ELSEVIER

Cognitive Brain Research 15 (2002) 71–83

BRAIN RESEARCH INTERACTIVE

www.bres-interactive.com

Interactive report

A computational approach to control in complex cognition

Thad A. Polk*, Patrick Simen, Richard L. Lewis, Eric Freedman

8 September 2002

Department of Psychology, University of Michigan, 525 E. University, Ann Arbor, MI 48109-1109, USA

Abstract

Cognitive deficits associated with dorsolateral prefrontal cortex (DLPFC) damage are often most apparent in higher cognitive tasks that involve problem solving and managing multiple goals. However, computational models of prefrontal deficits on such tasks are difficult to construct. Problem solving is most naturally modeled with symbolic systems (e.g. production systems), but the effects of lesions are most naturally modeled with subsymbolic systems (neural networks). We show that when we adopt a simple and plausible model of neural computation, there is a natural and explicit mapping from symbolic, goal-driven cognition onto neural computation. We exploit this mapping to construct a neural network model that is capable of solving complex problems in the Tower of London task. The model leads to a specific hypothesis about the role of DLPFC in such tasks, namely, that DLPFC represents internally generated subgoals that modulate competition among posterior representations. When intact, the model accurately simulates the behavior of college students even on the most difficult problems. Furthermore, when the subgoal component is lesioned, it accurately simulates the behavior of prefrontal patients, including the fact that their deficits are most apparent on the most difficult tasks and that they have special difficulty with tasks that require inhibiting a prepotent response.

1. Introduction

To date, most neurally inspired models of executive control have addressed relatively simple tasks that can be performed in about a second (e.g. Stroop, continuous performance) [7,10,11]. However, patients with executive deficits are often most impaired on complex cognitive tasks that involve planning or problem solving. Therefore, an important challenge for cognitive neuroscience is to develop computationally explicit theories of the role of executive control in complex cognition. In this paper, we take a step in that direction by proposing a neural network model of executive control in the Tower of London problem-solving task.

The most common approach to developing computational models of complex cognition is to use symbolic (and hybrid symbolic) production systems (e.g. ACT-R, Soar, EPIC) [1,28,34]. These systems are computationally powerful, programmable, and embody independently motivated theoretical assumptions about cognitive architecture—all virtues that make them ideal for constructing models of complex behavior. On the other hand, they are not based on neuron-like processing units and are therefore harder to map onto the brain than are neural networks. In particular, they provide a less natural means for modeling the effects of brain damage (but see Refs. [20,26]). Neural networks are easier to map onto the brain and are simple to damage, but they are harder to apply to complex, sequential behavior like problem solving (but see Refs. [4,14]).

In this paper, we make three major points. Firstly, many (but not all) aspects of goal-driven production systems map quite naturally onto neural networks. Secondly, given this mapping, it is possible to build plausible neural network models of complex problem solving. In particular, applying the mapping to the Tower of London task leads to a model that accurately simulates the behavior of intact adults as well as the behavior of patients with prefrontal damage. Thirdly, the resulting Tower of London model leads to an explicit hypothesis about the role of dorsolateral prefrontal cortex in problem solving, namely, that it represents internally generated subgoals that modulate among choices.

We begin by describing some key features of goal-driven production systems. Then we describe a simple model of neural computation and argue that many aspects of production systems can be mapped very naturally onto

*Corresponding author. Tel.: +1-734-647-6982; fax: +1-734-763-7480.

E-mail address: tpolk@umich.edu (T.A. Polk).

0926-6410/02/$ – see front matter
PII: S0926-6410(02)00217-3

neural computation. Finally, we use this mapping to develop a neural network model of the Tower of London task and show that it can model both intact and damaged behavior.

