properties of the brain. DFT, in contrast, does not provide such a specific mapping to such aspects of the physics and functions of neurons.

The SPA also has a broader characterization of the dynamics and interaction of the perceptual, motor, and cognitive systems. A central reason for this is that the SPA characterizes the dynamics of much higher-dimensional spaces than DFT typically does. Because the SPA distinguishes between the state space, and the neuron space in which the state space is embedded, it is possible to understand the evolution of a very complex neuron space with a few degrees of freedom in the state space. This mapping then makes it possible to extend the dimensionality of the state space to even higher dimensions, while keeping the principles that map dynamics to neural representation the same. In many ways, the SPA working memory of a 100-D semantic pointer is a straightforward extension of the 2-D working memory of a DFT sheet of neurons (Eliasmith, 2005b). However, the computations and dynamics available in that higher-dimensional space can be much more sophisticated. Specifically, the principles of the NEF allow explicit nonlinear control to be introduced into these dynamical systems, which opens a host of important possibilities for constructing networks with useful dynamics (e.g., for routing, gain control, binding, controlled oscillation, etc.).

As well, the NEF provides the SPA with methods that allow such systems to be constructed without the Hebbian learning typically relied on in DFT models. This allows much more complicated systems to be constructed, and compared to data from developed cognitive systems. To address developmental questions, learning

must be used. As I have emphasized in chapter 6, many varieties of biologically plausible learning are also available in the SPA.

Learning is a central focus of Leabra, and hence a point of comparison between the SPA and Leabra. As I have already mentioned, both approaches allow for the characterization of a wide variety of learning results. As well, much of the control architecture is shared between the SPA and Leabra. The main differences between the approaches lie at the extremes of levels of analysis. At the level of individual neurons, the SPA offers a characterization of neural dynamics and individual spike responses not available in Leabra. And crucially, the SPA does not rely on computational approximations like kWTA to produce such predictions. Instead, the detailed dynamics of, for example, basal ganglia function and adaptation arise out of the low-level dynamics of SPA networks, which are governed by various kinds of specific neurotransmitters and receptors. As well, the SPA more naturally incorporates the observed heterogeneity of neural responses.

At the opposite end of the spectrum, the SPA provides an explicit account of symbol-like representation not available in Leabra. This account is integrated with the neural details, the shared control structures, and the shared aspects of learning. It also solves the “interface” problem that has confronted the attempted integration of Leabra with ACT-R (Jilk et al., 2008, p. 213). After all, there is no interface between “symbolic” semantic pointers and “perceptual” or “motor” semantic pointers: semantic pointers are semantic pointers. In many ways, the SPA provides to Leabra what ACT-R was intended to bring to the integration of these two approaches.

I have described many of the similarities between the SPA and ACT-R, but perhaps the differences are most obvious. The main difference is clearly commitments to different representational substrates. For ACT-R, the problem of mapping symbols to neurons looms large. For the SPA, it was a starting point. Consequently, relating the models produced by the SPA to specific neural details comes much easier to the SPA than to ACT-R, and I believe this conceptual benefit of the SPA outweighs the thirty year head start ACT-R has with respect to building actual cognitive models. Interestingly, many subtle differences result from this switch in representational assumptions.

For instance, the SPA allows for a more flexible and powerful characterization of rule application, that I have discussed in detail above (section 7.4). This is partly because it explicitly addresses the “partial matching” problem that ACT-R is faced with, which the integration with Leabra was intended to help address. In short, the switch in representations has changed *how* rules can be expressed (i.e., as vector transformations) what *kinds* of rules can be expressed (i.e. more

statistical mappings), and how available rules can be *matched* to current context (i.e., partial matching). In applying the SPA, we have often found that standard variable-binding rules used in ACT-R are often not necessary in SPA models. For instance, in section 5.6, we employ many rules that simply copy information between elements of the architecture, rather than explicitly including variables in the rules themselves. If this kind of approach proves generally useful, it may prove an important change in how cognitive rules are characterized.

It is also interesting to notice how the SPA's representational story relates to a conceptual difference between how ACT-R and the SPA treat levels. ACT-R proponents, unlike those of the ICS, suggest that there is a "best" level of description for cognitive systems: "In science, choosing the best level of abstraction for developing a theory is a strategic decision...in both [symbolicist and connectionist] cases, the units are a significant abstraction... I believe ACT-R has found the best level of abstraction [for understanding the mind]" (Anderson, 2007, p. 38-39). While Anderson leaves open the possibility that the best level is not yet available, it seems to me to simply be a mistake to expect that there is *a* best level for understanding the mind. In contrast with the notion of descriptive pragmatism I introduced earlier, the notion that there is a single best level will always cry out for an answer to the question: "The best for what purpose?" If this purpose changes, even slightly, so will the most appropriate level of description. I believe the SPA's focus on relating levels, rather than picking levels, allows descriptions to be systematically adjusted to meet the relevant purposes for a given question about cognition. Sometimes the best description will demand reference to single cell spike patterns, or a characterization of the effects of specific neurotransmitters. Other times, the best description will demand the specification of which rule was applied, or what sentence was generated by the system. A unified approach, able to relate many such descriptions across many different levels strikes me as the goal of most scientific enterprises. I am not alone: for instance, Craver (2007) has argued in detail that good explanations in neuroscience should contact many specific mechanisms at multiple levels of descriptions, and across disciplinary fields (see also Bechtel and Richardson, 1993; Bechtel, 2005; Thagard and Litt, 2008).

In general, I take it that the SPA spans the relevant levels of description in a more convincing manner than past approaches, including ACT-R. A major reason that the SPA can employ highly detailed neural representations as well as abstract symbol-like representations, is that the methods for characterizing representation are flexibly specified. For instance, the NEF mapping of vector spaces on to neurons does not impose any specific form on the neural nonlinearity. While most models I have presented use a simple spiking leaky integrate-and-fire model, the

complexity of useful single neuron models in the SPA is limited only by the researcher's knowledge and available computational resources. In previous work, we have used adapting LIF neurons (Singh and Eliasmith, 2006), reduced bursting neurons (Tripp and Eliasmith, 2007), and detailed conductance-based neurons (Eliasmith and Anderson, 2003). The fundamental approach remains the same. Similarly, the amount of detail in the synaptic model is not constrained by the NEF approach. Consequently, if deemed relevant, a research can introduce uncertain vesicle release, saturating synaptic conductances, axonal delays, etc. In most of our models we introduce a generic noise term to account for these kinds of uncertainties, but the precise mechanisms leading to such variability can be included if necessary for a desired explanation.

Much of this breadth stems from its adoption of the NEF, which has been, and continues to be, centrally concerned with neural plausibility. For instance, in previous work it has been shown that any network constructed with the NEF can be made to be consistent with Dale's Principle³, hence capturing a major architectural constraint ignored by much past modelling (Parisien et al., 2008). In addition, a wide variety of single-cell dynamics, including adaptation, bursting, Poisson-like firing, etc., has been directly incorporated into the methods of the NEF, helping to make it generally applicable to cortical and subcortical systems (Tripp and Eliasmith, 2007). The continuing development of a neurally responsible theoretical substrate for the SPA means that the SPA characterization of biological cognition continues to incorporate ever more subtle aspects of biology.

Each such added detail improves the realism with which the model's mechanisms approximate those that have been characterized by neuroscientists. And, each such added detail opens a field of neuroscience from which data can be drawn to be directly compared to the model. Furthermore, many such details result in changes in neural spike patterns, one of the most commonly measured properties of neural systems. Thus, model results from the SPA approach are directly comparable to one of the most common sources of neural data. Furthermore, given our understanding of the relationship between single cell activity and other methods for recording neural signals, we can use these single cell models to predict other kinds of data, such as that gathered from local field potentials (LFPs), electroencephalograms (EEGs), fMRI, and so on. Essentially, different kinds of filters can be used to process the model data and provide comparisons to other methods of measuring brains (see, e.g., section 5.8).

³Dale's Principle states that the vast majority of neurons in the brain are either inhibitory or excitatory, not both.

So, in contrast to past approaches, there is no settled “level of abstraction” at which cognition must be understood in the SPA. Rather, which details are needed for which explanations remains flexible: determined by the question that is asked, and hence the kind of answer that is most relevant. If the details of neurotransmitter release are relevant, they can be incorporated. If purely behavioral predictions are most relevant, they can be provided. I believe there is no other approach currently on offer than provides this breadth of coverage of the brain sciences.

To conclude this chapter, I will return to the QCC with which I started. Referring to table 9.1, it is possible to see how the SPA fares compared with the other approaches. I have suggested that the SPA might be considered an amalgamation of bits and pieces of past approaches. But I believe returning to the QCC shows how the whole is greater than the sum of its parts.

When it comes to Systematicity, compositionality, and productivity, the SPA scores at least as well as ACT-R. This is because its representational commitments allow it to capture the same idealizations of representational capacity as any other approach, while not violating scalability constraints. I would argue that it does a better than past approaches because, in not violating those constraints, it imposes reasonable limits on the idealizations of such criteria. The same considerations apply to the massive binding criterion: the SPA alone makes sense of how the brain can meet such constraints while not violating scalability (see, e.g., section 4.7).

For syntactic generalization, the SPA, like ACT-R, provides a mechanism that can reproduce the speed with which such problems are solved by people (see section 7.3.7). Unlike ACT-R, the SPA provides a natural explanation of content effects that follows directly from its account of semantics (see section 6.6).

The robustness of past approaches is highly variable, and often poorly tested. The SPA allows standard demonstrations of robustness by the random destruction of neurons (see figure 6.20), changes in dimensionality and strategy (Rasmussen and Eliasmith, 2011), and input variability (see section 3.5), as well as a more general guarantee of robustness of the components to noise (section 2.3). Because the SPA relies on the NEF, which was developed to account for a high degree of heterogeneity and expected noise levels found in cortex, it is not surprising that the SPA is robust. The SPA also takes advantage of the robust dynamics found in neural integrators, and other adaptive dynamical systems. In short, the SPA incorporates robustness at all of its levels of description, from single cells on up.

I earlier suggested that current approaches cover a broad range of adaptation phenomena, but that none covered all of them. Again, I think the SPA is unique in its coverage of a broad range of adaptive behavior. It incorporates biologically

realistic learning of several kinds (e.g., syntactic generalization, STDP), structures for flexible routing of information (e.g. basal ganglia, cortical routing), and a general reliance on methods of adaptive control (e.g., for motor control, and reinforcement learning). Unlike past approaches, the problem of rapid variable creation thus poses little challenge for the SPA (see section 7.3.7).

With respect to the memory criterion, the methods of the SPA have been directly applied to working memory phenomena, resulting in a good model of serial working memory. Nevertheless, I have also provided examples that include various kinds of long-term memory (e.g., the Tower of Hanoi, clean-up, the bandit task). I do not think that the SPA can claim any special status when it comes to accounting for memory phenomena. However, it also does not have any obvious impediment to accounting for particular memory phenomena. As with most other cognitive phenomena, the SPA account uniquely spans a variety of levels of analysis.

I have made much of the potential scalability of the SPA. However, because it is at an early stage of development, it has not been applied to tasks as complex as those considered by ACT-R. Nevertheless, unlike past approaches it provides principles for describing such tasks in a scalable manner. As well, in chapter 7 I have provided a characterization of a single system which addresses a wide variety of simple behaviors within the context of a single model. Consequently, the SPA offers more than just “in principle” scalability. Nevertheless, the SPA, like all other approaches, has much more to do to convincingly address this criterion.

Finally, with respect to the two scientific merit criteria, I think the SPA fares especially well. As I have already detailed above, the ability of the SPA to address a large variety of relevant data is perhaps its most distinguishing feature. As well, it is a very compact theory. The three principles of the NEF, and the four aspects of the SPA are simple to state (although they can take two books to explain). As well, the SPA does not allow arbitrariness in its mappings to time or to neural structures, unlike past approaches. The least constrained aspect of the SPA stems from a lack of knowledge about the interactions between brain structures. The SPA does not yet explicitly fix functional specifications to all parts of the brain. I would argue that this state of affairs reflects our limited knowledge of such functions, but that the SPA should serve to help provide such specifications. Nevertheless, this clearly leaves room for a certain amount of arbitrariness in SPA-based models. As with other approaches, construction of more models that perform a wide variety of tasks without any intervention, will help to limit this arbitrariness. However, this limitation is shared with other approaches, while the non-arbitrariness of temporal constraints and compact specification of the architecture are strengths of the SPA.

In sum, the SPA not only provides unique advantages, such as flexible biological plausibility, it does not suffer any of the most salient drawbacks of past approaches. Of course, much remains to be done to truly demonstrate that the SPA can underwrite large-scale biological cognition. In the next chapter I turn to consideration of some of the central challenges for the SPA, and briefly consider several conceptual consequences of thinking about cognition from the perspective the SPA.

Chapter 10

Consequences and challenges

10.1 Conceptual consequences

If the SPA is as different from past approaches as I have been suggesting, it will be unsurprising that adopting it has a variety of consequences for our understanding of cognitive systems. At several points throughout the book I have hinted at some of the conceptual consequences of the SPA (see, e.g., sections 2.4 and 7.4 on “levels” and a unification of symbols and probabilities, respectively). But, there are several I have not described in detail elsewhere.

For instance, having presented the SPA as both an integration and extension of the three standard approaches and cognitive science, suggests that the distinction between these “paradigms” is not a useful one for the progress of the field. After all, it makes little sense to suggest that symbolism is correct if adoption of that view makes it exceedingly difficult to connect our theory to crucial neuroscientific data that constrains cognitive theories. Similarly, claiming that connectionism is the right approach seems unwarranted if we must reject major components of that view (e.g. concept-level localist representations), and significantly alter other parts (e.g., by introducing dynamic spiking networks). Lastly, we cannot claim that dynamicism has won the day if it does not provide the resources needed to capture high-level cognitive function. Instead it seems more likely that integrating past approaches, as seen in the amalgamation of ACT-R and Leabra (Jilk et al., 2008), will move the field ahead. The SPA is another such integration.

As such, the SPA suggests a variety of ways to re-think central notions in the behavioral sciences. In what follows I consider only four: representation, concepts, reasoning, and dynamics. I am certain that I do not do justice to the

wide variety and subtlety of views on any of these notions. Instead, I take the short discussions I present to be food-for-thought more than carefully constructed arguments. The philosopher in me feels guilty for giving such short shrift to such important issues. Nevertheless, I think it is helpful to consider the variety of ramifications adopting the SPA may have, even if only briefly.

10.1.1 Representation

In some quarters, the notion of representation has been closely identified with classical digital computers (e.g., van Gelder, 1995). However, behavioral scientists typically have a much more liberal notion, freely talking about representations in neural networks, analog computers, and even in the structure of the dynamical system state spaces (Schöner, 2008). In the SPA I have adopted this more liberal view.

This is at least partly because I believe the SPA can provide an account which inter-relates the many kinds of representation identified in behavioral science research. Like the NEF on which it is based, the SPA is quite flexible in what it calls a representation. SPA representations can be scalar values, vectors, functions, vector fields, symbols, images, and so on. In every case, however, mental representations can be characterized as vectors implemented in neurons. The representational commitments of the SPA are thus both flexible and unified.

One unique aspect of the SPA characterization of representation is that the vectors identified as representations are typically *not* vectors of neural activity. In many ways, the representational commitments of the SPA will strike many as being much like those of connectionists. However, the SPA provides two consistent perspectives on the activities of neurons. One relates to directly measurable properties of the system, such as spiking patterns. The other relates to the underlying vector space that is taken to be represented by those measurable properties. This distinction is crucial for bolstering our understanding of neural function because the mapping between the biological wetware and an abstract vector space provides a means of relating brain function to a more general scientific understanding of the world. For example, it allows us to understand certain neurons in area MT as being sensitive to visual velocity. It also enables us to understand neurons in primary motor cortex as representing variables related to the dynamics of movement. Essentially, this mapping gives us a way to connect concise, scientific, and low-dimensional descriptions of such properties to the often very high-dimensional neural activities that we record from brains interacting with those properties.

We may, in a sense, think of representations of vectors in the SPA as being

“doubly distributed”. That is, representations of symbols are not only distributed over the vector space, but that vector space is also distributed over neural activities. I take this as an important strength of the SPA because it allows us to relate high-level characterizations of cognition in terms of symbols to detailed spiking data. As well, it simultaneously allows us to characterize the internal structure of such symbols, providing an increase in the variety and subtlety of manipulations we can define over such representations (see section 7.4).

This way of understanding mental representations has consequences for both neuroscientific and psychological understandings of representation. In neuroscience, taking a single cell to have a preferred direction in a high-dimensional state space can go a long way to helping us understand what is often considered the perplexing variety of neural responses observed with respect input stimuli. For instance, in working memory, allowing for higher dimensional representations provides for much simpler characterizations of the variety of dynamics observed during working memory delay periods (Singh and Eliasmith, 2006; Machens et al., 2010). As well, the seeming complexity of remapping in prefrontal cortex (Hoshi, 2006) can be easily understood in the SPA as routing different sources of information into the same high-dimensional vector space (e.g., visual information in one task, and auditory information in another task). Furthermore, the SPA characterization of representation emphasizes the importance of specifying not only encoding, but also decoding assumptions when making representational claims about particular neural systems. Often, the term “representation” has been misleadingly used in neuroscience to identify only encodings (Eliasmith, 2005a), though more recent work often addresses decoding as well.

In fact, semantic pointers can be seen as a natural generalization of well-known neural representation schemes in neuroscience. Perhaps the most probed representations in the brain are those in primary visual cortex (V1). In V1, individual cells are often characterized as sensitive to specifically oriented bars in particular retinal locations, with their activity falling off as the stimulus either rotates from the preferred orientation, or moves out of the specific retinal location (Kandel et al., 2000, pp. 532-540). It is often fruitful to consider these individual cells as carrying specific content about input stimuli (e.g., oriented bars at a location). In this very particular sense, we can identify each cell with a specific label indicating what it means to the animal. While this could be interpreted as an unusual type of localist representation, a more natural interpretation arises if we consider a large population of such neurons, in which case they can be seen as a distributed representation of an entire visual image.