2. Goal-driven production systems

Most computational models of complex cognition are based on production systems. Production systems use symbolic IF–THEN rules, which are asymmetric associations between patterns of symbols and which match against memory and take actions and/or change memory. Fig. 1 provides an example of a production rule. The condition side (above the THEN) specifies the conditions under which the rule will fire. If all of the elements specified on the condition side are satisfied, then the production will fire (i.e. its actions will be taken). The action side (below the THEN) specifies the actions that the production rule will take when its conditions are satisfied, in this case, adding a new element to working memory. Changes to working memory may then lead other productions to match and fire leading to further changes in working memory.

The symbolic representations in production systems commonly take the form of attribute–value pairs. For example, there are three attribute–value pairs in the production rule in Fig. 1. The condition side tests that the attribute Letter1 has the value A and that the attribute Letter2 has the value T. If both these conditions are satisfied, then the action side assigns the value AT to the attribute Word.[1]

Using production systems, it is possible to build models that behave flexibly and opportunistically in response to changes in the environment. In particular, they provide a natural model of data-driven, interruptible control. Unlike traditional programming languages in which top-down control is passed explicitly from one operation to the next (typically to the next line in the program or to a subroutine), individual production rules fire opportunistically whenever their conditions are satisfied. The flow of control is not laid out in advance but is determined at run-time as a function of the dynamically evolving contents of working memory. In fact, production systems were originally proposed as a psychological theory of control [32] in response to the lack of explicit models of control in cognitive psychology, and in response to the limitations of existing control structures found in computer science at the time. This was an important step toward eliminating the homunculus from cognitive psychology by creating the possibility of computationally complete models of cognition in which memory, process, and control are all specified as part of the theory.

```
IF
   Letter1 = A
   AND
   Letter2 = T
THEN
   Word = AT
```

Fig. 1. An example of a simple production with two conditions and an action that changes the value of the attribute Word.

[1]This description of production systems glosses over many important details, including variables, variable binding, conflict resolution, and working memory element identifiers.

In addition to being able to respond flexibly to changes in the environment (data-driven control), most production system models include a mechanism for top-down control via goals and subgoals. In both ACT-R and Soar, for example, most productions include a condition that tests for a specific goal/subgoal so that only productions that are relevant to the current goal/subgoal are considered. This top-down control can come either from externally provided goals (e.g. instructions) or from subgoals that were generated internally by the system itself (e.g. removing an obstacle that is impeding progress toward an externally provided goal). There are significant theoretical differences between the control structures of ACT-R, EPIC, and Soar which have important implications for neural models of control; for our present purposes, however, we will focus on the common functionality of supporting both goal-modulated and data-driven behavior.

2.1. An example from the Tower of London task

Fig. 2 illustrates the interaction of data-driven control and top-down, goal-driven control in a Tower of London (TOL) task. The TOL task involves moving three colored balls in an initial configuration until they match a given goal configuration [44]. Unlike the Tower of Hanoi task, there are no constraints specifying which balls can be placed on which others, but the pegs differ in how many balls they can hold at one time (one peg can hold three balls, another can hold two, and the other can hold only one). Participants are often asked to try to figure out how to achieve the goal in the minimum number of moves and are sometimes asked to plan out the entire sequence of moves before they begin [36,44].

A typical production system model for this task would include production rules that propose legal moves. For example, starting from the initial configuration in Fig. 2, two production rules might match, one for each of the two legal moves (moving the blue ball to the long peg and moving the red ball to the long peg). A purely data-driven production system without goal-driven control might use a fixed conflict resolution strategy to choose between these options, or simply choose randomly. More typically, a goal or subgoal helps to resolve conflicts like these. For example, the goal configuration in Fig. 2 specifies three goals to be achieved (getting each of the three balls into

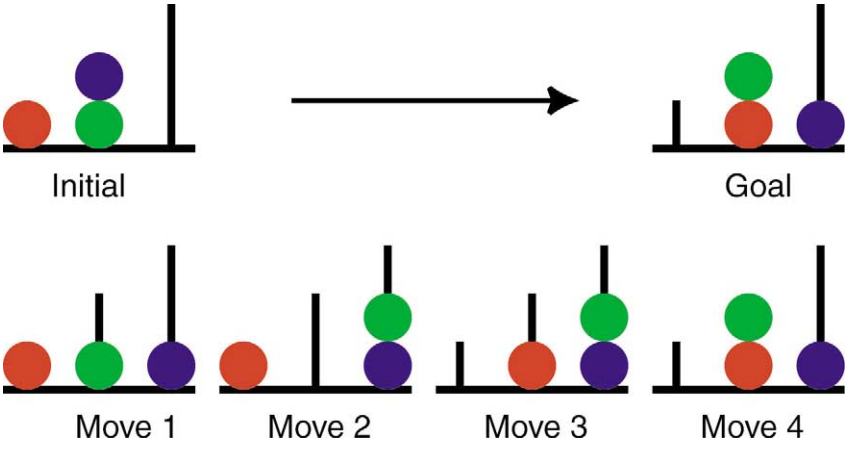

Fig. 2. A Tower of London problem defined by its initial and goal states, and a sequence of moves leading to a solution.

their goal position). Such goals can provide top-down control that leads the system to prefer the goal-achieving move (the blue ball) over the other legal move.

In this case, the top-down control was based on an externally provided goal, namely, the explicit goal configuration in the problem statement. Internally generated subgoals can also provide top-down control. For example, after moving the blue ball to the long peg, the red ball needs to move into the green ball's location. But in order to achieve this (external) goal, the green ball has to be moved first. Of the four legal moves (moving red to the middle or long pegs, moving green to the long peg, moving blue back to the middle peg), only the green move achieves this (internally-generated) subgoal and so it would be preferred.

To summarize, modern production system architectures combine flexible, data-driven control with the ability to exert top-down control as well. Flexible, data-driven control arises from the use of asymmetric associations between patterns of symbols (production rules) which can assume control whenever their conditions are satisfied. Top-down control is provided by goals and subgoals that lead the system to prefer goal-achieving actions to other possible actions. Such top-down control can arise from both externally specified goals as well as internally generated subgoals. The combination of data-driven and goal-driven control is one of the features that make production systems such a natural choice for modeling complex cognitive behavior.

3. A simple model of neural computation

Having just described some of the major features of symbolic models of control, we now describe a simple model of neural computation and argue that there is a natural mapping between the two. This model has proved to provide leverage in explaining a variety of cognitive functions [5,13,16,22,25,31,40–42,47] and is based on three simple and widely accepted assumptions about neural computation. (1) Representations in cortex are generally distributed across a population of neurons, rather than being localized to individual cells. (2) There is massive connectivity among neurons within local areas of cortex and this connectivity is recurrent rather than unidirectional. (3) Synaptic efficiency is modified based on the correlation between pre- and post-synaptic activity (correlation-based Hebbian learning).

Although these assumptions abstract away from a host of details about neural computation, nevertheless they are sufficient to give rise to some important emergent properties. For our purposes, the most important of these is that some distributed patterns of activation constitute discrete, stable states to which the system will naturally converge [23,24]. These activation patterns are termed attractors and the networks themselves are known as attractor networks (or Hopfield networks).

Fig. 3 illustrates the behavior of an attractor network. Assume that this network is frequently presented with input that leads to activation of the red distributed pattern (assumption 1). Assuming massive, recurrent connectivity (assumption 2), many of the units involved in the red pattern will be interconnected. Furthermore, under Hebbian learning (assumption 3), each time this activation pattern occurs the connections between these units will be strengthened. If this pattern occurs frequently enough, the strengthening of these connections will lead the red pattern to become a discrete, stable state for the system (i.e. an attractor). For example, suppose that after frequent exposure to the red pattern, the network is presented with input that activates all the units in the red pattern except for one. All the other units in the pattern that are interconnected with that unit will have developed strong excitatory connections to that unit and hence they will immediately activate it. More generally, given any input pattern, the network will converge on the discrete attractor pattern that is most similar to that input pattern. Furthermore, once the network has converged on an attractor pattern, the excitat-