The SPA uses representations that are localist and distributed in precisely this

same way, at many levels of description. For example, these “localist” single cells in V1 are often taken to represent coefficients of Fourier-like decompositions of images (Olshausen and Field, 1996).¹ This same kind of decomposition forms the basis of SPA visual representation (see section 3.5). More interestingly, the SPA extends this kind of analysis to lexical representation. That is, at higher levels “localist” single cells are taken to represent coefficients of Fourier-like decompositions of concepts. In both cases, the neural tuning curve is taken to be the basis for the decomposition, and the neural activity is the currently relevant coefficient. Furthermore, just as there is significant redundancy in V1 single cell tuning (suggesting that many neurons participate to encode a given coefficient), there is also significant redundancy in the SPA representation of semantic pointers. Applying this representational scheme to symbolic concepts is thus a natural generalization of existing neural evidence.

In a more psychological context, there are at least three major consequences of SPA representations: 1) the internal structure of symbols is important for characterizing a wide variety of cognitive behavior; 2) mental representations are better characterized as “symbol-like” rather than symbols; and 3) mental representations are best thought of as often temporary processing states of activity, rather than as objects that reside in a specific location in the system. The first consequence allows us to identify more sophisticated kinds of transformations of representations (see section 7.4), as well as providing a clear relationship between perceptual and lexical/conceptual representation.

The second consequence arises because of the poor scalability of attempts to do otherwise. Characterizing mental representations as symbol-like carries with it a reminder that many cognitive properties, such as systematicity, compositionality, and productivity, are *idealizations* and hence not practical constraints on cognitive architectures. In contrast, the hard constraints provided by the available physical resources of the brain seem likely only to be respected if the system traffics in symbol-like representations.

The third consequence helps to highlight the importance of vector *processing*, in contrast to the mere “activation” of putative representational states. In essence, the SPA suggests that we think about the brain as more of a processor of neural signals rather than as a storehouse of neural representations. That is, it suggests that we should think of a neural population as a temporary way-station for con-

¹Representations similar in these respects to those found in V1 have been identified in auditory cortex, the hippocampal complex (including place cells and head direction cells), motor areas, and many other parts of the brain.

stantly changing representational content, rather than a home-base for some small subset of re-activating representations. Doing so provides an understanding of how the brain can be a flexible, adaptive, and computationally powerful system. This shift in metaphors also highlights the temporal aspects of neural representation in the SPA. Because the representational contents of a given neural population can change quite dramatically from moment to moment, the dynamics of neural representation must be considered when constructing SPA models. I return to the importance of dynamics shortly.

All three consequences also help identify the fact that most representations in the SPA are a kind of semantic promissory note. Semantic pointers carry with them a means of accessing much more sophisticated semantics than they themselves directly encode. Thinking of central representational structures as not being unified semantic wholes, like symbols, provides a subtler, and more biologically plausible view of how mental representation might be organized. The idea that semantics are encoded through compression and dereferencing provides a natural explanation for the various amounts of time and effort it takes to extract semantics of words from experimental subjects.

Notably, my use of “semantics” here refers largely to the relationship between representational vehicles described by the SPA. The relationship of those vehicles to the external world is also usually considered an important part of “semantics”. While I have included such relationships in several SPA models, arguing that this demonstrates the SPA’s ability to address the symbol grounding problem (see chapter 3), I have not described a general semantic theory in this book. However, I have described a theory of “neurosemantics” in detail in past work (Eliasmith, 2000, 2006), which I take to be fully compatible with the SPA. Encouragingly, this theory has been used by others to account for semantic phenomena as well (Parisien and Thagard, 2008). However, the details of that theory are beyond the scope of this book.

10.1.2 Concepts

Recent theoretical work on concepts seems to have come to a kind of agreement that “in short, concepts are a mess” (Murphy, 2002, p. 492). For some, this is just an observation about the state of the field that means more work needs to be done. For others, such as Machery (2009), this leads to the conclusion that “the notion of concepts ought to be eliminated from the theoretical vocabulary of psychology” (p. 4). Machery’s argument begins with the observation that there are many different things concepts are used to explain (e.g., motor planning, analogy,

categorization, etc.). He then proceeds to suggest that it makes no sense to identify concepts with just one of these functions, as that would be a clearly incomplete account, of concepts. He also argues that it has proven untenable to specify a theory that accounts for all of these different uses of the term. He argues that this state of affairs drives us to accepting his “Heterogeneity Hypothesis,” i.e., that there are many kinds of conceptual phenomena and there is no unification of the kinds – the consequence of accepting this hypothesis is that the notion of “concept” should be eliminated.

A central element of this argument is Machery’s concern that there is no clear unification behind the notion of a “concept”. However, I believe the SPA has the resources to demonstrate how a particular mental representation can be used to explain a wide variety of conceptual phenomena. Semantic pointers, after all, can be constructed out of the binding and collecting of other vectors that participate in different semantic spaces. As I described in section 4.7, semantic pointers can contain content from a variety of perceptual areas, motor areas, and can also encode realistic relations, allowable dynamic transformations, and so on. And, importantly, these sub-elements of a concept, which can themselves contain many sub-elements, are not merely “stuck together” to satisfy our theoretical need for conceptual representations. Instead, these structures can be used to define semantic spaces at a more abstract level than their constituent elements. These more abstract spaces can help the system perform various kinds of rapid, or approximate comparisons between elements within that space, saving valuable processing time when deep semantic analysis is not needed. So, not only do semantic pointers provide a means of combining a wide variety of information into a single mental representation, but doing so has behavioral consequences.

Note that this kind of unification is essentially a functional (not a representational) one. That is, the presence of a semantic pointer encoding a complex structure does not unify our account of concepts on its own. It is only in the context of the SPA, which can decode, manipulate, generate, etc. such representations, that we can provide a unification of conceptual phenomena. Semantic pointers, alone, do not carry enough information to fully capture the semantics of the thing they are about. Nevertheless, a functional unification is one that is sufficient to allay Machery’s concern, and hence avoid the conclusion that “concept” is not a useful concept.

As I mentioned in my earlier discussion of semantics (section 3.3), the SPA can be thought of as a computational account consistent with Barsalou’s work on perceptual symbol systems (Barsalou, 1999). However, the SPA account differs from this and other neo-empiricist views of concepts (e.g., Prinz, 2002). Those

accounts typically argue that concepts do not provide amodal representations as has often been assumed. However, as Machery (2009) points out, the empirical evidence offered in support of this conclusion does not contradict some amodal accounts. I would suggest that semantic pointers can be seen as one such account. That is, semantic pointers may be tightly tied to perceptual information, perhaps even for many different modalities, but they may also include elements which are not modal, such as information about statistical relationships between co-occurrences of words (see section 3.4). In short, semantic pointers offer a way of understanding how concepts can preserve access to deeper perceptual information, while still being able to be manipulated independently of *actually* having access to that information, and independently of that information's affect on at least some aspects of the conceptual content.

I believe this combination of access to amodal and modal conceptual information and functional unification in the SPA allows semantic pointers to account, in principle, for the four major theories of concepts. Let me briefly consider each in turn. The first is the “classical” view, which proposes that having a concept is having a set of necessary and sufficient conditions. While generally out-of-favor, there do seem to be some concepts, such as mathematical ones, that might be best described in this manner. Semantic pointers can capture such concepts by their being defined by purely syntactic relations to other lexical items: e.g., $\text{triangle} = \text{closedShape} \otimes \text{property1} + \text{threeSides} \otimes \text{property2} + \text{straightEdges} \otimes \text{property3}$.

The second major theory is prototype theory. The SPA account of prototypes has been discussed in the perceptual case in section 3.5, where the prototype is the mean of the clustering of the set of examples. In Spaun, this mean was used as the perceptual basis for higher-level concepts. In the more conceptual case, prototypes are naturally accounted for as slot-filler structures, as discussed in section 4.7.

Third, exemplar theories can be accounted for by having specific perceptual vectors bound into the concept of interest at a higher level (e.g., a specific slot), or low-level, specific examples of a category can be generated by using the statistical model in early perceptual areas in a top-down manner to generate a sample (as shown in figure 3.7).

Finally, the theory theory of concepts can be accounted for by the fact that semantic pointers can consist of such a wide variety of representations, including allowable transformations, and relations to other concepts, as demanded by the theory. This can include information about what features of the concept are essential, which are perceptual, how those can change under various transformations, and so on.

Of course, these considerations are so brief that they can at best be considered

suggestive. Nevertheless, they are at least suggestive: that is even these brief considerations make it plausible that the SPA can provide the resources needed to unify our understanding of concepts. Again, I will emphasize that the contents and behavioral efficacy of any semantic pointer depend on its being situated in the architecture. So, while we can identify a single vehicle with the concept for expediency, the contents of that concept is determined by semantics that are only fully defined by accessing representations spread throughout the system.

The SPA suggests, then, that we can come to better understanding of concepts by providing a computational definition that allows us to examine our simulations in a way analogous to how we examine human subjects. Perhaps we can begin to tackle the disarray in conceptual theorizing by adopting a unification through a single functional architecture. Such an architecture will need to be shown to give rise to the wide variety of complex results generated by psychologists in relation to concepts. The examples I provided in this book go only a small way to realizing this goal. But, I believe the SPA may provide some basic resources for tackling the problem of systematizing our understanding of concepts.

10.1.3 Inference

Like “concept”, “inference” suffers from a very broad application in the behavioral sciences. For instance, the term is used to refer to automatic, “sub-personal” processes that lead us to have representations of the world that go beyond the information available to our senses. For example, when I assume that objects have a back side even though I cannot see it, I am often said to be relying on mechanisms of perceptual inference. In other instances, the term is used to characterize effortful reasoning that helps us determine regularities in our environment. In some such cases, we do not explicitly know the rules we are using to reason in this way. In other cases, we may know the rule explicitly, such as when we are performing logical inference.

To determine if an inference has been made, experimenters typically observe the choice behavior of a subject. More often than not, such behaviors can be naturally characterized as a kind of decision-making. As a result, decision-making and inference often go hand-in-hand in the behavioral sciences. This is unsurprising, since we might expect inference to result in a representation that can support the choice of actions. Given the huge variety of decision-making and inference phenomena, it is tempting to make an argument analogous to that which Machery made about concepts: that the diffuseness of the phenomena suggests we can neither identify a specific phenomenon as being diagnostic of inference nor can we

hope to find a unified theory of inference.

Perhaps not surprisingly, my suggestion is that the SPA can play a role in helping us to understand inference and decision-making by characterizing both in the context of a complete architecture. In other words, I believe that the SPA can help us define single unified models, like Spaun, which integrate a variety of observable brain processes in a consistent and coherent way. This, then, allows us to taxonomize the variety of inference phenomena we observe by relating them directly to mechanisms explicated by our model. Thus, the unification of what we label “inference” comes from its many varieties being explained by a unified underlying architecture.

One reason we might think that such a unification can be provided by the SPA, is that the SPA is in the unique position of being able to integrate the more symbolic kinds of inference with the more statistical kinds of inference, and the more linguistic kinds of inference with the more perceptual kinds of inference. In section 7.4, I argued that the consistent matrix-vector and statistical interpretations that we can apply to the cortical-basal ganglia-thalamus circuit is unique to the SPA. This was partly because the representations in the SPA are symbol-like while retaining significant internal structure.

Perhaps surprisingly, this duality can go a long way towards capturing many of the kinds of inference that we would like to explain. For example, in describing how a semantic pointer comes about through compression (section 3.5), I was also describing how sub-personal perceptual inference can be captured by the SPA. Similarly, when I was discussing the bandit task (section 6.5), I was also discussing how sub-personal states help to infer appropriate actions given changing environmental regularities. When I was describing how we can explain induction in the Raven’s matrix task (section 4.6), I was also describing how effortful inference can be used to identify abstract regularities. And finally, language-based logic-like inference and planning can also be captured in the architecture, as was demonstrated by the Tower of Hanoi (section 5.8) and Wason card task (section 6.6) examples.

As an aside, we can think of the Raven’s model as providing a mechanistic solution to the classical philosophical inference problem called the “problem of induction”. This is the problem of understanding why people infer rules that go beyond the available evidence (since presumably we have only counted to a finitely large number, yet we infer that the rule that generates the next number is to add one). The Raven’s matrix model suggests that such an inference mechanism is a natural one to build into a biological, cognitive system.

In sum, I will note that, as in the case of concepts, a solid unification of “in-

ference” has a long way to go. While a model like Spaun goes some way to integrating this variety of inferencing processes within the context of a single model, it clearly does not fully capture the complexity of even these examples. Nevertheless, it does suggest that we ought to be able understand the wide variety of inference phenomena that we encounter in the behavioral sciences by employing unified architectures like the SPA.

10.1.4 Dynamics

In past work, I have been quite critical of dynamicism as a cognitive approach (Eliasmith, 1996, 1997, 1998, 2009b). However, I believe that the SPA provides a way of addressing those criticisms, while still embracing the compelling insights of the view. For instance, dynamicists generally do not map the parameters in their proposed models (e.g. “motivation”) to the underlying physical system that is being modeled (i.e., neurons, networks, brain areas, etc.) in any detail (see, e.g., Busemeyer and Townsend, 1993). Clearly, the SPA provides a mapping down to the level of individual neurons, allowing us to understand the mechanisms which give rise to the dynamics that we observe in brains and in behavior. As well, I and others (e.g., Bechtel, 1998) have criticized the anti-representationalism that is often espoused by dynamicists, something clearly not embraced by the SPA.

More generally, I have had serious concerns regarding the approach to dynamics taken by all three standard approaches. As I discussed to some extent in section 9.2, none of the three approaches provides principles for mapping cognitive models onto independently measurable, low-level system dynamics, such as membrane or synaptic time constants, refractory periods, and so on. In short, unlike past approaches, the SPA is concerned with real-time dynamics “all the way through”. It is concerned with the dynamics inherent in the physical body which must be controlled by the nervous system (section 3.6). It is concerned with the dynamics of recurrently connected spiking attractor networks (section 6.2). And, it is concerned with the dynamics of complex cognitive processes (section 5.8). Above all, the SPA is concerned with how all of the various temporal measurements we make of the system, at many levels of analysis, interact to give rise to the variety of observed dynamics.

One main consequence of this focus on real-time and real physics, is that the SPA is in a position to embrace the constructive insights of researchers who emphasize the embodiment and embeddedness of cognitive systems in their environment. I have argued elsewhere that a problematic tendency of these researchers is to blur the lines between agents and environments (Eliasmith, 2009a). Indeed it

has been claimed that “nothing [other than the presence of skin] seems different” (Clark and Chalmers, 2002, p. 644) between brain-brain and brain-world interactions. The suggested conclusion is that our characterization of cognition cannot stop at the boundaries of an agent. Ironically, I believe that the very dynamical nature of these systems, which is often appealed to in drawing such conclusions, suggests exactly why there *is* a difference between the inside and outside of the nervous system. In short, the degree and speed of coupling inside the nervous system is generally much greater than that between the nervous system and the body, or the body and the world. One straightforward reason for this is that the body has mass. Hence, the dynamics tend to be considerably different than those of the transmission of electrical signals, which have essentially no mass.

This suggestion, that we can rely on differences in dynamics to identify useful system boundaries for scientific exploration, has a further consequence for a long-held position in philosophy called “functionalism”. Functionalism is the view that what makes a system the kind of system it is, for example a mind, is determined by the functions that it computes. On a standard understanding of what counts as computation (e.g., as characterized by Turing machines) functionalism suggests that two systems operating at very different speeds, that is with very different dynamics, could both be computing the same function (Turing machines ignore the amount of time taken to compute a function). As a consequence, philosophers have sometimes argued that an interstellar gas cloud, or a disembodied spirit, might have a mind because it might be computing the same function as, or functionally isomorphic to, humans (Putnam, 1975). However, such conclusions clearly do not hold if we think that the length of time it takes to compute a function matters for whether or not that function is being realized. Mathematically, all that I am suggesting is that functions describing cognition be written as not only functions of state, but also functions of time. As I have suggested elsewhere, such a “temporal functionalism” provides a much more fine-grained, and I believe plausible, basis for a taxonomy of cognitive systems (Eliasmith, 2003).

The importance of time for understanding what cognitive functions a biological system can actually compute is illustrated by the discussion of control in chapter 5. Indeed, the obvious practical importance of control and routing make it hard to dissociate the functions a system can perform from its dynamics: consider the consequences of replacing digital telephone switches with human operators. One interesting consequence of the SPA focus on control, is that it helps to expand our concepts of adaptation and of learning in cognitive science. As mentioned in section 6.4, learning is most often associated with changing weights, or constructing new, permanent representations in long-term memory. However, as was made

evident by the Raven's matrix task, the ability of past input to influence future response can often be a consequence of controlled recursive dynamics. In some cases, as emphasized by Bob Hadley's rapid variable creation task (section 7.3.7), such dynamics seem the only possible explanation for the observed cognitive phenomena.

In sum, the SPA has conceptual consequences for embodiment, system boundaries, functionalism, and learning. However, the SPA also has practical consequences for contemporary discussions of dynamics in experimental neuroscience. In recent years, there has been an increasing emphasis on the kinds of frequencies and temporal correlations observed during tasks in a wide variety of brain structures (Fries, 2009). Consideration of such phenomena has not played a role in the development of the SPA. Nevertheless, it is an important empirical constraint on SPA models.