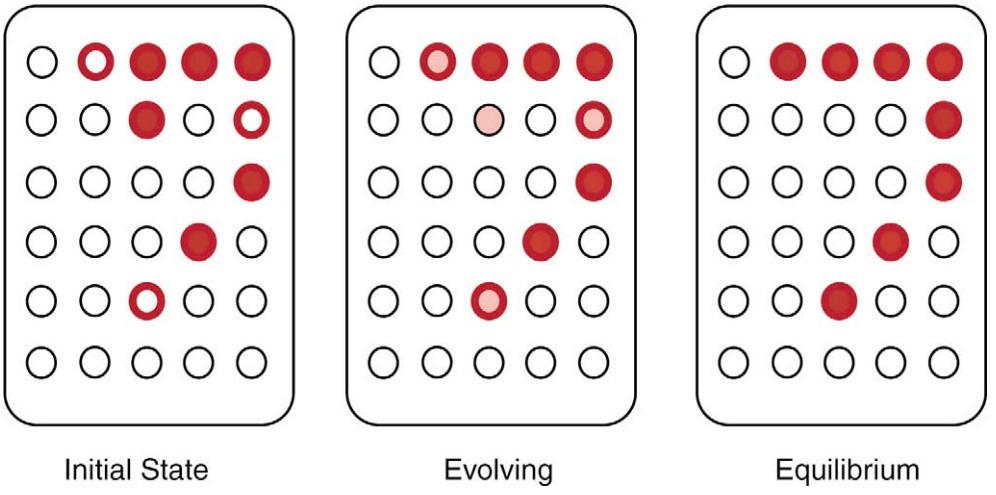

Fig. 3. Convergence of a network to the attractor pattern shown on the right after initialization to the pattern shown on the left. Units that are inactive in the attractor pattern gradually shut off if they are active, and units that are active in the attractor pattern gradually turn on if they are inactive (these units are shown with a red outline in the first time slice).

ory connections among the units themselves will lead the activation to reverberate so that the pattern can be maintained even after the input has been removed.[2]

4. Mapping goal-driven production systems onto neural networks

We now describe a mapping from some of the key features of goal-driven production systems onto attractor networks. This mapping supports three important features: (1) symbolic representations (in particular, patterns of attribute–value pairs), (2) asymmetric associations between patterns of symbols, and (3) goal-driven control of behavior.

4.1. Attractor network realization of symbolic attribute–value representations

Production systems typically use some form of attribute–value representation for the contents of working memory and the condition and action patterns in the productions. Attribute–value pairs are a convenient formalism that supports the representation of features and relations. For example, the representation of a red square might include the features ⟨color red⟩ and ⟨shape square⟩. Although the attribute–value scheme is a relatively theoretically neutral medium (e.g. there is no commitment at this level to any ontology of relations or features), it does support combinatorial variety of representation, a functional requirement in complex cognition [34].

Inherent in symbolic computational models are three important computational properties that help support this representational capacity. These properties can be taken for granted in symbolic models [27,33,45], but may not be transparent in neural network models and so it is important to make them explicit. Firstly, the combinatorial power of attribute–value representations depends directly on the separation of attributes and values—a kind of modularity built into the representational format. This separation is what permits novel combinations of attributes and values to be entertained. Secondly, attribute–value representations are effectively equivalence classes that map aspects of an arbitrarily rich and continuous world into categories relevant to current goals. Thirdly, attribute–value representations are stable patterns. They are robust against functionally irrelevant perturbations of the external (and internal) environment, and persist for long enough stretches of time to influence behavior appropriately.

All three of these properties of attribute–value representations are naturally supported to some degree by attractor networks. The separation of attributes and values can be mapped directly onto the distinction between attractor networks and attractor patterns. For example, the *size* attribute might correspond to a single attractor network in which the values (e.g. *small*, *large*) correspond to particular attractor states in that network. The attractor states are discrete equivalence classes because they represent points within an infinite representational space to which the network gravitates. The gravitation or settling from some point in the space to the attractor constitutes the process of mapping points to their equivalence class. Attractors are stable representations because attractors are

[2]This description of attractor networks also glosses over a number of important details, including the interaction among different attractor patterns in the same network, how input can change the energy landscape of the network and thereby change the stable states, and how different patterns can come to compete with each other (e.g. by including anti-Hebbian learning or assuming that most units inhibit each other until positive Hebbian learning overcomes that default inhibition). For a more detailed discussion of attractor networks, we refer the interested reader to Hertz et al. [21].

stable points in the dynamical system. These properties of attractor networks contribute to their widespread use as pattern classifier systems and associative memories.

As in a production system, the semantics of the attractors (what the networks and attractors denote) is not inherent in the form of the patterns or networks, but is a function of the associations of those patterns with other patterns in the system, ultimately grounded in perceptual-motor systems. These associations, discussed next, are the second key component of the mapping.

4.2. Attractor network realization of production rules

Productions are asymmetric associations between patterns of symbols (patterns of attribute–value pairs). The natural mapping to attractor networks is that productions correspond to asymmetric associations between sets of attractor patterns. For example, Fig. 4 shows two production rules: *If Letter*1 = *A then Response is* '*A*', and *If Letter*2 = *T then Response is* '*T*'. The first production rule is implemented by an association between the attractor pattern representing the attribute–value pair ⟨Letter1 A⟩, and the attractor pattern representing attribute–value pair ⟨Response 'A'⟩. The association consists of excitatory connections from one attractor pattern to another, such that when the A pattern becomes active in the letter1 network, the 'A' pattern in the response network is also activated. The second production rule is implemented in a similar fashion. Such associations provide the fundamental mechanism of *data-driven* or bottom-up processing inherent in production systems. At this level, no control is exerted; if the condition part of a production rule is satisfied (i.e. if the appropriate attractor patterns are activated), then the production will fire and activate its action pattern.

The productions in Fig. 4 contain only a single condition, but of course in general productions may specify a conjunction of multiple conditions. The functionality of multiple conditions is approximated in our mapping onto attractor networks in the following way. Each of the conditions is encoded by a separate association from the appropriate attribute pattern to the action pattern(s), just as for productions with single conditions. The difference is that the strengths of each of these associations are reduced so that the network representing the action side of the production does not settle into its attractor pattern unless all of the condition associations are providing input. This is an approximate mapping because networks constructed in this way are not guaranteed to behave exactly like a production system when the productions have overlapping sets of conditions. For the present task, however, the functionality is sufficient. (Multiple actions on the right-hand side of a production can be easily handled by treating each action as a separate production.)

These associations between attractor networks add a new source of constant input to the networks, thereby changing the dynamics of the attractors. One way to think about what is happening is that the new continuous source of input reshapes the attractor landscape so that it is strongly biased in favor of the target pattern of the production rule. The amount of bias, or the size of the