For example, Pesaran et al. (2002) performed working memory experiments while recording from monkey cortex, and performed a spectrographic analysis. This analysis has been taken to show that during working memory, there is a shift in frequencies, increasing power in the gamma band (i.e. 25-90 Hz) during the delay period. As shown in figure 10.1, the single cell and population frequency spectrograms are for the model and the data are similar. Some have suggested that such results indicate that gamma band synchronization is a fundamental mechanism of neural computation (Fries, 2009). However, I would be hesitant to call them "fundamental", as there was no need to consider them in order to construct models which give rise to the same patterns. Others have suggested that such patterns are merely epiphenomenal (Tovée and Rolls, 1992). Such phenomena are unlikely to be strictly epiphenomenal, since the presence or absence of such synchronization will have consequences for precise currents in the dendrites in neurons receiving such signals. I am inclined to think that such phenomena are neither fundamental nor epiphenomenal, but instead they are the consequences of the implementational constraints imposed by using neurons for implementing cognitive systems. Not surprisingly, those implementational constraints will have some functional implications themselves, though they do not seem to be so severe as to dominate the information processing occurring within the brain.

10.2 Challenges for the SPA

While I am optimistic about the contributions the SPA may make to our understanding of biological cognition, it would be rash to suppose that no challenges

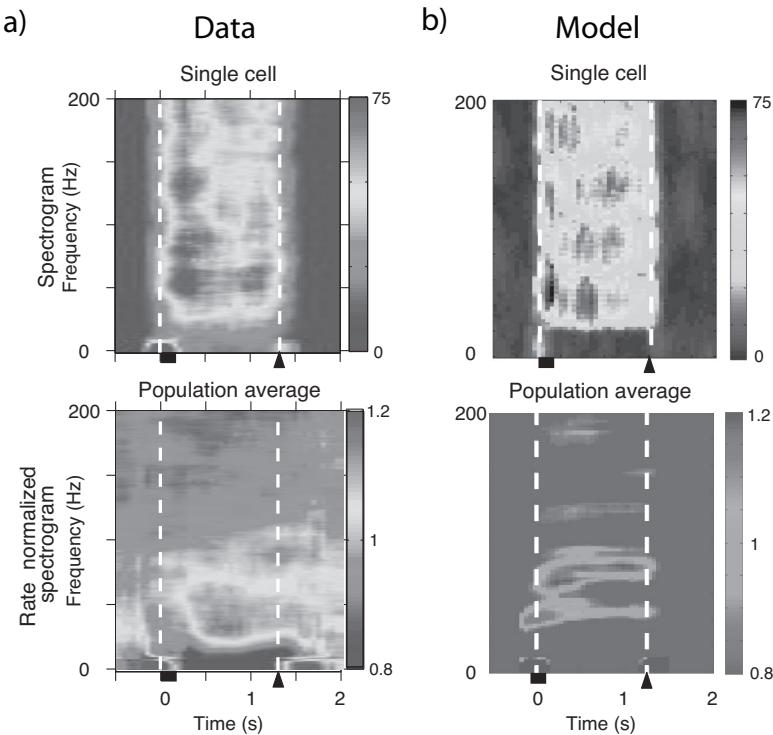


Figure 10.1: Frequency profiles during working memory. a) Spike spectrograms for a single neuron (top) and the population average (bottom) during a simple working memory task. b) The same analysis performed on an SPA working memory component. Both the model and the data show a shift from minimal frequency response before the stimulus to stimulus-driven low frequency information (at the first white line) to significant higher frequency content during the delay period, with a final reduction after response (at the second white line). Both population averages show frequency content mainly in the gamma range. (Data adapted from Pesaran et al. (2002) with permission)

remain for the development of the SPA. I have mentioned various issues throughout the book, but attempt to gather and expand on some of them in the following three sections.

10.2.1 Representation

The precise nature of time-dependent representation in a biological system depends on the physical properties of the neurons that underwrite that representation. Consequently, low-level biological properties have representational consequences in the SPA. So, any simplifications that are made in simulations introduce the possibility of mischaracterizing representation. While the NEF goes a long way to incorporating a wide variety of low-level biophysical properties of cells, there are some assumptions in the software implementation in Nengo which may need to be revisited as SPA models grow more sophisticated. For example, we typically assume that all synapses in a particular projection between populations have the same dynamics. However, we know that there is a distribution of time constants, and the processes that determine the precise synaptic dynamics of each cell are somewhat variable.

In general, we have so far assumed that all neurons are point neurons. That is, they have no spatial extent. Consequently, we do not currently account for the location of synapses on a receiving cell's dendritic tree or soma. In addition, Nengo neuron models do not capture some more detailed phenomena such as dendritic spiking, back propagation of somatic potentials up dendritic trees, or other space dependent dynamics. To summarize, many of our models could stand to have more synaptic heterogeneity and could better account for spatial effects. Both of these improvements, however, come at high computational costs.

In many ways, the SPA is more focused on representation in populations of neurons than in single neurons. At this level of analysis, there are many challenges specific to functional properties of a given part of the brain. For example, there is always the challenge of determining the appropriate dimensionality of the space to be represented by a population of cells. Relatedly, determining how much and what kinds of compression need to occur through sequential neural processing of semantic pointers may vary from area to area, and hence needs to be considered carefully for each new model. In general, these observations simply boil down to the fact that many specific representational properties are likely to depend on the particular functions we are modelling in a given SPA model. Perhaps the most challenging area where such questions must be answered is in the domain of language.

As I mentioned when introducing the notion of binding, understanding linguistic processing requires determining the appropriate way to encode linguistic structure. I chose a particular, simple, method that binds the value of variables to be role that the variable plays in the structure. However, there are many questions that arise when attempting to determine how language-like structures should be encoded. For example, should adjectives be bound to nouns, or should they be bound to their own role and added to nouns? Do agent and theme roles need to be bound to the verbs they are used to modify in order to appropriately preserve semantics? And so on. It remains an enormous challenge to determine which alternative is best for capturing observed cognitive behavior.

Furthermore, even the SPA assumptions regarding the nature of an appropriate binding operation can easily be challenged. The choice of circular convolution for binding is difficult to directly motivate from data. It could well be that one of the other VSAs is ultimately more appropriate for modeling cognitive behavior. Perhaps attempting to implement a sophisticated linguistic approach like Smolensky's Harmony Theory in the SPA will lead to more stringent constraints on which kind of representations are adequate for capturing linguistic behavior.

A more practical challenge for relating SPA representations to neuroscientific data, lies in improving the mappings between low-level neural data and a variety of measured neural signals. For example, local field potentials (LFPs) are often recorded from populations of neurons and compared to individual spikes recorded from the same populations. Similarly, event related potentials (ERPs) and electroencephalogram's (EEGs), while more removed from individual spiking neurons, should be able to be inferred based on such a model. These and other neural signals provide important empirical constraints for SPA models. Presumably, such challenges can be met in the same manner as we have done with fMRI (section 5.8), but this remains to be seen.

10.2.2 Architecture

When pressed, I would be willing to concede that the SPA is less of an architecture, and more of an architecture sketch combined with a protocol. That is, the SPA, as it stands, is not overly committed to a specific arrangement and attribution of functions to parts of the brain. As a result, a major challenge for the SPA is to fill in the details of this schema. I believe a natural approach for doing so is to adopt Michael Anderson's "massive redeployment hypothesis" (Anderson, 2010). This is the idea, consistent with the SPA, that neural areas compute generically specifiable functions that are used in a wide variety of seemingly disparate tasks.

The SPA provides a means of understanding how the same neural resources can be driven by very different inputs, i.e., by redeploying the resources of cortex using routing strategies. Nevertheless, which areas of cortex perform which functions is highly underspecified by the SPA as it stands.

One of the functions that the SPA is most explicit about, is that of the basal ganglia. As initially described, the SPA employs the basal ganglia to perform action selection. However, there is good evidence that if the output of the basal ganglia is removed, actions can still be selected, though less smoothly and quickly. This is consistent with recent hypotheses, supported by anatomical and physiological experiments, which suggests that the basal ganglia is important for novel action sequences and teaching the cortex, and not for well-learned actions (Turner and Desmurget, 2010). However, the statistical reinterpretation that I provided in section 7.4, very naturally fits with such a view. After all, in that interpretation it is clear that cortical processing must precede basal ganglia processing, and can result in action selection even if basal ganglia does not play its typical role. Specifically, this is because the role of the basal ganglia under that interpretation is more clearly one of refinement, than one on which all subsequent cortical processing depends. Indeed, this is also true of the matrix-vector interpretation, but is less explicit. In both cases, we can think of the architecture as having a kind of “default routing” which will allow somewhat inflexible processing to result in reasonably appropriate actions for given inputs. As a result, however, it becomes clear that a major challenge for the SPA is to provide detailed hypotheses regarding such default cortical interactions.

Thinking of the basal ganglia as more of a teacher than a “selector” also raises aspects of learning that are not yet well integrated into the SPA. For example, connections between thalamus and cortex are likely as sensitive to environmental contingencies as those from cortex to striatum, but they are not accounted for by learning in the SPA. Similarly, if we think basal ganglia as performing a corrective role in tuning cortico-cortical connections, we should include such modulation effects in projections to cortex.

Even many of those elements of cortical processing that we have included in our architecture need to be significantly extended. For example, visual routing and object recognition should be implemented in the same network, although I have considered them separately in the models presented in sections 5.5 and 3.5. As well, several challenges remain regarding clean-up memories (see section 4.5). Not only do we need to better understand which anatomical areas act as clean-up memories in cortex, we also need methods for rapidly constructing such memories, extending such memories as new information becomes available, and so on.

Conspicuously, a model of hippocampus is absent from the SPA, but it may be able to account for some of these aspects of clean-up memory. This, of course, remains to be seen.

Indeed, there is a long list of brain functions that have not even been mentioned in my discussion of the SPA. For example, emotion and stimulus valuation is largely absent. These play a crucial role in guiding behavior, and need to be integrated into the SPA, both introducing new areas (e.g. orbital frontal cortex, cingulate, etc.) and interacting with current SPA elements, such as the basal ganglia. As well, perceptual processing in visual, olfactory, tactile, and other modalities has remained largely unaddressed. Similarly, the role of the cerebellum, and many parts of brainstem have not been considered in any kind of detail. The same can be said for the microstructure of thalamus, as well as the many other nuclei that are not yet mentioned in the SPA. In short, the SPA as it stands provides a somewhat minimal coverage of anatomical structures. My defense should not be a surprising one: the SPA is a new approach.

In closing I will mention one final challenge for the SPA: brain development. The methods for brain building that I have described here sidestep important and interesting questions about how brains change over the lifetime of their owners. While I have discussed various aspects of adaptation and learning, I do not pretend to be providing an account that would allow us to understand how brains evolve from infancy to adulthood. Perhaps the SPA can be used to guide such a story, but it clearly remains an immense challenge to do so.

10.2.3 Scaling

Many of the architectural challenges in the previous section could be equally well considered scaling challenges because many are demands for more elements in the architecture. Undoubtedly, as the number of elements increases, scaling challenges related to integration can become critical. However, there is another kind of scaling problem that we face even with a fixed number of anatomical elements. Consider, for example, the Spaun model. Despite the fact that it has several anatomical areas integrated in the model, a large number of them are quite simple: reinforcement learning is applied to only a few simple actions; motor control is applied to a single arm; perceptual input is in a fixed location; and semantics is limited to the domain of numbers. These issues of scaling arise without consideration of adding elements to the model: that is my focus here.

To begin, reinforcement learning (RL) should be demonstrated in more complex tasks like the Tower of Hanoi, where the representations and set of pos-

sible actions are of much greater complexity. One of the major challenges for RL is correctly representing state and action spaces. Consequently it is crucial to demonstrate that the representational assumptions of the SPA are consistent with RL algorithms that may be implemented in basal ganglia-cortex interactions. Even more convincing would be the generation of the representations from sensory inputs. This would necessitate scaling the kinds of adaptation and learning employed in the SPA independently of RL, as well.

Such an extension would have implications for semantics in the SPA. Currently, the semantics of representations in most of the models presented are quite limited. Scaling these to include additional modalities, dynamic transformations, and more sophisticated lexical relationships remains a challenge. One potentially fruitful strategy for addressing this problem is to attempt to build on successful past approaches (e.g., Rogers and McClelland, 2004) that are consistent with SPA commitments. This would also allow models to explicitly scale to adult-sized vocabularies, as I have only made in principle arguments that this will work. Such models would also allow the SPA to begin to address challenges related to scaling the sophistication of linguist behaviors, as discussed in the section representational challenges.

A quite different scaling challenge is increasing the degrees of freedom in the motor system. Not only do current SPA implementations only account for the arm, and not the rest of the body, the models have fewer degrees of freedom than the human arm, and are constrained to move in a plane. One excellent test for scaling the SPA motor system would be to embed the SPA into an actual physical body. This would help explicitly realize the ability of the SPA to account for both more immediate environmental interactions, as well as aspects of cognitive performance. However, the computational demands of such real-time interaction are severe.

In fact, such computational demands are a general practical consequence of attempting to address the scaling challenges. To address these to some degree, we have written code for Nengo to run on GPUs, taking advantage of their massive parallel computational resources, which significantly speeds up Nengo models. However, ultimately, the low-power high-efficiency computing necessary to run large simulations in real time will most likely be realized through hardware implementation. For this reason it is an important, practical scaling challenge to implement SPA models on hardware architectures, such as Neurogrid (Silver et al., 2007) or SpiNNaker (Khan et al., 2008). Our lab is currently working in collaboration with Kwabena Boahen's group at Stanford on Neurogrid chips, which can currently simulate up to a million neurons in real time. However, we are only

beginning to address the many challenges that remain for implementing arbitrary SPA models in neuromorphic hardware.

10.3 Conclusion

Perhaps the greatest challenge for the SPA is for it to become any kind of serious contender in the behavioral sciences. The field is a vibrant and crowded one. Consequently, I will be happy if reading this book convinces even a handful of researchers that we ought to relate all of our cognitive theories to the biological substrate. I will be *very* happy if this book plays a role in increasing the number of cognitive models that are specified at the neural level. I will be ecstatic if the SPA itself is deemed useful in constructing such models. And, I will be shocked if major elements of the SPA turn out to be right. While I have attempted to show how the SPA can address a wide variety of the QCC, often better than its competitors, the many challenges that remain for the SPA make it unclear if anything recognizable as the SPA itself will remain as such challenges are addressed.

One thing that has become clear to me, is that addressing such challenges for the SPA, or any other architecture, is going to be the work of a community of researchers. This is not the observation that the behavioral sciences are interdisciplinary. Instead, it is the observation that to make real progress on advancing our understanding of biological cognition, groups with diverse expertise will have to work *together*, not just alongside one another. I suspect that the only practical way to do this is to have large-scale integrated theoretical constructs, likely in the form of a computational model or a family of such models, that all of these experts can test and extend. As a result, there needs to be a degree of agreement on modeling practices, software, databases, and so on. Consequently, we continue to spend significant effort developing not only an architecture, but tools that go beyond the architecture, for helping to distribute, test, and implement biologically realistic large-scale models.

Despite the enormous challenges that remain, I am optimistic. If the field more aggressively pursues the *integration* of results across disciplines by developing coherent methods and tools, then it will be progressively able to satisfy the QCC. In short, I think that unraveling the mysteries of biological cognition is only a matter of time.

Appendix A

Mathematical derivations for the NEF

A.1 Representation

A.1.1 Encoding

Consider a population of neurons whose activities $a_i(\mathbf{x})$ encode some vector, \mathbf{x} . These activities can be written

$$a_i(\mathbf{x}) = G_i [J_i(\mathbf{x})], \quad (\text{A.1})$$

where G_i is the nonlinear function describing the neuron's response function, and $J_i(\mathbf{x})$ is the current entering the soma. The somatic current is defined by

$$J_i(\mathbf{x}) = \alpha_i \langle \mathbf{x} \cdot \mathbf{e}_i \rangle + J_i^{bias} \quad (\text{A.2})$$

where $J_i(\mathbf{x})$ is the current in the soma, α_i is a gain and conversion factor, \mathbf{x} is the vector variable to be encoded, \mathbf{e}_i is the encoding vector which picks out the 'preferred stimulus' of the neuron (see section 2.5), and J_i^{bias} is a bias current that accounts for background activity.

The nonlinearity G_i which describes the neuron's activity as a result of this current is determined by physiological properties of the neuron(s) being modeled. The most common model used in the book is the leaky integrate-and-fire (LIF) neuron. For descriptions of many other neural models, some of which are in Nengo, see Bower and Beeman (1998); Koch (1999); Carnevale and Hines (2006).

The sub-threshold evolution of the LIF neuron voltage is described by

$$\dot{V}(t) = -\frac{1}{\tau_{RC}}(V(t) - J(\mathbf{x})R) \quad (\text{A.3})$$

where V is the voltage across the membrane, $J(\mathbf{x})$ is the input current, R is the passive membrane resistance, and τ_{RC} is the membrane time constant. When the membrane voltage crosses a threshold V_{thresh} , a spike is emitted, and the cell is reset to its resting state for a time period equal to the absolute refractory time constant τ_{ref} .

The output activity of the cell is thus represented as a train of delta functions, placed at the times of spikes t_m as $a_i(\mathbf{x}) = \sum_m \delta(t - t_m)$. The overall encoding equation is thus

$$\delta(t - t_m) = G_i \left[\alpha_i \langle \mathbf{x} \cdot \mathbf{e}_i \rangle + J_i^{bias} \right]. \quad (\text{A.4})$$

where $\delta(t)$ represents a neural spike, m indexes the spikes, i indexes the neurons, $G[]$ is the spiking neural model, α_i is the gain of the neuron, \mathbf{x} is the variable being encoded, \mathbf{e} is the encoding (or preferred direction) vector, and J^{bias} is the background current.