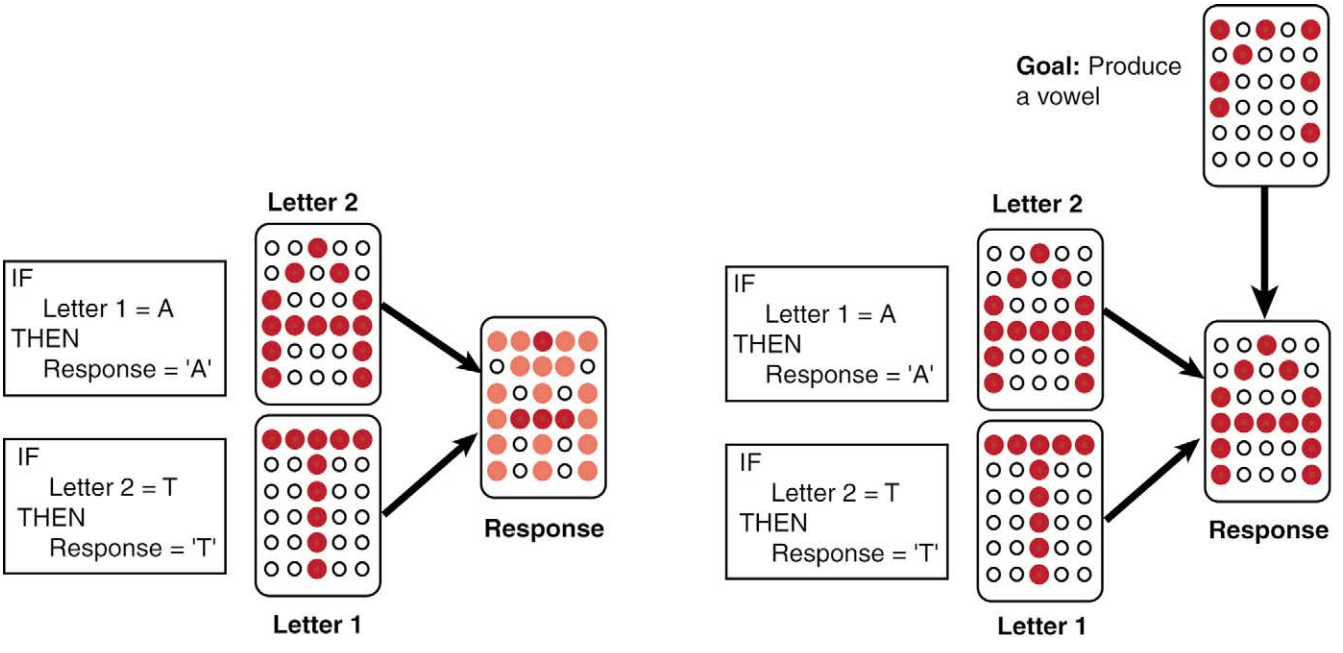

Fig. 4. Encoding of two productions which lead to conflicting responses on the left, and goal-driven selection of a unique response on the right. In the situation on the left, the response network would ultimately settle on the representation of either A or T with equal probability. On the right however, A is favored over T by excitation from the goal network, which represents a preference for a vowel response rather than a consonant response.

attractor basins, is a function of the strength of the associations between the condition side and action side attractor patterns.

4.3. Goal-driven control of behavior

In production systems, goals are explicit representations that bias behavior toward actions that the agent believes to be on the path toward achieving the goals. There are a variety of mechanisms in current production system architectures that accomplish this, but there are at least three critical functions that must be supported in all goal-driven systems. (1) There must be some way to encode arbitrary associations between goal representations and actions. These associations represent task-specific heuristic knowledge about how to accomplish the task. (2) There must be some way to encode arbitrary associations between the current state and appropriate goals/subgoals to pursue. (3) There must be some mechanism that permits the knowledge about candidate actions and their desirability with respect to current goals to be brought together dynamically to exert a choice over what to do next (whether an overt external action or internal cognitive step).

All three of these functions of goal-driven control are directly supported by aspects of the mapping we have just provided from productions to attractor networks. The associations between goals and candidate actions are encoded as excitatory or inhibitory connections between attractors representing active goals and attractors representing possible actions to take. Similarly, the associations between the current state and (sub)goals are encoded as excitatory or inhibitory connections between attractors representing aspects of the state and appropriate (sub)goals. Finally, attractor dynamics provides the mechanism of choice: associations from goals to actions bias attractor competition toward those actions that the agent's knowledge indicates are best given the current goal, and the choice is effected when the attractor settles into the nearest stable pattern given all the input.

A simple example will help illustrate. Suppose that two different production rules have their conditions satisfied and are competing to activate different patterns of an attractor network. Fig. 4, left, shows two productions, one activating response 'A' and another activating response 'T'. In the absence of other input, both response patterns will be activated, and the response network will eventually settle into one of the attractors ('A' or 'T') depending on the strengths of the associations, the relative strengths of the attractor patterns themselves, and possible noise in the system.