A.1.2 Decoding

To define the decoding, we need to determine the postsynaptic current (PSC) and the optimal decoding weight. A simple model of the PSC is

$$h(t) = e^{-t/\tau_{PSC}}. \quad (\text{A.5})$$

where τ_{PSC} is the time constant of decay of the PSC. This varies with the type of neurotransmitter being used. Typical values are 5ms for AMPA, 10ms for GABA, and 100ms for NMDA receptors.¹

Given an input spike train $\delta(t - t_m)$ generated from the encoding above, the ‘filtering’ of the neural spikes by the PSC gives an ‘activity’ of²

$$a_i(\mathbf{x}) = \sum_m^M h(t - t_m)$$

¹A wide variety of time constants, with supporting empirical evidence can be found at <http://compneuro.uwaterloo.ca/cnrglab/?q=node/537>.

²It is important to keep in mind that the PSC filtering, and the weighting by a synaptic weight happen at the same time, not in sequence as I am describing here. In addition, the decoders here constitute only part of a neural connection weight.

To determine the optimal linear decoding weight to decode this activity, we need to minimize the error between the representation of the signal under noise, and the original signal:

$$E = \int_R \left[\mathbf{x} - \sum_i^N (a_i(\mathbf{x}) + \eta_i) \mathbf{d}_i \right]^2 d\mathbf{x} d\eta \quad (\text{A.6})$$

where N is the number of neurons, η is a random perturbation chosen from a Gaussian distribution with mean zero, \mathbf{d}_i are the decoders to be determined, and R is the range over which the representation is to be optimized (e.g., a unit hypersphere). In matrix notation, the solution to this common kind of optimization is:

$$\mathbf{D} = \Gamma^{-1} \Upsilon$$

where \mathbf{D} is the $N \times D$ matrix of optimal decoders (one decoder \mathbf{d}_i in each row of the matrix), Υ is the matrix where each row $\Upsilon_i = \int_R \mathbf{x} a_i(\mathbf{x}) d\mathbf{x}$, and Γ is the correlation matrix of neural activities where each element $\Gamma_{ij} = \int_R a_i(\mathbf{x}) a_j(\mathbf{x}) d\mathbf{x}$ and $\Gamma_{ii} = \Gamma_{ii} + \sigma^2$. The σ^2 is the variance of the noise from which the η_i are picked.

The overall decoding equation is thus

$$\hat{\mathbf{x}} = \sum_{i,m}^{N,M} h_i(t - t_m) \mathbf{d}_i. \quad (\text{A.7})$$

where N is the number of neurons, M is the number of spikes, i indexes the neurons, m indexes the spikes, $h(t)$ is the PSC of the neuron, $\hat{\mathbf{x}}$ is the estimate of the variable being represented, and \mathbf{d}_i is the decoder for neuron i to estimate \mathbf{x} .

A.2 Transformation

To define a transformational decoder, we follow the same procedure as in section A.1.2, but minimize a slightly different error to determine the population decoders, namely:

$$E = \int_R \left[f(\mathbf{x}) - \sum_i^N (a_i(\mathbf{x}) + \eta_i) \mathbf{d}_i \right]^2 d\mathbf{x} d\eta$$

where we have simply substituted $f(\mathbf{x})$ for \mathbf{x} in equation A.6. This results in the related solution:

$$\mathbf{D}^f = \Gamma^{-1} \Upsilon^f$$

where $\Upsilon_i^f = \int_R f(\mathbf{x})a(\mathbf{x})d\mathbf{x}$, resulting in the matrix \mathbf{D}^f where the transformation decoders \mathbf{d}_i^f can be used to give an estimate of the original transformation $f(\mathbf{x})$ as:

$$\hat{f}(\mathbf{x}) = \sum_i^N a(\mathbf{x})\mathbf{d}_i^f.$$

Thus, over time we have the overall decoding equation for transformations as:

$$\hat{f}(\mathbf{x}) = \sum_{i,m}^{N,M} h_i(t - t_m)\mathbf{d}_i^f \quad (\text{A.8})$$

where N is the number of neurons, M is the number of spikes, i indexes the neurons, m indexes the spikes, $h(t)$ is the PSC of the neuron, $\hat{f}(\mathbf{x})$ is the estimate of the transformation being performed, \mathbf{x} is the representation being transformed, and \mathbf{d}_i^f is the decoder for neuron i to compute f .

A.3 Dynamics

The equation describing figure 2.11a is:

$$\dot{\mathbf{x}}(t) = \mathbf{Ax}(t) + \mathbf{Bu}(t). \quad (\text{A.9})$$

Notably, the input matrix \mathbf{B} and the dynamics matrix \mathbf{A} completely describe the dynamics of an linear time-invariant (LTI) system, given the state variables $\mathbf{x}(t)$ and the input $\mathbf{u}(t)$. Taking the Laplace transform of (A.9) gives:

$$\mathbf{x}(s) = h(s) [\mathbf{Ax}(s) + \mathbf{Bu}(s)],$$

where $h(s) = \frac{1}{s}$.

In the case of the neural system, the transfer function $h(s)$ is not $\frac{1}{s}$, but is determined by the intrinsic properties of the component cells. Because it is reasonable to assume that the dynamics of the synaptic PSC dominate the dynamics of the cellular response as a whole (Eliasmith and Anderson, 2003), it is reasonable to characterize the dynamics of neural populations based on their synaptic dynamics, i.e. using $h(t)$ from equation (A.5). The Laplace transform of this filter is:

$$h'(s) = \frac{1}{1 + s\tau}.$$

Given the change in filters from $h(s)$ to $h'(s)$, we need to determine how to change \mathbf{A} and \mathbf{B} in order to preserve the dynamics defined in the original system (i.e. the one using $h(s)$). In other words, letting the neural dynamics be defined by \mathbf{A}' and \mathbf{B}' , we need to determine the relation between matrices \mathbf{A} and \mathbf{A}' and matrices \mathbf{B} and \mathbf{B}' given the differences between $h(s)$ and $h'(s)$. To do so, we can solve for $s\mathbf{x}(s)$ in both cases and equate the resulting expressions:

$$\begin{aligned}\mathbf{x}(s) &= \frac{1}{s} [\mathbf{Ax}(s) + \mathbf{Bu}(s)] \\ s\mathbf{x}(s) &= [\mathbf{Ax}(s) + \mathbf{Bu}(s)]\end{aligned}$$

and

$$\begin{aligned}\mathbf{x}(s) &= \frac{1}{1+s\tau} [\mathbf{A}'\mathbf{x}(s) + \mathbf{B}'\mathbf{u}(s)] \\ s\mathbf{x}(s) &= \frac{1}{\tau} (\mathbf{A}'\mathbf{x}(s) - \mathbf{x}(s) + \mathbf{Bu}(s))\end{aligned}$$

so

$$\frac{1}{\tau} (\mathbf{A}'\mathbf{x}(s) - \mathbf{x}(s) + \mathbf{Bu}(s)) = [\mathbf{Ax}(s) + \mathbf{Bu}(s)].$$

Rearranging and solving gives:

$$\mathbf{A}' = \tau\mathbf{A} + \mathbf{I} \tag{A.10}$$

$$\mathbf{B}' = \tau\mathbf{B} \tag{A.11}$$

where τ is the synaptic time constant of the neurons representing \mathbf{x} .

Notably, this procedure assumes nothing about \mathbf{A} or \mathbf{B} . However, it does assume that the system is LTI. For many nonlinear and time-varying systems, however, similar derivations follow. For the time-varying case, it is possible to simply replace \mathbf{A} with $\mathbf{A}(t)$ in the preceding derivation for LTI systems. For any nonlinear system that can be written in the form

$$\mathbf{x}(s) = \frac{1}{s} F(\mathbf{x}(s), \mathbf{u}(s), s),$$

a similar derivation leads to

$$F'(\mathbf{x}(s), \mathbf{u}(s), s) = \tau F(\mathbf{x}(s), \mathbf{u}(s), s) + \mathbf{x}(s).$$

In short, the main difference between synaptic dynamics and integration is that there is an exponential ‘forgetting’ of the current state with synaptic dynamics.

As long as this difference is accounted for by τ (which determines the speed of forgetting) and $\mathbf{x}(s)$ (which is a ‘reminder’ of the current state), any dynamical system can be implemented in a recurrent spiking network with PSC dynamics given by equation A.5. Importantly, however, the implementation of a dynamical system may not be successful if the representation of the state space by the network is not sufficiently accurate.

Appendix B

Mathematical derivations for the SPA

B.1 Binding and unbinding HRRs

The SPA uses circular convolution for binding and unbinding, as first proposed by Plate (1991). Circular convolution is a variant of the more common convolution operation. Convolution is widely used in linear system analysis and signal processing, and has many properties in common with scalar and matrix multiplication (e.g., it is commutative, associative, distributive over addition, a zero vector and an identity vector both exist, and most vectors have exact inverses). It is useful in these domains because it can be used to compute the output of a linear system given any input signal. Given the impulse response filter $y(t)$ that defines a linear system, and an input signal $x(t)$, the convolution is

$$z(t) = x(t) * y(t) = \int_{-\infty}^{\infty} x(\tau)y(t - \tau) d\tau.$$

It is standard to visualize convolution as flipping the filter around the ordinate (y-axis), and sliding it over the signal, computing the integral of the product of the overlap of the two functions at each position, which gives $z(t)$. In the case where $y(t)$ and $x(t)$ are discrete vectors, the length of the result is equal to the sum of the lengths of the original vectors, minus one. So, if the vectors are the same size, the result is approximately double the original length. This could lead to scaling problems in the case of using the operation for binding.

Circular convolution is the same operation, but the as the elements of $y(t)$ slide off of the end of $x(t)$, they are wrapped back onto the beginning of $x(t)$.

Consequently, we can define the discrete circular convolution of two vectors as

$$\begin{aligned}\mathbf{z} &= \mathbf{x} \circledast \mathbf{y} \\ z_j &= \sum_{k=0}^{D-1} x_k y_{j-k}\end{aligned}$$

where subscripts are modulo D (the length of the filter vector). The result of this form of convolution for two equal length vectors is equal to their original length because of the wrapping.

A computationally efficient algorithm for computing the circular convolution of two vectors takes advantage of the Discrete Fourier Transform (DFT) and Inverse Discrete Fourier Transform (IDFT). In general, Fourier Transforms are closely related to convolution because the Fourier Transform of any convolution is a multiplication in the complimentary domain. The same is true for circular convolution. So, the circular convolution of two finite-length vectors can be expressed in terms of DFT and IDFT as follows:

$$\mathbf{x} \circledast \mathbf{y} = IDFT(DFT(\mathbf{x}) \cdot DFT(\mathbf{y})) \quad (\text{B.1})$$

where ‘.’ indicates element-wise multiplication.

In a neural circuit, we can take advantage of this identity by noting that $DFT(\mathbf{x})$ is a linear transformation of the vector \mathbf{x} , and can thus be written as a matrix multiplication, $DFT(\mathbf{x}) = \mathbf{C}_{DFT}\mathbf{x}$, where \mathbf{C}_{DFT} is a $D \times D$ constant coefficient matrix that can be pre-computed for a given size of the vector space. The IDFT operation can similarly be written as a matrix multiplication $IDFT(\mathbf{x}) = \mathbf{C}_{DFT}^{-1}\mathbf{x} = \mathbf{C}_{IDFT}\mathbf{x}$. We can thus rewrite circular convolution as matrix multiplication, i.e.

$$\mathbf{x} \circledast \mathbf{y} = \mathbf{C}_{IDFT}((\mathbf{C}_{DFT}\mathbf{x}) \cdot (\mathbf{C}_{DFT}\mathbf{y})) \quad (\text{B.2})$$

in which all of the matrices are constant for any values of \mathbf{x} and \mathbf{y} .

The NEF can be used to compute arbitrary linear transformations of vectors, as well as how to compute point-wise vector products. Consequently, in order to compute the circular convolution $\mathbf{z} = \mathbf{x} \circledast \mathbf{y}$ we simply combine these two techniques.

We must also be able to unbind vectors. To do so, we can invert the convolution operation by defining what is sometimes called a “correlation” operation (\oplus). More simply, we can think of this as standard matrix inversion, i.e.

$$\mathbf{x} = \mathbf{z} \oplus \mathbf{y} = \mathbf{z} \circledast \mathbf{y}^{-1}$$

As discussed by Plate (2003), while most vectors do have an exact inverse under convolution, it is numerically advantageous to use an approximate inverse (called the involution) of a vector in the unbinding process. This is because the exact inverse \mathbf{y}^{-1} is unstable when elements of $DFT(\mathbf{y})$ are near zero.

Involution of a vector \mathbf{x} is defined as $\mathbf{x}' = [x_0, x_{D-1}, x_{D-2}, \dots, x_1]$. Notice that this is simply a flip of all the elements after the first element. Thus, it is a simple permutation of the original matrix. Unbinding then becomes

$$\mathbf{x} = \mathbf{z} \circledast \mathbf{y}'.$$

Notice that the involution is a linear transformation. That is, we can define a permutation matrix \mathbf{S} such that $\mathbf{S}\mathbf{x} = \mathbf{x}'$. As a result, the neural circuit for circular convolution is easily modified to compute circular correlation, by using the matrix $\mathbf{C}_S = \mathbf{C}_{DFT}\mathbf{S}$ in place of the second DFT matrix in the original circuit (equation B.2). Specifically,

$$\mathbf{z} \circledast \mathbf{y}' = \mathbf{C}_{IDFT}((\mathbf{C}_{DFT}\mathbf{z}) \cdot (\mathbf{C}_S\mathbf{y}))$$

where again all matrices are constant for any vectors that need unbinding.

B.2 Learning high-level transformations

To model learning of different transformations in different contexts, we need to derive a biologically plausible learning rule that can infer these transformations. Neumann (2001) noted that to find some unknown transformation \mathbf{T} between two vectors \mathbf{A} and \mathbf{B} , we can solve

$$\mathbf{T} = circ \left(\sum_i^m \mathbf{B}_i \oplus \mathbf{B}_i \right)^{-1} \left(\sum_i^m \mathbf{B}_i \oplus \mathbf{A}_i \right) \quad (B.3)$$

where $circ(\cdot)$ is the circulant matrix, \oplus is circular correlation, i indexes training examples, and m is the number of examples.

Noting $\mathbf{B}_i \oplus \mathbf{B}_i \approx 1$, equation B.3 can be simplified to $\mathbf{T} = \frac{1}{m} \sum_i^m \mathbf{B}_i \oplus \mathbf{A}_i$. Thus, we can define a standard delta rule to determine how to update \mathbf{T} as additional training examples are made available:

$$\mathbf{T}_{i+1} = \mathbf{T}_i - w_i (\mathbf{T}_i - \mathbf{B}_i \oplus \mathbf{A}_i), \quad (B.4)$$

where w_i is an adaptive learning rate inversely proportional to i . Or, we can allow w_i to be a constant, which would cause the effect of train examples on the estimate to exponentially decay. We have used both approaches to successfully run the Raven's model described in section 4.6.

Essentially, equation B.4 updates the current estimate of the transformation vector \mathbf{T} based on the difference between that transformation and what the current examples suggest it should be (i.e., $\mathbf{A}_i \oplus \mathbf{B}_i$). Intuitively, this means that we are estimating the transformation by trying to find the average transformation consistent with all of the available examples.

There are different means of implementing such a rule. For example, in section 4.6, a working memory is used to retain a running estimate of \mathbf{T} by updating the representations in the memory. In such a case, no neural connection weights are changed as the activities of the neurons themselves represent \mathbf{T} . In contrast, in section 6.6, the STDP learning rule is used to learn the mapping from a context signal to desired transformation \mathbf{T} . In this case, the neural connection weights between the context signal(s) and the population generating the relevant transformation vector(s) are manipulated. As such, the rule is indirectly implemented by using it to generate an error signal that trains the mapping. The rule could also be directly embedded into a set of connection weights by using the STDP rule on a circuit whose input was the vector to be transformed, and constructing an error signal appropriately.

B.3 Ordinal serial encoding model

The equations that define the encoding and decoding of the OSE model for serial recall are as follows.

Encoding:

$$\begin{aligned}\mathbf{M}_i^{in} &= \gamma \mathbf{M}_{i-1}^{in} + (\mathbf{P}_i \circledast \mathbf{I}_i) + \mathbf{I}_i \\ \mathbf{M}_i^{epis} &= \rho \mathbf{M}_{i-1}^{epis} + (\mathbf{P}_i \circledast \mathbf{I}_i) + \mathbf{I}_i \\ \mathbf{M}_i^{OSE} &= \mathbf{M}_i^{in} + \mathbf{M}_i^{epis}\end{aligned}$$

Decoding:

$$\mathbf{I}_i = cleanup(\mathbf{M}^{OSE} \circledast \mathbf{P}'_i)$$

where \mathbf{M}^{in} input working memory trace, γ the rate of decay of the old memory trace, \mathbf{M}^{epis} the memory trace in the episodic memory, ρ scaling factor related to primacy, M^{OSE} is the overall encode memory of the list, \mathbf{P} a position vector, \mathbf{I} an item vector, i indexes the associated item number, and $cleanup()$ indicates the application of the clean-up memory to the semantic pointer inside the brackets.

B.4 Spike-timing dependent plasticity

???Update section with new rule:

The rule employed here is a combination of an STDP rule from Pfister and Gerstner (2006) and the rule derived by MacNeil and Eliasmith (ref???) which relates neural activity to the representation of vectors using the NEF methods. This second rule is a Hebbian rule, of the form:

$$\Delta\omega_{ij} = \kappa\alpha_j \mathbf{e}_j \mathbf{E} a_i$$

where ω is the connection weight, κ is the learning rate parameter, α is the gain of the cell, \mathbf{e} is the encoding vector of the cell, \mathbf{E} is an error term, a is the activity of the cell, and i and j index the pre and postsynaptic cells respectively. The gain and encoding vectors are described in more detail in appendix A.1.1. The error term is an input to the cell that captures the error in the performance of the network. This term can be either a modulatory input, like dopamine, or another direct spiking input carrying the relevant information.

The STDP rule from Pfister and Gerstner (2006) is in two parts, one describes the effects of a pre-synaptic spike:

$$\Delta\omega_{ij}(t^{pre}) = -e^{\frac{-(t^{pre}-t^{post1})}{\tau_-}} \left[A_2^- + A_3^- e^{\frac{-(t^{pre}-t^{pre2})}{\tau_x}} \right].$$

The other describes the effect of a postsynaptic spike:

$$\Delta\omega_{ij}(t^{post}) = -e^{\frac{-(t^{post}-t^{pre1})}{\tau_+}} \left[A_2^+ + A_3^+ e^{\frac{-(t^{post}-t^{post2})}{\tau_y}} \right].$$

where ω is the connection weight, $t^{pre/post}$ is the time that the current pre/postsynaptic spike occurred, $t^{post1/pre1}$ is the time that the last post/pre-synaptic spike occurred, $t^{pre2/post2}$ is the time that the last pre/postsynaptic spike occurred, $A_2^{-/+}$ are the weight of the pair-based rule, and $A_3^{-/+}$ are the weight of the triplet-based

rule, $\tau_{+/-}$ determine the shapes of the exponential weights given to the pair-based rule, and $\tau_{x/y}$ determine the shapes of the exponential weights given to the triplet-based rule.

The rule that results from combining these two previous rules follows from substituting the STDP rule in for the activity variable of the NEF rule (i.e. a_i):

$$\Delta\omega_{ij}(t^{pre}) = \kappa\alpha_j \mathbf{e}_j \mathbf{E} \left(-e^{\frac{-(t^{pre}-t^{post1})}{\tau_-}} \left[A_2^- + A_3^- e^{\frac{-(t^{pre}-t^{pre2})}{\tau_x}} \right] \right)$$

and

$$\Delta\omega_{ij}(t^{post}) = \kappa\alpha_j \mathbf{e}_j \mathbf{E} \left(-e^{\frac{-(t^{post}-t^{pre1})}{\tau_+}} \left[A_2^+ + A_3^+ e^{\frac{-(t^{post}-t^{post2})}{\tau_y}} \right] \right).$$

B.5 Number of neurons for representing structure

The following calculations show the number of neurons required by the different architectures discussed. In the text we sometimes assume a simple vocabulary of 6,000 symbols, but here we assume a vocabulary of 60,000 symbols (Crystal, 2003).

For LISA, we can divide the 60,000 symbols into 40,000 nouns and 20,000 relations, and assume all relations are of the form *relation(noun1,noun2)*. More complex relations (including higher-order relations) will require exponentially more neurons.

$$\begin{aligned} \text{groups} &= \text{relations} \times \text{nouns} \times \text{nouns} \\ &= 20\,000 \times 40\,000 \times 40\,000 \\ &= 32\,000\,000\,000\,000 \end{aligned}$$

Each neural group requires multiple neurons to represent it (at least 30).

$$\begin{aligned} \text{neurons} &= \text{groups} \times 100 \\ &= 2\,300\,000\,000\,000\,000 \end{aligned}$$

To determine the cortical area required, we assume ~10 million neurons per square centimeter (22.8 billion neurons (Pakkenberg and Gundersen, 1997) divided by 2,500 cm² of cortex (Peters and Jones, 1984)).

$$\begin{aligned} \text{area} &= \text{neurons} / 10\,000\,000 \\ &= 230\,000\,000\,\text{cm}^2. \end{aligned}$$

For the Neural Blackboard Architecture, the number of neurons required is dominated by the connections between the symbols and the word assemblies. Each connection requires 8 neural groups, the authors suggestion 100 assemblies

per grammatical role (van der Velde & de Kamps, 2006), and we assume at least 100 neurons per neural group are needed to maintain accurate representation.

$$\begin{aligned}\text{groups} &= \text{words} \times \text{assemblies} \times 8 \\ &= 60\,000 \times 100 \times 8 \\ &= 48\,000\,000 \\ \text{neurons} &= \text{groups} \times 100 \\ &= 4\,800\,000\,000 \\ \text{area} &= \text{neurons} / 10\,000\,000 = 480 \text{ cm}^2\end{aligned}$$

For the SPA, the number of dimensions required for similar structures (60,000 words, 8 possible bindings, 99% correct) is about 500 as reported in the main text (see figure 4.7). If, by analogy to the assumptions in LISA and the NBA, we allow each dimension to be a population of 100 neurons, representing any such structure requires:

$$\begin{aligned}\text{neurons} &= \text{dimensions} \times 100 = 50\,000 \\ \text{area} &= \text{neurons}/100\,000 \approx 1 \text{ mm}^2\end{aligned}$$

This does not include the clean up memory, which is crucial for decoding. As mentioned in the main text, this would add 3.5 mm² of cortical area for a total of approximately 5 mm² (see section 4.5).

For the ICS, making the same assumptions as for the SPA, the number neurons required for the sentence “Bill believes John loves Mary” can be determined as follows. First, we allow the sentence to be represented the same as in the SPA:

$S = \text{relation} \otimes \text{believes} + \text{subject} \otimes \text{Bill} + \text{object} \otimes (\text{relation} \otimes \text{loves} + \text{subject} \otimes \text{John} + \text{object} \otimes \text{Mary})$ Each vector is assumed to have 500 dimensions, in order to represent 60,000 words as above, so the maximum number of dimensions is:

$$\begin{aligned}\text{dimensions} &= 500 \times 500 \times 500 = 125\,000\,000 \\ \text{neurons} &= \text{dimensions} \times 100 = 12\,500\,000\,000 \\ \text{area} &= \text{neurons}/10\,000\,000 = 1\,250 \text{ cm}^2\end{aligned}$$

Which is about half the 2,500 cm² of available cortex.

Appendix C

SPA model details

C.1 Tower of Hanoi

This appendix includes two tables. Table C.1 lists all of the cortical elements of the Tower of Hanoi model. Table C.2 lists all of the rules that were used by those elements and the basal ganglia to solve the task.

Table C.1: The model elements of the SPA Tower of Hanoi model found in cortex. The model also includes a basal ganglia and thalamus.

Name	Type	Description
state	buffer	Used to control the different stages of the problem-solving algorithm.
focus	buffer	Stores the disk currently being attended to (D0, D1, D2, D3)
focuspeg	sensory	Automatically contains the location of the focus disk (A, B, C)
goal	buffer	Stores the disk we are trying to move (D0, D1, D2, D3)
goaltarget	buffer	Stores the location we want to move the goal disk to (A, B, C)
goalcurrent	sensory	Automatically contains the location of the goal disk (A, B, C)
goalfinal	sensory	Automatically contains the final desired location of the goal disk (A, B, C)
largest	sensory	Automatically contains the largest visible disk (D3)
mem1	memory	Stores an association between mem1 and mem2 in working memory
mem2	memory	Stores an association between mem1 and mem2 in working memory
request	memory	Indicates one element of a pair to attempt to recall from working memory
recall	memory	The vector associated with the currently requested vector
movedisk	motor	Tells the motor system which disk to move (D0, D1, D2, D3)
movepeg	motor	Tells the motor system where to move the disk to (A, B, C)
motor	sensory	Automatically contains DONE if the motor action is finished

Table C.2: A list of the 16 rules used in the Tower of Hanoi simulation. For the matching, equality signs indicate the result of a dot product, where ‘=’ is summed and ‘≠’ is subtracted. For the execution, all statements refer to setting the lefthand model element to the value indicated on the righthand side.

Rule Name	Match (IF)	Execute (THEN)
LookDone	focus≠D0 + goal=focus + goalcurrent=goaltarget + state≠STORE	focus=goal⊗NEXT goal=goal⊗NEXT goaltarget=goalfinal
LookNotDone	focus≠D0 + goal=focus + goalcurrent≠goaltarget + state≠STORE	focus=goal⊗NEXT
InTheWay1	focus≠goal + focuspeg=goalcurrent + focuspeg≠goaltarget + state≠STORE	focus=goal⊗NEXT
InTheWay2	focus≠goal + focuspeg≠goalcurrent + focuspeg=goaltarget + state≠STORE	focus=goal⊗NEXT
NotInTheWay	focus≠goal + focuspeg≠goalcurrent + focuspeg≠goaltarget + focus≠D0	focus=focus⊗NEXT
MoveD0	focus=D0 + goal=D0 + goalcurrent≠goaltarget	movedisk=D0 movepeg=goaltarget
MoveGoal	focus=D0 + goal≠D0 + focuspeg≠goaltarget + goaltarget≠goalcurrent + focuspeg≠goalcurrent	movedisk=goal movepeg=goaltarget
MoveDone	motor=DONE + goal≠largest + state≠RECALL	state=RECALL goal=goal⊗NEXT ⁻¹
MoveDone2	motor=DONE + goal=largest + state≠RECALL	focus=largest⊗NEXT ⁻¹ goal=largest⊗NEXT ⁻¹ goaltarget=goalfinal state=HANOI
Store	state=STORE + recall≠goaltarget	mem1=goal mem2=goaltarget request=goal
StoreDone	state=STORE + recall=goaltarget	state=FIND
FindFree1	state=FIND + focus≠goal + focuspeg=goalcurrent + focuspeg≠goaltarget	goaltarget=A+B+C-focuspeg -goaltarget goal=focus state=HANOI
FindFree2	state=FIND + focus≠goal +	goaltarget=A+B+C-

Bibliography

- Adolphs, R., Bechara, A., Tranel, D., Damasio, H., and Damasio, A. R. (1995). *Neuropsychological approaches to reasoning and decision-making*. Neurobiology of decision-making. Springer Verlag, New York.
- Albin, R. L., Young, A. B., and Penney, J. B. (1989). The functional anatomy of basal ganglia disorders. *Trends in Neurosciences*, 12:366–375.
- Aldridge, J. W., Berridge, K., Herman, M., and Zimmer, L. (1993). Neuronal coding of serial order: Syntax of grooming in the neostriatum. *Psychological Science*, 4(6):391–395.
- Allport, D. A. (1985). Distributed memory, modular subsystems and dysphasia. In Newman, S. K. and Epstein, R., editors, *Current Perspectives in Dysphasia*, pages 207–244. Churchill Livingstone, Edinburgh.
- Almor, A. and Sloman, S. A. (1996). Is deontic reasoning special? *Psychological Review*, 103:374–380.
- Altmann, E. M. and Trafton, J. G. (2002). Memory for goals: An activation-based model. *Cognitive Science*, 26:39–83.
- Amari, S. (1975). Homogeneous nets of neuron-like elements. *Biological Cybernetics*, 17:211–220.
- Amit, D. J. (1989). *Modeling brain function: The world of attractor neural networks*. Cambridge University Press, New York, NY.
- Amit, D. J., Fusi, S., and Yakovlev, V. (1997). A paradigmatic working memory (attractor) cell in IT cortex. *Neural Computation*, 9(5):1071–1092.
- Andersen, R. A., Essick, G. K., and Siegel, R. M. (1985). The encoding of spatial location by posterior parietal neurons. *Science*, 230:456–458.

- Anderson, J. (1996). Act: A simple theory of complex cognition. *American Psychologist*, 51(4):355–365.
- Anderson, J., John, B. E., Just, M., Carpenter, P. A., Kieras, D. E., and Meyer, D. E. (1995a). Production system models of complex cognition. In Moore, J. D. and Lehman, J. F., editors, *Proceedings of the Seventeenth Annual Conference of the Cognitive Science ...*, page 9. Routledge.
- Anderson, J. R. (1976). *Language, memory, and thought*. Lawrence Erlbaum, Hillsdale, NJ.
- Anderson, J. R. (1983). *The architecture of cognition*. Harvard University Press, Cambridge, MA.
- Anderson, J. R. (2007). *How can the human mind occur in the physical universe?* Oxford University Press.
- Anderson, J. R., Albert, M. V., and Fincham, J. M. (2005). Tracing problem solving in real time: fMRI analysis of the subject-paced Tower of Hanoi. *Journal of cognitive neuroscience*, 17(8):1261–74.
- Anderson, J. R. and Betz, J. (2001). A hybrid model of categorization. *Psychonomic Bulletin and Review*, 8:629–647.
- Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., and Qin, Y. (2004). An integrated theory of the mind. *Psychological Review*, 111(4):1036–1060.
- Anderson, J. R., Corbett, A., Koedinger, K. R., and Pelletier, R. (1995b). Cognitive tutors: Lessons learned. *Journal of the Learning Sciences*, 4(2):167–207.
- Anderson, J. R., Kushmerick, N., and Lebiere, C. (1993). *Tower of Hanoi and goal structures*. Erlbaum Associates.
- Anderson, J. R. and Lebiere, C. (1998). *The atomic components of thought*. Erlbaum Associates.
- Anderson, M. (2010). Neural re-use as a fundamental organizational principle of the brain. *Behavioral and Brain Sciences*, 33(3).

- Askay, E., Baker, R., Seung, H. S., and Tank, D. (2000). Anatomy and discharge properties of pre-motor neurons in the goldfish medulla that have eye-position signals during fixations. *Journal of Neurophysiology*, 84:1035–1049.
- Aziz-Zadeh, L. and Damasio, A. (2008). Embodied semantics for actions: findings from functional brain imaging. *Journal of physiology, Paris*, 102(1-3):35–9.
- Baars, B. (1988). *A cognitive theory of consciousness*. Cambridge University Press, New York, NY.
- Baddeley, A. (1998). Recent developments in working memory. *Current opinion in neurobiology*, 8(2):234–238.
- Ballard, D. H. (1991). Animate vision. *Artificial Intelligence*, 48:57–86.
- Barr, M. (2004). Visual objects in context. *Nature Reviews Neuroscience*, 5:617–629.
- Barsalou, L. W. (1999). Perceptual symbol systems. *Behavioral and Brain Sciences*, 22:577–660.
- Barsalou, L. W. (2003). Abstraction in perceptual symbol systems. *Philosophical Transactions of the Royal Society of London: Biological Sciences*, 358:1177–1187.
- Barsalou, L. W. (2009). Simulation, situated conceptualization, and prediction. *Philosophical Transactions of the Royal Society of London: Biological Sciences*, 364(1281-1289).
- Batterman, R. (2002). *The Devil in the Details: Asymptotic Reasoning in Explanation, Reduction, and Emergence*. Oxford University Press, (New York).
- Bayer, H. M. and Glimcher, P. W. (2005). Midbrain dopamine neurons encode a quantitative reward prediction error signal. *Neuron*, 47:129–141.
- Beal, M. (1998). Variational algorithms for approximate Bayesian inference.
- Bechtel, W. (1988). *Philosophy of science: An introduction for cognitive science*. Erlbaum, Hillsdale, NJ.

- Bechtel, W. (1998). Representations and cognitive explanations: Assessing the dynamicist challenge in cognitive science. *Cognitive Science*, 22:295–318.
- Bechtel, W. (2005). The challenge of characterizing operations in the mechanisms underlying behavior. *Journal of the Experimental Analysis of Behavior*, 84:313–325.
- Bechtel, W. (2008). *Mental mechanisms: Philosophical Perspectives on Cognitive Neuroscience*. Routledge.
- Bechtel, W. and Abrahamsen, A. (2001). *Connectionism and the mind: Parallel processing, dynamics, and evolution in networks*, volume 2nd. Blackwell, Oxford.
- Bechtel, W. and Graham, G. (1999). *A companion to cognitive science*. Blackwell, London.
- Bechtel, W. and Richardson, R. C. (1993). *Discovering complexity: Decomposition and localization as strategies in scientific research*. Princeton University Press, Princeton, NJ.
- Becker, S. (2005). A computational principle for hippocampal learning and neurogenesis. *Hippocampus*, 15(6):722–738.
- Beckmann, H. and Lauer, M. (1997). The human striatum in schizophrenia. II. Increased number of striatal neurons in schizophrenics. *Psychiatry research*, 68(2-3):99–109.
- Beiser, D. G. and Houk, J. C. (1998). Model of cortical-basal ganglionic processing: Encoding the serial order of sensory events. *Journal of Neurophysiology*, 79:3168–3188.
- Bekolay, T. and Eliamith, C. (2011). Learning cognitive operation in spiking neural networks. In *Proceedings of the 33rd annual conference of the cognitive science society*.
- Bekolay, T. and Eliasmith, C. (2011). A general error-modulated STDP learning rule applied to reinforcement learning in the basal ganglia. In *Computational and Systems Neuroscience 2011*, Salt Lake City, UT.
- Bernstein, N. (1967). *The Coordination and Regulation of Movements*. Pergamon Press, New York.

- Bi, G. Q. and Poo, M. M. (1998). Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 18(24):10464–72.
- Bialek, W. and Rieke, F. (1992). Reliability and information transmission in spiking neurons. *Trends in Neuroscience*, 15(11):428–434.
- Bienenstock, E., Cooper, L. N., and Munro, P. (1982). On the development of neuron selectivity: Orientation specificity and binocular interaction in visual cortex. *Journal of Neuroscience*, 2:32–48.
- Bingham, E. and Mannila, H. (2001). Random projection in dimensionality reduction. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '01*, pages 245–250, New York, New York, USA. ACM Press.
- Bobier, B. (2011). *The attentional routing circuit: A model of selective visuospatial attention*. Phd, University of Waterloo.
- Bors, D. (2003). The effect of practice on Raven's Advanced Progressive Matrices. *Learning and Individual Differences*, 13(4):291–312.
- Botvinick, M. M. and Plaut, D. C. (2006). Short-Term Memory for Serial Order: A Recurrent Neural Network Model. *Psychological Review*, 113(2):201–233.
- Bower, J. M. and Beeman, D. (1998). *The book of GENESIS: Exploring realistic neural models with the GEneral NEural SImulation System*. Springer Verlag.
- Bowers, J. S. (2009). On the Biological Plausibility of Grandmother Cells : Implications for Neural Network Theories in Psychology and Neuroscience. *Psychological Review*, 116(1):220 –251.
- Brady, M. (2009). Speech as a Problem of Motor Control in Robotics. In Taatgen, N. and van Rijn, H., editors, *Proceedings of the Thirty-First Annual Conference of the Cognitive Science Society*, pages 2558–2563.
- Brooks, R. (1991). Intelligence without representation. *Artificial Intelligence*, 47:139–159.
- Bruner, J. S., Goodnow, J. J., and Austin, G. A. (1956). *A study of thinking*. Wiley, New York.