Fig. 4, right, shows the same two production rules with an additional goal representation biasing the competition in favor of vowel responses. The basic mechanism is the same as any production rule in the system: the goal→action association effectively changes the attractor landscape so that larger basins correspond to goal-favored actions. It is important to note that the goal associations can be both excitatory and inhibitory, and can bias the competition toward (or away from) *sets* of action patterns, not just single actions. The basic attractor dynamics then (typically) push the system into a single stable pattern that represents the chosen action. The outcome of the competition will of course depend on the strengths of the attractors and the strengths of both the data-driven and goal-driven associations.

This mapping is sufficiently well-specified that we have implemented it as an explicit production rule compiler. The compiler takes as input production rules written in a restricted language and produces as output Matlab code that implements attractor networks that behave like the input production rules. There are still several important computational features of symbolic production system architectures that are not fully captured in this mapping. Most prominent among these are variable binding, robust sequencing, structured representations, and, as noted above, the full functionality of multiple conditions. We have two reactions to these deficiencies. On the one hand, neural networks certainly lack important functionality that symbolic models provide and addressing these weaknesses is an important topic for future research. On the other hand, to the extent that our mapping is appropriate, it suggests that standard production systems may also be too powerful as models of the neural computations that support cognition. For example, the attractor networks only approximate conjunctive conditions because associations among common attractors interact—they are not completely independent and modular as they are in a symbolic system. We suspect that this may be a case where the attractor network model characterization of computational power is closer to reality than the symbolic model. In any case, all of these functions represent important and difficult research topics. For present purposes, we simply note that the system is powerful enough to model aspects of cognitive tasks that are more complex than is typically attempted with neural networks.[3]

[3]We should also note that this composition of more complex neural networks from multiple attractor networks does not yield systems that are mathematically guaranteed to behave in certain ways—unlike single attractor networks, whose behavior is well understood and analytically tractable. In particular, while the presence of continuous external input from other networks can be usefully thought of as reshaping the attractor landscape, the dynamic nature of that input means that the network no longer satisfies the formal definition of an attractor network. It is therefore crucial to empirically demonstrate that models developed with this mapping do in fact behave properly. Once basic functionality is established for a particular task, we can explore whether the model provides an adequate account of the behavior of normal subjects when supplied with the task heuristics that subjects are assumed to have. Once the empirical adequacy of the model is established for normal subjects, we can also explore how well lesioned versions of the model provide an account of prefrontal patients, and use the models to develop explicit hypotheses about cortical function. The next section describes one such exploration of an attractor network model of the Tower of London task.

5. An attractor network model of Tower of London

As previously discussed, a central challenge for cognitive neuroscience is to develop computationally explicit and neurally inspired models of executive control in complex cognitive tasks. Models of complex tasks can be constructed in goal-driven production systems, but mapping such models onto the brain (and especially onto brain damage) is less straightforward. Conversely, neural networks are a natural approach to developing neurally inspired models of intact and damaged behavior, but they are difficult to apply to complex cognition. We have exploited the mapping (and compiler) just described to construct a neural network model of the Tower of London task and have used it to explore potential neural mechanisms of executive control in complex problem solving.

We began by constructing a production system model of the task. Doing so served two purposes. Firstly, production systems are a convenient formalism in which to specify assumptions about the representations and knowledge used in the task and to verify their sufficiency for actually performing the task. Secondly, once we had built a production system model of the task, we could exploit the mapping we had developed to build a neural network model of the task with which we could explore the neural mechanisms of control and the effects of damage to specific parts of the network.

At the very least, any model of Tower of London must be able to represent the current state (where the balls are, which balls are moveable/clear and which are blocked), the goal (where the balls need to go), and potential moves. Also, as previously discussed, any plausible model must almost certainly be able to represent internally generated subgoals, such as the subgoal to remove a ball that is blocking a goal-achieving move. Table 1 shows the set of attributes and values that we adopted to represent each of these four kinds of information as well as the types of knowledge (in the form of production rules) that we included.