- Buckner, R. L. and Wheeler, M. E. (2001). The cognitive neuroscience of remembering. *Nature Reviews Neuroscience*, 2:626–634.
- Budiu, R. and Anderson, J. R. (2004). Interpretation-Based Processing: A Unified Theory of Semantic Sentence Processing. *Cognitive Science*, 28:1–44.
- Buffalo, E. A., Fries, P., Landman, R., Liang, H., and Desimone, R. (2010). A backward progression of attentional effects in the ventral stream. *Proceedings of the National Academy of Sciences of the United States of America*, 107(1):361–5.
- Bunge, S. A. (2006). A Brain-Based Account of the Development of Rule Use in Childhood. *Current Directions in Psychological Science*, 15(3):118 – 121.
- Burger, P., Mehlb, E., Cameron, P. L., Maycoxa, P. R., Baumert, M., Friedrich, L., De Camilli, P., and Jahn, R. (1989). Synaptic vesicles immunoisolated from rat cerebral cortex contain high levels of glutamate. *Neuron*, 3(6):715–720.
- Busemeyer, J. R. and Townsend, J. T. (1993). Decision field theory: A dynamic-cognitive approach to decision making in an uncertain environment. *Psychological review*, 100(3):432–459.
- Calabresi, P., Picconi, B., Tozzi, A., and Di Filippo, M. (2007). Dopamine-mediated regulation of corticostriatal synaptic plasticity. *Trends in neurosciences*, 30(5):211–9.
- Canessa, N., Gorini, A., Cappa, S. F., Piattelli-Palmarini, M., Danna, M., Fazio, F., and Perani, D. (2005). The effect of social content on deductive reasoning: an fMRI study. *Human brain mapping*, 26(1):30–43.
- Caporale, N. and Dan, Y. (2008). Spike timing-dependent plasticity: a Hebbian learning rule. *Annual review of neuroscience*, 31:25–46.
- Carnap, R. (1931). Psychology in the language of physics. *Erkenntnis*, 2.
- Carnevale, N. and Hines, M. (2006). *The NEURON Book*. Cambridge University Press, Cambridge, UK.
- Carpenter, P., Just, M., and Shell, P. (1990). What one intelligence test measures: a theoretical account of the processing in the Raven Progressive Matrices Test. *Psychological Review*, 97(3):404–31.

- Chaabani, I. and Scheessele, M. R. (2007). Human Performance on the USPS Database. Technical report, Indiana University South Bend.
- Cheng, P. W. and Holyoak, K. J. (1985). Pragmatic reasoning schemas. *Cognitive psychology*, 17:391–416.
- Chomsky, N. (1959). A review of B. F. Skinner's Verbal Behavior. *Language*, 35(1):26–58.
- Choo, X. (2010). *The Ordinal Serial Encoding Model: Serial Memory in Spiking Neurons*. Masters, University of Waterloo.
- Churchland, M. M., Cunningham, J. P., Kaufman, M. T., Ryu, S. I., and Shenoy, K. V. (2010). Cortical Preparatory Activity: Representation of Movement or First Cog in a Dynamical Machine? *Neuron*, 68(3):387–400.
- Churchland, P. (1979). *Scientific realism and the plasticity of mind*. Cambridge University Press, Cambridge.
- Churchland, P. S., Ramachandran, V. S., and Sejnowski, T. J. (1994). *A critique of pure vision*. Large-scale neuronal theories of the brain. MIT Press, Cambridge, MA.
- Clark, A. (1997). *Being there: Putting brain, body and world together again*. MIT Press, Cambridge, MA.
- Clark, A. and Chalmers, D. (2002). *The extended mind*. Philosophy of mind: Classical and contemporary readings. Oxford University Press.
- Collins, A. and Quillian, M. (1969). Retrieval time from semantic memory. *Journal of Verbal Learning and Verbal Behavior*, 8(2):240–247.
- Conklin, J. and Eliasmith, C. (2005). An attractor network model of path integration in the rat. *Journal of Computational Neuroscience*, 18:183–203.
- Cooper, R. (2002). *Modelling High-Level Cognitive Processes*. Lawrence Erlbaum Associates, Hillsdale, N.J.
- Cosmides, L. (1989). The logic of social exchange: Has natural selection shaped how humans reason? Studies with the Wason selection task. *Cognition*, 31:187–276.

- Cowan, N. (2001). The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *Behavioral and Brain Sciences*, 24:87–185.
- Cox, J. R. and Griggs, R. A. (1982). The effects of experience on performance in Wason's selection task. *Memory & cognition*, 10(5):496–502.
- Craver, C. (2007). *Explaining the brain*. Oxford University Press, Oxford, UK.
- Crystal, D. (2003). *Cambridge encyclopedia of the english language*. Cambridge University Press, Cambridge, UK.
- Damasio, A. R. (1989). Time-locked multiregional retroactivation: a systems-level proposal for the neural substrates of recall and recognition. *Cognition*, 33:25–62.
- Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990a). Indexing By Latent Semantic Analysis. *Journal of the American Society For Information Science*, 41:391–407.
- Deerwester, S., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990b). Indexing by latent semantic analysis. *Journal of the American Society For Information Science*, 41(6):391–407.
- DeLong, M. R. (1990). Primate models of movement disorders of basal ganglia origin. *Trends in Neurosciences*, 13(7):281–285.
- Dempster, A., Laird, N., and Rubin, D. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society Series B*, 39(1):1–38.
- Dennett, D. and Viger, C. (1999). Sort-of symbols? *Behavioral and Brain Sciences*, 22(4):613.
- Dewolf, T. (2010). *NOCH: A framework for biologically plausible models of neural motor control*. Masters thesis, University of Waterloo.
- Dewolf, T. and Eliamith, C. The neural optimal control hierarchy for motor control. *Neural Engineering*.
- DeWolf, T. and Eliasmith, C. (2010). NOCH: A framework for biologically plausible models of neural motor control. In *Neural Control of Movement 20th Annual Conference*, Naples, FL.

- Dretske, F. (1994). *If you can't make one, you don't know how it works*, chapter XIX, pages 615–678. Midwest Studies in Philosophy. University of Minnesota Press, Minneapolis.
- Edelman, S. and Breen, E. (1999). On the virtues of going all the way. *Behavioral and Brain Sciences*, 22(4):614.
- Eliasmith, C. (1995). Mind as a dynamic system.
- Eliasmith, C. (1996). The third contender: a critical examination of the dynamicist theory of cognition. *Philosophical Psychology*, 9(4):441–463.
- Eliasmith, C. (1997). Computation and dynamical models of mind. *Minds and Machines*, 7:531–541.
- Eliasmith, C. (1998). Commentary: Dynamical models and van Gelder's dynamism: Two different things. *Behavioral and Brain Sciences*, 21:616–665.
- Eliasmith, C. (2000). *How neurons mean: A neurocomputational theory of representational content*. phdthesis, Washington University in St. Louis, Department of Philosophy.
- Eliasmith, C. (2003). Moving beyond metaphors: Understanding the mind for what it is. *Journal of Philosophy*, 100:493–520.
- Eliasmith, C. (2004). *Learning context sensitive logical inference in a neurobiological simulation*, pages 17–20. AAAI Press.
- Eliasmith, C. (2005a). A new perspective on representational problems. *Journal of Cognitive Science*, 6:97–123.
- Eliasmith, C. (2005b). A unified approach to building and controlling spiking attractor networks. *Neural computation*, 17(6):1276–1314.
- Eliasmith, C. (2005c). Cognition with neurons: A large-scale, biologically realistic model of the Wason task. In Bara, G., Barsalou, L., and Bucciarelli, M., editors, *Proceedings of the 27th Annual Meeting of the Cognitive Science Society*, pages 624–630.
- Eliasmith, C. (2006). *Neurosemantics and categories*. Handbook of Categorization in Cognitive Science. Elsevier, Amsterdam.

- Eliasmith, C. (2009a). *Dynamics, control, and cognition*. New York, NY.
- Eliasmith, C. (2009b). How we ought to understand computation in the brain. *Studies in History and Philosophy of Science*, 41:313–320.
- Eliasmith, C. and Anderson, C. H. (2001). Beyond bumps: Spiking networks that store sets of functions. *Neurocomputing*, 38:581–586.
- Eliasmith, C. and Anderson, C. H. (2003). *Neural engineering: Computation, representation and dynamics in neurobiological systems*. MIT Press, Cambridge, MA.
- Eliasmith, C. and Thagard, P. (1997). Particles, waves and explanatory coherence. *British Journal of the Philosophy of Science*, 48:1–19.
- Engel, A. K., Fries, P., and Singer, W. (2001). Dynamic predictions: oscillations and synchrony in top-down processing. *Nature reviews.Neuroscience*, 2(10):704–716.
- Evans, G. (1982). *Varieties of reference*. Oxford University Press, New York.
- Faubel, C. and Sch, G. (2010). Learning Objects on the Fly: Object Recognition for the Here and Now. In *International Joint Conference on Neural Networks*. IEEE Press.
- Feeney, A. and Handley, S. J. (2000). The suppression of Q card selections: Evidence for deductive inference in Wason’s selection task. *Quarterly Journal of Experimental Psychology*, 53(4):1224–1242.
- Feldman, D. E. (2009). Synaptic mechanisms for plasticity in neocortex. *Annual review of neuroscience*, 32:33–55.
- Felleman, D. J. and Essen, D. C. V. (1991). Distributed hierarchical processing in primate visual cortex. *Cerebral Cortex*, 1:1–47.
- Felleman, D. J., Xiao, Y., and McClendon, E. (1997). Modular organization of occipito-temporal pathways: Cortical connections between visual area 4 and visual area 2 and posterior inferotemporal ventral area in macaque monkeys. *Journal of Neuroscience*, 17(9):3185–3200.
- Fillmore, C. (1975). An alternative to checklist theories of meaning. In *First Annual Meeting of the Berkeley Linguistics Society*, pages 123–131.

- Fischer, B. J. (2005). *A model of the computations leading to a representation of auditory space in the midbrain of the barn owl*. Phd, Washington University in St. Louis.
- Fischer, B. J., Peña, J. L., and Konishi, M. (2007). Emergence of multiplicative auditory responses in the midbrain of the barn owl. *Journal of neurophysiology*, 98(3):1181–93.
- Fishbein, J. M. (2008). *Integrating Structure and Meaning: Using Holographic Reduced Representations to Improve Automatic Text Classification*. Masters, University of Waterloo.
- Fleetwood, M. and Byrne, M. (2006). Modeling the Visual Search of Displays: A Revised ACT-R Model of Icon Search Based on Eye-Tracking Data. *Human-Computer Interaction*, 21(2):153–197.
- Fodor (1974). Special sciences (or: The disunity of science as a working hypothesis). *Synthese*, 28(2):97.
- Fodor, J. (1975). *The language of thought*. Crowell, New York.
- Fodor, J. (1987). *Psychosemantics*. MIT Press, Cambridge, MA.
- Fodor, J. (1995). West coast fuzzy: Why we don't know how brains work (review of Paul Churchland's The engine of reason, the seat of the soul). *The Times Literary Supplement*, (August).
- Fodor, J. (1998). *Concepts: Where cognitive science went wrong*. Oxford University Press, New York.
- Fodor, J. and McLaughlin, B. (1990). Connectionism and the problem of systematicity: Why Smolensky's solution doesn't work. *Cognition*, 35:183–204.
- Fodor, J. and Pylyshyn, Z. (1988a). Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28:3–71.
- Fodor, J. and Pylyshyn, Z. (1988b). Connectionism and cognitive science: A critical analysis. *Behavioral and Brain Sciences*, 28:3–71.
- Forbes, A. R. (1964). An item analysis of the advanced matrices. *British Journal of Educational Psychology*, 34:1–14.

- Frank, M. J. and O'Reilly, R. C. (2006). A mechanistic account of striatal dopamine function in human cognition: psychopharmacological studies with cabergoline and haloperidol. *Behavioral neuroscience*, 120(3):497–517.
- Fries, P. (2009). Neuronal gamma-band synchronization as a fundamental process in cortical computation. *Annual review of neuroscience*, 32:209–24.
- Friston, K. (2003). Learning and inference in the brain. *Neural Networks*, 16(9):1325–1352.
- Fuster, J. M. (2000). Executive frontal functions. *Experimental brain research*, 133(1):66–70.
- Gayler, R. W. (1998). Multiplicative binding, representation operators and analogy. In Holyoak, K., Gentner, D., and Kokinov, B., editors, *Advances in analogy research: Integration of theory and data from the cognitive, computational, and neural sciences*.
- Gayler, R. W. (2003). Vector Symbolic Architectures answer Jackendoff's challenges for cognitive neuroscience. In Slezak, P., editor, *ICCS/ASCS International Conference on Cognitive Science*, pages 133–138.
- Georgopoulos, A. P., Kalasaka, J. F., Crutcher, M. D., Caminiti, R., and Massey, J. T. (1984). The representation of movement direction in the motor cortex: Single cell and population studies. In Edelman, G. M., Gail, W. E., and Cowan, W. M., editors, *Dynamic aspects of neocortical function*. Neurosciences Research Foundation.
- Georgopoulos, A. P., Schwartz, A. B., and Kettner, R. E. (1986). Neuronal population coding of movement direction. *Science*, 243(1416-19).
- Georgopoulos, A. P., Taira, M., and Lukashin, A. (1993). Cognitive neurophysiology of the motor cortex. *Science*, 260:47–52.
- Gershman, S., Cohen, J., and Niv, Y. (2010). Learning to selectively attend. In *Proceedings of the 32nd annual conference of the cognitive science society*, pages 1270–1275.
- Gibbs Jr., R. W. (2006). *Embodiment and Cognitive Science*. Cambridge. Cambridge University Press, Cambridge.

- Gibson, J. J. (1977). *The Theory of Affordances*.
- Gisiger, T. and Boukadoum, M. (2011). Mechanisms Gating the Flow of Information in the Cortex: What They Might Look Like and What Their Uses may be. *Frontiers in Computational Neuroscience*, 5(January):1–15.
- Glaser, W. R. (1992). Picture naming. *Cognition*, 42:61–105.
- Goel, V. (2005). *Cognitive Neuroscience of Deductive Reasoning*. Cambridge Handbook of Thinking & Reasoning. Cambridge University Press.
- Gonchar, Y. and Burkhalter, A. (1999). Connectivity of GABAergic calretinin-immunoreactive neurons in rat primary visual cortex. *Cerebral Cortex*, 9(7):683–696.
- Goodale, M. A. and Milner, A. D. (1992). Separate pathways for perception and action. *Trends in Neuroscience*, 15:20–25.
- Gray, J. R., Chabris, C. F., and Braver, T. S. (2003). Neural mechanisms of general fluid intelligence. *Nature Neuroscience*, 6(3):316–22.
- Gupta, A., Wang, Y., and Markram, H. (2000). Organizing Principles for a Diversity of GABAergic Interneurons and Synapses in the Neocortex. *Science*, 287:273–278.
- Gurney, K., Prescott, T., and Redgrave, P. (2001). A computational model of action selection in the basal ganglia. *Biological Cybernetics*, 84:401–423.
- Hadley, R. F. (2009). The problem of rapid variable creation. *Neural computation*, 21(2):510–32.
- Hardie, J. and Spruston, N. (2009). Synaptic depolarization is more effective than back-propagating action potentials during induction of associative long-term potentiation in hippocampal pyramidal neurons. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 29(10):3233–41.
- Harman, G. (1982). Conceptual role semantics. *Notre Dame Journal of Formal Logic*, 23:242–256.
- Harnad, S. (1990). The symbol grounding problem. *Physica D*, 42:335–346.

- Haugeland, J. (1993). Mind embedded and embodied. In *Mind and Cognition: An International Symposium*, Taipei, Taiwan. Academia Sinica.
- Hayes-Roth, B. and Hayes-Roth, F. (1979). A cognitive model of planning. *Cognitive Science*, 3:275–310.
- Hazy, T. E., Frank, M. J., and O’reilly, R. C. (2007). Towards an executive without a homunculus: computational models of the prefrontal cortex/basal ganglia system. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, 362(1485):1601–13.
- Hebb, D. O. (1949). *The organization of behavior*. Wiley, New York, NY.
- Hellwig, B. (2000). A quantitative analysis of the local connectivity between pyramidal neurons in layers 2/3 of the rat visual cortex. *Biological Cybernetics*, 82(2):111–121.
- Hempel, C. G. (1966). *Philosophy of natural science*. Prentice-Hall, Englewood Cliffs, N J.
- Henson, R. (1998). Short-term memory for serial order: The start-end model. *Cognitive psychology*, 36:73–137.
- Henson, R., Noriss, D., Page, M., and Baddeley, A. (1996). Unchained memory: Error patterns rule out chaining models of immediate serial recall. *The quarterly journal of experimental psychology A*, 49(1):80–115.
- Herd, S. a., Banich, M. T., and O'Reilly, R. C. (2006). Neural mechanisms of cognitive control: an integrative model of stroop task performance and fMRI data. *Journal of cognitive neuroscience*, 18(1):22–32.
- Hier, D., Yoon, W., Mohr, J., Price, T., and Wolf, P. (1994). Gender and aphasia in the stroke data bank. *Brain and Language*, 47:155–167.
- Hinton, G. (1990). Connectionist learning procedures, pages 185–234. Machine learning: Paradigms and methods. MIT Press, Cambridge MA.
- Hinton, G. (2010). Where do features come from? In *Outstanding questions in cognitive science: A symposium honoring ten years of the David E. Rumelhart prize in cognitive science*. Cognitive Science Society.

- Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.
- Hochstein, E. (2011). *Intentionality as methodology*. PhD thesis, University of Waterloo.
- Hollerman, J. R. and Schultz, W. (1998). Dopamine neurons report an error in the temporal prediction of reward during learning. *Nature neuroscience*, 1(4):304–9.
- Holmgren, C., Harkany, T., Svinnenfors, B., and Zilberman, Y. (2003). Pyramidal cell communication within local networks in layer 2/3 of rat neocortex. *The Journal of physiology*, 551(Pt 1):139–53.
- Holyoak, K. and Thagard, P. (1995). *Mental leaps: Analogy in creative thought*. MIT Press, Cambridge, MA.
- Hoshi, E. (2006). Functional specialization within the dorsolateral prefrontal cortex: a review of anatomical and physiological studies of non-human primates. *Neuroscience research*, 54(2):73–84.
- Hummel, J. E., Burns, B., and Holyoak, K. J. (1994). *Analogical mapping by dynamic binding: Preliminary investigations*, chapter 2. Advances in connectionist and neural computation theory: Analogical connections. Ablex, Norwood, NJ.
- Hummel, J. E. and Holyoak, K. J. (1997). Distributed representations of structure: a theory of analogical access and mapping. *Psychology Review*, 104(3):427–466.
- Hummel, J. E. and Holyoak, K. J. (2003). A symbolic-connectionist theory of relational inference and generalization. *Psychological review*, 110(2):220–264.
- Hurwitz, M. (2010). *Dynamic judgments of spatial extent: Behavioural, neural and computational studies*. PhD, University of Waterloo.
- Hutchins, E. (1995). *Cognition in the Wild*. MIT Press, Cambridge, MA.
- Ikezu, T. and Gendelman, H. (2008). *Neuroimmune Pharmacology*. Springer.
- Jackendoff, R. (2002). *Foundations of language: Brain, meaning, grammar, evolution*. Oxford University Press.

- Jahnke, J. C. (1968). Delayed recall and the serial-position effect of short-term memory. *Journal of Experimental Psychology*, 76(4):618.
- Jancke, D., Erlhagen, W., Dinse, H., Akhavan, A., Steinhage, A., Schöner, G., and Giese, M. (1999). Population representation of retinal position in cat primary visual cortex: interaction and dynamics. *Journal of Neuroscience*, 19(20):9016–9028.
- Jilk, D., Lebiere, C., O'Reilly, R., and Anderson, J. (2008). SAL: an explicitly pluralistic cognitive architecture. *Journal of Experimental & Theoretical Artificial Intelligence*, 20(3):197–218.
- Johnson-Laird, P. N. (1983). *Mental models: Towards a cognitive science of language, inference, and consciousness*. Harvard Press, Cambridge, MA.
- Johnson-Laird, P. N., Legrenzi, P., and Legrenzi, S. (1972). Reasoning and a sense of reality. *British Journal of Psychology*, 63:395–400.
- Kalisch, R., Korenfeld, E., Stephan, K. E., Weiskopf, N., Seymour, B., and Dolan, R. J. (2006). Context-dependent human extinction memory is mediated by a ventromedial prefrontal and hippocampal network. *Journal of Neuroscience*, 26(37):9503–9511.
- Kan, I. P., Barsalou, L. W., Solomon, K. O., Minor, J. K., and Thompson-Schill, S. L. (2003). Role of mental imagery in a property verification task: fMRI evidence for perceptual representations of conceptual knowledge. *Cognitive Neuropsychology*, 20:525–540.
- Kandel, E., Schwartz, J. H., and Jessell, T. M. (2000). *Principles of neural science*. McGraw Hill, New York, NY.
- Kanerva, P. (1994). *The spatter code for encoding concepts at many levels*, chapter 1, pages 226–229. Proceedings of the International Conference on Artificial Neural Networks. Springer-Verlag, Sorrento, Italy.
- Kaplan, G. B. and Gzelis, C. (2001). Hopfield networks for solving Tower of Hanoi problems. *ARI: An Interdiscipl. J. Phys. Eng. Sci.*, 52:23.
- Kawato, M. (1995). Cerebellum and motor control. In Arbib, M., editor, *The handbook of brain theory and neural networks*. MIT Press, Cambridge, MA.

- Khan, M., Lester, D., Plana, L., Rast, A., Jin, X., Painkras, E., and Furber, S. (2008). *SpiNNaker: Mapping neural networks onto a massively-parallel chip multiprocessor*. IEEE.
- Kieras, D. E. and Meyer, D. E. (1997). An overview of the EPIC architecture for cognition and performance with application to human-computer interaction. *Human-Computer Interaction*, 4(12):391–438.
- Kim, H., Sul, J. H., Huh, N., Lee, D., and Jung, M. W. (2009). Role of striatum in updating values of chosen actions. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 29(47):14701–12.
- Kim, R., Alterman, R., Kelly, P. J., Fazzini, E., Eidelberg, D., Beric, A., and Sterio, D. (1997). Efficacy of bilateral pallidotomy. *Neurosurgical FOCUS*, 2(3):E10.
- Kitcher, P. (1993). *The advancement of science*. Oxford University Press, Oxford.
- Koch, C. (1999). *Biophysics of computation: Information processing in single neurons*. Oxford University Press, New York, NY.
- Kosslyn, S. M., Ganis, G., and Thompson, W. L. (2000). Neural foundations of imagery. *Nature Reviews Neuroscience*, 2(635–642).
- Koulakov, A. A., Raghavachari, S., Kepecs, A., and Lisman, J. E. (2002). Model for a robust neural integrator. *Nature neuroscience*, 5(8):775–782.
- Kuo, D. and Eliasmith, C. (2005). Integrating behavioral and neural data in a model of zebrafish network interaction. *Biological Cybernetics*, 93(3):178–187.
- Lakoff, G. (1987). *Women, fire, and dangerous things*. University of Chicago Press, Chicago.
- Landauer, T. and Dumais, S. T. (1997). A solution to plato's problem: The latent semantic analysis theory of acquisition, induction and representation of knowledge. *Psychological Review*, 104(2):211–240.
- Landers, D. M. and Feltz, D. L. (2007). *The Effects of Mental Practice on Motor Skill Learning and Performance: A Meta-Analysis*. Human Kinetics.

- Langacker, R. W. (1986). An introduction to cognitive grammar. *Cognitive Science*, 10:1–40.
- Lass, Y. and Abeles, M. (1975). Transmission of information by the axon. {I}: Noise and memory in the myelinated nerve fiber of the frog. *Biological Cybernetics*, 19:61–67.
- Laubach, M., Caetano, M. S., Liu, B., Smith, N. J., Narayanan, N. S., and Eliasmith, C. (2010). Neural circuits for persistent activity in medial prefrontal cortex. In *Society for Neuroscience Abstracts*, page 200.18.
- Lebiere, C. and Anderson, J. R. (1993). A connectionist implementation of the ACT-R production system. In *Fifteenth Annual Conference of the Cognitive Science Society*, pages 635–640, Hillsdale, NJ. Erlbaum.
- Lee, H., Ekanadham, C., and Ng, A. (2007). Sparse deep belief net model for visual area V2. *Advances in neural information processing systems*, 20:1–8.
- Lee, J. and Maunsell, J. (2010). The effect of attention on neuronal responses to high and low contrast stimuli. *Journal of neurophysiology*, 104(2):960–971.
- Lee, W. (1984). Neuromotor synergies as a basis for coordinated intentional action. *Journal of Motor Behavior*, 16:135–170.
- Legendre, G., Miyata, Y., and Smolensky, P. (1994). *Principles for an integrated connectionist/symbolic theory of higher cognition*. Lawrence Erlbaum Associates, Hillsdale, NJ.
- Lewis, F. L. (1992). *Applied optimal control and estimation*. Prentice-Hall, New York.
- Lieberman, P. (2006). *Toward an evolutionary biology of language*. The Belknap Press of Harvard University Press, Cambridge, MA.
- Lieberman, P. (2007). The Evolution of Human Speech: Its Anatomical and Neural Bases. *Current Anthropology*, 48(1).
- Liepa, P. (1977). Models of content addressable distributed associative memory. Technical report, University of Toronto.

- Lipinski, J., Spencer, J. P., Samuelson, L. K., and Schöner, G. (2006). Spamling: A dynamical model of spatial working memory and spatial language. In *Proceedings of the 28th Annual Conference of the Cognitive Science Society*, pages 768–773, Vancouver, Canada.
- Litt, A., Eliasmith, C., and Thagard, P. (2008). Neural affective decision theory: Choices, brains, and emotions. *Cognitive Systems Research*, 9:252–273.
- Lund, J. S., Yoshioka, T., and Levitt, J. B. (1993). Comparison of intrinsic connectivity in different areas of macaque monkey cerebral cortex. *Cerebral cortex (New York, N.Y. : 1991)*, 3(2):148–62.
- Machamer, P., Darden, L., and Craver, C. (2000). Thinking about mechanisms. *Philosophy of Science*, 67:1–25.
- Machens, C. K., Romo, R., and Brody, C. D. (2010). Functional, but not anatomical, separation of "what" and "when" in prefrontal cortex. *The Journal of Neuroscience*, 30(1):350–60.
- Machery, E. (2009). *Doing Without Concepts*. Oxford University Press, New York.
- Maia, T. V. (2009). Reinforcement learning, conditioning, and the brain: Successes and challenges. *Cognitive, affective & behavioral neuroscience*, 9(4):343–64.
- Maratsos, M. and Kuczaj, S. (1976). Is “Not N’t”? A Study in Syntactic Generalization. Technical report, Education Resources Information Centre, Standford University Dept. of Linguistics.
- Marcus, G. F. (2001). *The algebraic mind*. MIT Press, Cambridge, MA.
- Markram, H., Lübke, J., Frotscher, M., and Sakmann, B. (1997). Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs. *Science (New York, N.Y.)*, 275(5297):213–5.
- Marr, D. (1982). *Vision*. Freeman, San Francisco.
- Martin, A. (2007). The representation of object concepts in the brain. *Annual Review of Psychology*, 58:25–45.

- McClelland, J. L. and Rumelhart, D. E. (1986). *Parallel distributed processing: Explorations in the microstructure of cognition*. Number 2. MIT Press/Bradford Books, Cambridge MA.
- McCormick, D. A., Connors, B. W., Lighthall, J. W., and Prince, D. A. (1985). Comparative electrophysiology of pyramidal and sparsely spiny stellate neuron. *Journal of Neurophysiology*, 54:782–806.
- McKay, T. J. (1999). *Reasons, Explanations, and Decisions: Guidelines for Critical Thinking*. Wadsworth Publishing.
- Mehta, A. D. (2000). Intermodal Selective Attention in Monkeys. I: Distribution and Timing of Effects across Visual Areas. *Cerebral Cortex*, 10(4):343–358.
- Miller, G. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological review*, 63:81–97.
- Miller, P., Brody, C. D., Romo, R., and Wang, X. J. (2003). A Recurrent Network Model of Somatosensory Parametric Working Memory in the Prefrontal Cortex. *Cerebral Cortex*, 13:1208–1218.
- Mink, J. W. (1996). The basal ganglia: Focused selection and inhibition of competing motor programs. *Progress in Neurobiology*, 50:381–425.
- Murdock, B. B. (1983). A distributed memory model for serial-order information. *Psychological review*, 90(4):316–338.
- Murdock, B. B. (1993). Todam2: A model for the storage and retrieval of item, associative and serial-order information. *Psychological review*, 100(2):183–203.
- Murphy, G. L. (2002). *The Big Book of Concepts*. MIT Press, Cambridge MA.
- Nambu, A., Tokuno, H., and Takada, M. (2002). Functional significance of cortico-subthalamo-pallidal 'hyperdirect' pathway. *Neuroscience Research*, 43:111–117.
- Neumann, J. (2001). Holistic Processing of Hierarchical Structures in Connectionist Networks.
- Newell, A. (1990). *Unified theories of cognition*. Harvard University Press, Cambridge, MA.

- Newell, A., Shaw, C., and Simon, H. (1958). Elements of a theory of human problem solving. *Psychological review*, 65:151–166.
- Newell, A. and Simon, H. (1963). *GPS: A Program that simulates human thought*. McGraw-Hill, New York.
- Newell, A. and Simon, H. A. (1976). *GPS, a program that simulates human thought*. Computers and thought. McGraw-Hill, New York, NY.
- Newton, I. (1729). *The mathematical principles of natural philosophy*. Benjamin Motte, Middle-temple-gate.
- Niklasson, L. F. and van Gelder, T. (1994). On being systematically connectionist. *Mind and Language*, 9:288–302.
- Norman, D. A. (1986). *Reflections on Cognition and parallel distributed processing*. MIT Press/Bradford, Cambridge, MA.
- Oaksford, M. and Chater, N. (1994). A rational analysis of the selection task as optimal data selection. *Psychological review*, 101(4):608–631.
- Oaksford, M. and Chater, N. (1996). Rational explanation of the selection task. *Psychological review*, 103(2):381–391.
- Olshausen, B. A., Anderson, C. H., and Essen, D. C. V. (1993). A neurobiological model of visual attention and invariant pattern recognition based on dynamic routing of information. *Journal of Neuroscience*, 13(11):4700–4719.
- Olshausen, B. A. and Field, D. J. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609.
- Oppenheim, P. and Putnam, H. (1958). *Unity of science as a working hypothesis*, pages 3–36. University of Minnesota Press, Minneapolis.
- O'Reilly, R. C. and Frank, M. J. (2006). Making working memory work: a computational model of learning in the prefrontal cortex and basal ganglia. *Neural computation*, 18(2):283–328.
- O'Reilly, R. C., Frank, M. J., Hazy, T. E., and Watz, B. (2007). PVLV: the primary value and learned value Pavlovian learning algorithm. *Behavioral neuroscience*, 121(1):31–49.

- O'Reilly, R. C., Herd, S. A., and Pauli, W. M. (2010). Computational models of cognitive control. *Current opinion in neurobiology*, 20(2):257–61.
- O'Reilly, R. C. and Munakata, Y. (2000). *Computational Explorations in Cognitive Neuroscience: Understanding the Mind by Simulating the Brain*. The MIT Press, 1 edition.
- Osherson, D. N. and Smith, E. E. (1981). On the adequacy of prototype theory as a theory of concepts. *Cognition*, 935– 58.
- Oztop, E., Kawato, M., and Arbib, M. (2006). Mirror neurons and imitation : A computationally guided review. *Neural Networks*, 19:254–271.
- Paaß, G., Kindermann, J., and Leopold, E. (2004). Learning prototype ontologies by hierachical latent semantic analysis. In *Lernen, Wissensentdeckung und Adaptivität*, pages 193–205.
- Page, M. and Norris, D. (1998). The primacy model: A new model of immediate serial recall. *Psychological review*, 105(4):761–781.
- Paivio, A. (1971). *Imagery and verbal processes*. Holt, Rinehart and Winston.
- Paivio, A. (1986). *Mental representations: A dual coding approach*. Oxford University Press, New York.
- Pakkenberg, B. and Gundersen, H. J. (1997). Neocortical neuron number in humans: effect of sex and age. *The Journal of comparative neurology*, 384(2):312–20.
- Parisien, C., Anderson, C. H., and Eliasmith, C. (2008). Solving the problem of negative synaptic weights in cortical models. *Neural Computation*, 20:1473–1494.
- Parisien, C. and Thagard, P. (2008). Robosemantics: How Stanley the Volkswagen represents the world. *Minds and Machines*, 18:169–178.
- Parks, R. W. and Cardoso, J. (1997). Parallel distributed processing and executive functioning: Tower of Hanoi neural network model in healthy controls and left frontal lobe patients. *Int. J. Neurosci*, 89:217.

- Parsons, L. and Osherson, D. (2001). New evidence for distinct right and left brain systems for deductive versus probabilistic reasoning. *Cerebral Cortex*, 11:954–965.
- Parsons, L., Osherson, D., and Martinez, M. (1999). Distinct neural mechanisms for propositional logic and probabilistic reasoning. In *Proceedings of the Psychonomic Society Meeting*, pages 61–62.
- Perfetti, B., Saggino, A., Ferretti, A., Caulo, M., Romani, G. L., and Onofrij, M. (2009). Differential patterns of cortical activation as a function of fluid reasoning complexity. *Human Brain Mapping*, 30(2):497–510.
- Pesaran, B., Pezaris, J. S., Sahani, M., Mitra, P. P., and Andersen, R. A. (2002). Temporal structure in neuronal activity during working memory in macaque parietal cortex. *Nature neuroscience*, 5(8):805–11.
- Peters, A. and Jones, E. G. (1984). *Cerebral Cortex*, volume 1. Plenum Press, New York.
- Petersen, S., Robinson, D., and Keys, W. (1985). Pulvinar nuclei of the behaving rhesus monkey: visual responses and their modulation. *Journal of Neurophysiology*, 54(4):867.
- Petersen, S., Robinson, D., and Morris, J. (1987). Contributions of the pulvinar to visual spatial attention. *Neuropsychologia*, 25:97–105.
- Pew, R. W. and Mavor, A. S., editors (1998). *Modeling human and organizational behavior: Application to military simulations*. National Academy Press, Washington, DC.
- Pfister, J.-P. and Gerstner, W. (2006). Triplets of spikes in a model of spike timing-dependent plasticity. *Journal of neuroscience*, 26(38):9673–82.
- Plate, T. A. (1991). Holographic reduced representations: Convolution algebra for compositional distributed representations. In Mylopoulos, J., editor, *Proceedings of the 12th International Joint Conference on Artificial Intelligence*. Morgan Kaufmann.
- Plate, T. A. (1994). Distributed representations and nested compositional structure.

- Plate, T. A. (2003). *Holographic reduced representations*. CSLI Publication, Stanford, CA.
- Plaut, D. C. and Shallice, T. (1994). *Word Reading in Damaged Connectionist Networks: Computational and Neuropsychological Implications*, pages 294–323. Chapman & Hall, London.
- Poirazi, P., Brannon, T., and Mel, B. W. (2003). Pyramidal Neuron as Two-Layer Neural Network. *Neuron*, 37(6):989–999.
- Pollack, J. (1990). Recursive distributed representations. *Artificial Intelligence*, 46:77–105.
- Pollack, J. B. (1988). Recursive auto-associative memory: devising compositional distributed representations. In *Proceedings of the 10th Annual Conference of the Cognitive Science Society*.
- Polsky, A., Mel, B. W., and Schiller, J. (2004). Computational subunits in thin dendrites of pyramidal cells. *Nature neuroscience*, 7(6):621–7.
- Popper, K. (1959). *The logic of scientific discovery*. Hutchinson, London.
- Port, R. and van Gelder, T. (1995). *Mind as motion: Explorations in the dynamics of cognition*. MIT Press, Cambridge, MA.
- Postman, L. and Phillips, L. (1965). Short-term temporal changes in free recall. *The quarterly journal of experimental psychology*, 17(2):132–138.
- Prabhakaran, V., Smith, J., Desmond, J., Glover, G., and Gabrieli, E. (1997). Neural substrates of fluid reasoning: an fMRI study of neocortical activation during performance of the Raven's Progressive Matrices Test. *Cognitive Psychology*, 33:43–63.
- Prinz, J. (2002). *Furnishing the Mind: Concepts and the Perceptual Basis*. MIT Press.
- Pulvermüller, F. (1999). Words in the brain's language. *Behavioral and Brain Sciences*, 22:253–336.
- Putnam, H. (1975). *Philosophy and our mental life*, chapter 2, pages 291–303. Mind, language and reality: Philosophical papers. Cambridge University Press, Cambridge.

- Quine, W. V. O. and Ullian, J. (1970). *The web of belief*. Random House, New York.
- Quirk, G. J. and Beer, J. S. (2006). Prefrontal involvement in the regulation of emotion: convergence of rat and human studies. *Current opinion in neurobiology*, 16(6):723–7.
- Quiroga, R. Q., Reddy, L., Kreiman, G., Koch, C., and Fried, I. (2005). Invariant visual representation by single neurons in the human brain. *Nature*, 435(7045):1102–7.
- Quyen, M. L. V., Foucher, J., Lachaux, J.-P., Rodriguez, E., Lutz, A., Martinerie, J., and Varela, F. J. (2001). Comparison of Hilbert transform and wavelet methods for the analysis of neuronal synchrony. *Journal of Neuroscience Methods*, 111(2):83–98.
- Ramscar, M. and Gitcho, N. (2007). Developmental change and the nature of learning in childhood. *Trends in cognitive sciences*, 11(7):274–9.
- Rasmussen, D. (2010). *A neural modelling approach to investigating general intelligence*. Masters thesis, University of Waterloo.
- Rasmussen, D. and Eliasmith, C. (2011). A neural model of rule generation in inductive reasoning. *Trends in Cognitive Science*, in press.
- Raven, J. (1962). *Advanced Progressive Matrices (Sets I and II)*. Lewis, London.
- Redgrave, P., Prescott, T., and Gurney, K. (1999). The basal ganglia: a vertebrate solution to the selection problem? *Neuroscience*, 86:353–387.
- Regan, J. K. O. and Noë, A. (2001). A sensorimotor account of vision and visual consciousness. *Behavioral and Brain Sciences*, 24(5):939–1031.
- Reitman, J. (1974). Without surreptitious rehearsal, information in short-term memory decay. *Journal of Verbal Learning and Verbal Behavior*, 13:365–377.
- Reynolds, J. H., Chelazzi, L., and Desimone, R. (1999). Competitive mechanisms sub-serve attention in macaque areas V2 and V4. *Journal of Neuroscience*, 19:1736–1753.
- Rieke, F., Warland, D., de Ruyter van Steveninck, R., and Bialek, W. (1997). *Spikes: Exploring the neural code*. MIT Press, Cambridge, MA.

- Rinella, K., Bringsjord, S., and Yang, Y. (2001). *Efficacious logic instruction: People are not irremediably poor deductive reasoners*, pages 851–856. Proceedings of the 23rd Annual Conference of the Cognitive Science Society. Lawrence Erlbaum Associates, Mahwah, NJ.
- Ringach, D. L. (2002). Spatial Structure and Symmetry of Simple-Cell Receptive Fields in Macaque Primary Visual Cortex. *J Neurophysiol*, 88(1):455–463.
- Ritter, D. A., Bhatt, D. H., and Fethko, J. R. (2001). In Vivo Imaging of Zebrafish Reveals Differences in the Spinal Networks for Escape and Swimming Movements. *Journal of Neuroscience*, 21(22):8956–8965.
- Roberts, S. and Pashler, H. (2000). How persuasive is a good fit? A comment on theory testing. *Psychological Review*, 107(2):358–367.
- Rogers, T. T. and McClelland, J. L. (2004). *Semantic cognition: a parallel distributed processing approach*. MIT Press.
- Romo, R., Brody, C. D., Hernández, A., and Lemus, L. (1999a). Neuronal correlates of parametric working memory in the prefrontal cortex. *Nature*, 399:470–473.
- Romo, R., Brody, C. D., Hernandez, A., and Lemus, L. (1999b). Neuronal correlates of parametric working memory in the prefrontal cortex. *Nature*, 399:470–473.
- Rosenblueth, A., Wiener, N., and Bigelow, J. (1943). Behavior, purpose, and teleology. *Philosophy of Science*, 10:18–24.
- Rumelhart, D. E. and McClelland, J. L. (1986). *Parallel distributed processing: Explorations in the microstructure of cognition*. Number 1. MIT Press/Bradford Books, Cambridge MA.
- Rundus, D. (1971). Analysis of rehearsal processes in free recall. *Journal of experimental psychology*, 89(1):63–77.
- Ryan, L. and Clark, K. (1991). The role of the subthalamic nucleous in the response of globus pallidus neurons to stimulation of the prelimbic and agranular frontal cortices in rats. *Experimental Brain Research*, 86:641–651.
- Salinas, E. and Abbott, L. F. (1997). Invariant Visual Responses From Attentional Gain Fields. *J Neurophysiol*, 77(6):3267–3272.

- Salvucci, D. (2006). Modeling driver behavior in a cognitive architecture. *Human Factors*, 48:368–380.
- Scheier, C. and Pfeifer, R. (1995). Classification as sensory-motor coordination.
- Schiller, J., Major, G., Koester, H. J., and Schiller, Y. (2000). NMDA spikes in basal dendrites of cortical pyramidal neurons. *Nature*, 404(6775):285–9.
- Schöner, G. (2008). Dynamical systems approaches to cognition. *Cambridge Handbook of Computational Cognitive Modeling*, pages 101–126.
- Schöner, G. and Thelen, E. (2006). Using dynamic field theory to rethink infant habituation. *Psychological Review*, 113:273–299.
- Schultheis, H. (2009). Computational and Explanatory Power of Cognitive Architectures: The Case of ACT-R. In Howes, A., Peebles, D., and Cooper, R. P., editors, *International Conference on Cognitive Modeling*, pages 384–389.
- Schultz, W., Dayan, P., and Montague, P. R. (1997). A neural substrate of prediction and reward. *Science*, 275:1593–1599.
- Schutte, A. R., Spencer, J. P., and Schöner, G. (2003). Testing the dynamic field theory: Working memory for locations becomes more spatially precise over development. *Child Development*, (74):1393–1417.
- Searle, J. (1984). *Minds, Brains and Science: The 1984 Reith Lectures*. Harvard University Press, Cambridge, MA.
- Shadlen, M. N. and Movshon, J. A. (1999). Synchrony Unbound: A Critical Evaluation of the Temporal Binding Hypothesis. *Neuron*, 24(1):67–77.
- Shastri, L. and Ajjanagadde, V. (1993). From simple associations to systematic reasoning: A connectionist representation of rules, variables, and dynamic bindings. *Behavioral and Brain Sciences*, 16:417–494.
- Silver, R., Boahen, K., Grillner, S., Kopell, N., and Olsen, K. L. (2007). Neurotech for neuroscience: Unifying concepts, organizing principles, and emerging tools. *Journal of Neuroscience*, 27(44):807–819.
- Simmons, W. K., Hamann, S. B., Harenki, C. L., Hu, X. P., and Barsalou, L. W. (2008). fMRI evidence for word association and situated simulation in conceptual processing. *Journal of physiology, Paris*, 102:106–119.

- Simon, H. A. (1975). The functional equivalence of problem solving skills. *Cognitive Psychology*, 7(2):268–288.
- Singh, R. and Eliasmith, C. (2006). Higher-dimensional neurons explain the tuning and dynamics of working memory cells. *Journal of Neuroscience*, 26:3667–3678.
- Smith, E. E. (1989). *Concepts and induction*, pages 501–526. Foundations of cognitive science. MIT Press, Cambridge, MA.
- Smolensky, P. (1988). On the proper treatment of connectionism. *Behavioral and Brain Sciences*, 11(1):1–23.
- Smolensky, P. (1990). Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, 46:159–217.
- Smolensky, P. and Legendre, G. (2006a). *The Harmonic Mind: From Neural Computation to Optimality-Theoretic Grammar Volume 1: Cognitive Architecture*. MIT Press, Cambridge MA.
- Smolensky, P. and Legendre, G. (2006b). *The Harmonic Mind: From Neural Computation to Optimality-Theoretic Grammar Volume 2: Linguistic and Philosophical Implication*. MIT Press, Cambridge MA.
- Solomon, K. O. and Barsalou, L. W. (2004). Perceptual simulation in property verification. *Memory and Cognition*, 32:244–259.
- Song, S., Sjostrom, P. J., Reigl, M., Nelson, S., and Chklovskii, D. B. (2005). Highly nonrandom features of synaptic connectivity in local cortical circuits. *PLoS biology*, 3(3):e68.
- Sperber, D., Cara, E., and Girotto, R. (1995). Relevance theory explains the selection task. *Cognition*, 57:31–95.
- Spruston, N., Jonas, P., and Sakmann, B. (1995). Dendritic glutamate receptor channel in rat hippocampal CA3 and CA1 pyramidal neurons. *Journal of Physiology*, 482:325–352.
- Squire, L. (1992). Memory and the hippocampus: a synthesis from findings with rats, monkeys, and humans. *Psychological review*, 99:195–231.

- Stepniewska, I. (2004). *The pulvinar complex*. CRC Press LLC, Cambridge, MA.
- Stevens, C. F. and Wang, Y. (1994). Changes in reliability of synaptic function as a mechanism for plasticity. *Nature*, 371:704–707.
- Stewart, T., Choo, X., and Eliasmith, C. (2010a). Dynamic Behaviour of a Spiking Model of Action Selection in the Basal Ganglia. In Salvucci, D. D. and Ganzelmann, G., editors, *10th International Conference on Cognitive Modeling*.
- Stewart, T. and Eliasmith, C. (2011). Neural Cognitive Modelling: A Biologically Constrained Spiking Neuron Model of the Tower of Hanoi Task. In Carlson, L., Hölscher, C., and Shipley, T., editors, *Proceedings of the 33rd Annual Conference of the Cognitive Science Society*, Austin, TX. Cognitive Science Society.
- Stewart, T., Tang, Y., and Eliasmith, C. (2010b). A biologically realistic cleanup memory: Autoassociation in spiking neurons. *Cognitive Systems Research*.
- Stewart, T. C., Bekolay, T., and Eliasmith, C. (2011). Neural representations of compositional structures: Representing and manipulating vector spaces with spiking neurons. *Connection Science*.
- Sun, R. (2006). *The CLARION cognitive architecture: Extending cognitive modeling to social simulation*. Cambridge University Press.
- Taatgen, N. and Anderson, J. (2010). The past, present, and future of cognitive architectures. *Topics In Cognitive Science*, 2:693–704.
- Taatgen, N. and Anderson, J. R. (2002). Why do children learn to say "broke"? A model of learning the past tense without feedback. *Cognition*, 86:123–155.
- Tanaka, K. (1993). Neuronal mechanisms of object recognition. *Science*, 262(5134):685–688.
- Tang, Y. and Eliasmith, C. (2010). Deep networks for robust visual recognition. In Fürnkranz, J. and Joachims, T., editors, *Proceedings of the 27th International Conference on Machine Learning*.
- Thagard, P. (2011). *Cognitive architectures*. Cambridge University Press, Cambridge.
- Thagard, P. (2012). *The cognitive science of science: Explanation, discovery, and conceptual change*. MIT Press, Cambridge, MA.

- Thagard, P. and Litt, A. (2008). *Models of scientific explanation*, pages 549–564. Cambridge University Press, Cambridge.
- Thelen, E., Schöner, G., Scheier, C., and Smith, L. (2001). The dynamics of embodiment: A field theory of infant perseverative reaching. *Brain and Behavioral Sciences*, (24):1–33.
- Todorov, E. (2007). *Optimal control theory*.
- Todorov, E. (2009). *Parallels between sensory and motor information processing*. MIT Press.
- Tovée, M. and Rolls, E. (1992). The functional nature of neuronal oscillations. *Trends in Neuroscience*, 15:387.
- Treue, S. and Martinez-Trujillo, J. (1999). Feature-based attention influences motion processing gain in macaque visual cortex. *Nature*, 399:575–579.
- Tripp, B. P. and Eliasmith, C. (2007). Neural populations can induce reliable postsynaptic currents without observable spike rate changes or precise spike timing. *Cerebral cortex (New York, N.Y. : 1991)*, 17(8):1830–40.
- Tripp, B. P. and Eliasmith, C. (2010). Population models of temporal differentiation. *Neural computation*, 22(3):621–59.
- Turner, R. S. and Desmurget, M. (2010). Basal ganglia contributions to motor control: a vigorous tutor. *Current opinion in neurobiology*, 20(6):704–16.
- Turrigiano, G. G. and Nelson, S. B. (2004). Homeostatic plasticity in the developing nervous system. *Nature reviews. Neuroscience*, 5(2):97–107.
- Ulanovsky, N., Las, L., Farkas, D., and Nelken, I. (2004). Multiple time scales of adaptation in auditory cortex neurons. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 24(46):10440–53.
- van der Velde, F. and de Kamps, M. (2006). Neural blackboard architectures of combinatorial structures in cognition. *Behavioral and Brain Sciences*, (29):37–108.
- Van Essen, D., Olshausen, B., Anderson, C., and Gallant, J. (1991). Pattern recognition, attention, and information bottlenecks in the primate visual system. In *Proceedings of SPIE Conf. on Visual Information Processing: From Neurons to Chips*, volume 1473, page 17.

- van Gelder, T. (1995). What might cognition be if not computation? *Journal of Philosophy*, 92(7).
- van Gelder, T. (1998). The dynamical hypothesis in cognitive science. *Behavioral and Brain Sciences*, 21(5):615–665.
- van Gelder, T. and Port, R. (1995). *It's about time: An overview of the dynamical approach to cognition*. Mind as motion: Explorations in the dynamics of cognition. MIT Press, Cambridge, MA.
- Varela, J. A., Sen, K., Gibson, J., Fost, J., Abbott, L. F., and Nelson, S. B. (1997). A Quantitative Description of Short-Term Plasticity at Excitatory Synapses in Layer 2/3 of Rat Primary Visual Cortex. *J. Neurosci.*, 17(20):7926–7940.
- Verguts, T. and De Boeck, P. (2002). The induction of solution rules in Ravens Progressive Matrices Test. *European Journal of Cognitive Psychology*, 14:521–547.
- Vigneau, F., Caissie, A., and Bors, D. (2006). Eye-movement analysis demonstrates strategic influences on intelligence. *Intelligence*, 34(3):261–272.
- von der Malsburg, C. (1981). The correlation theory of brain function.
- Warden, M. R. and Miller, E. K. (2007). The representation of multiple objects in prefrontal neuronal delay activity. *Cerebral cortex (New York, N.Y. : 1991)*, 17 Suppl 1(suppl_1):i41–50.
- Wason, P. C. (1966). *Reasoning*. New horizons in psychology. Penguin, Harmondsworth.
- Weiskopf, D. A. (2010). Embodied cognition and linguistic comprehension. *Studies in History and Philosophy of Science*, 41:294–304.
- West, L. J., Pierce, C. M., and Thomas, W. (1962). Lysergic Acid Diethylamides: Its effects on a male Asiatic elephant. *Science*, 138:1100–1103.
- Whitlock, J. R., Heynen, A. J., Shuler, M. G., and Bear, M. F. (2006). Learning induces long-term potentiation in the hippocampus. *Science (New York, N.Y.)*, 313(5790):1093–7.
- Wilson, J. *Metaphysical Emergence : Weak and Strong*. Stephen mu edition.

- Wolfrum, P. and von der Malsburg, C. (2007). What is the optimal architecture for visual information routing? *Neural computation*, 19(12):3293–309.
- Wolpert, D. M. and Kawato, M. (1998). Multiple paired forward and inverse models for motor control. *Neural Networks*, 11:1317–1329.
- Womelsdorf, T., Anton-Erxleben, K., Pieper, F., and Treue, S. (2006). Dynamic shifts of visual receptive fields in cortical area MT by spatial attention. *Nature neuroscience*, 9(9):1156–60.
- Womelsdorf, T., Anton-Erxleben, K., and Treue, S. (2008). Receptive field shift and shrinkage in macaque middle temporal area through attentional gain modulation. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 28(36):8934–44.
- Yang, Y. and Liu, X. (1999). A re-examination of text categorization methods. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '99*, pages 42–49, New York, New York, USA. ACM Press.
- Zipser, D., Kehoe, B., Littlewort, G., and Fuster, J. (1993). A spiking network model of short-term active memory. *J. Neurosci.*, 13(8):3406–3420.
- Zucker, R. S. (1973). Changes in the statistics of transmitter release during facilitation. *J. Physiol.*, 229(3):787–810.