A single attribute was used to represent moves and its values corresponded to legal moves (e.g. move red to position6, move green to position2 . . .). Similarly, we used a single attribute to represent the goal state and the current subgoal, both of which had values corresponding to ball-position pairings (e.g. red in position6). In the current state, we included one attribute for each of the six positions on the gameboard (position1 on the short peg, position2 and position3 on the medium peg, and positions4–6 on the long peg). The values of these attributes represent the content of the specified position in the current configuration (empty, blue, red, or green). We also included one status attribute for each of the three balls whose value indicates whether that ball is clear (moveable) or blocked. Finally, we also found it necessary to represent

Table 1
Representations and knowledge encoded in the production system and attractor network model of the Tower of London

Representational medium		
Type of content	Attribute/network	Values/attractors
Current state	Position1, . . . , position6	Empty, red, green, blue
	Red-status, blue-status, green-status	Blocked, clear
	In-target, above-source	Empty, red, green, blue
	Free-position	Position1, . . . , position6, none
Move	Move	Red-to-1, . . . , red-to-6
		Green-to-1, . . . , green-to-6
		Blue-to-1, . . . , blue-to-6
		None
Goal	Goal	Red-to-1, . . . , red-to-6
		Green-to-1, . . . , green-to-6
		Blue-to-1, . . . , blue-to-6
		TOL (top-level), none
Subgoal	Subgoal	Red-to-1, . . . , red-to-6
		Green-to-1, . . . , green-to-6
		Blue-to-1, . . . , blue-to-6
		None
Knowledge		
Type of content	Conditions/source of association	Actions/target of association
Legal moves	Position1, . . . , position6	Move
	Red-status, blue-status, green-status	
Achieve (sub)goals	Goal, subgoal	Move
Remove blockers	In-target, above-source, free-position	Subgoal
Inhibit what's done	Position1, . . . , position6	Goal, subgoal

The top of the table shows the attributes/values and corresponding networks/attractors that were used as the representational medium for the system. The bottom of the table shows the production rules and corresponding associations between attractors that were used to encode the knowledge in the system.

three pieces of information about the current state in relation to the current goal. Specifically, the system needs to represent explicitly what is in the target position for the current goal, what if anything is above the ball to which the goal refers, and what position if any could be used as a placeholder for obstructing balls. (This information is needed to generate new subgoals to remove balls that are blocking goal-achieving moves.) We therefore included attributes for in-target, above-source, and free-position as part of the current state. For example, suppose the current goal is to get the blue ball to position4 (the bottom of the long peg). Further, suppose the red ball is currently in position4, that nothing is above the blue ball, and that position1 (on the short peg, which for this goal is neither the source peg nor the target peg) is currently empty. In that case, in-target would be red, above-source would be empty, and free-position would be position1.

In addition to the representational medium in Table 1, the system required knowledge, in the form of production rules, about how to perform the task. Four types of knowledge were necessary. Firstly, the system needed to know that it should consider legal moves and should not consider illegal moves. We therefore included production rules that tested aspects of the current state and proposed moves that were legal (the ball to be moved is not blocked and the target position is free and supported). Secondly, the system needed to know that it should make moves that achieve goals or subgoals rather than just moving randomly. This knowledge corresponded to production rules that tested the goal/subgoal and voted for moves that achieved a (sub)goal and against moves that did not. Thirdly, the system needed to know how to generate new subgoals to remove balls currently blocking goal-achieving moves. To implement this knowledge, we included productions that proposed moving balls that were in-target or above-source (such balls would be blocking a goal-achieving move) to free-position. Fourthly, the system needed to stop working on goals that had already been achieved. We implemented this knowledge in production rules that tested the current location of a ball and rejected goals to move that ball to its current location.

In addition to these four types of knowledge, the system also required the ability to update the current state after each move. We assume this ability depends on vision (at least when moves are physically made rather than just imagined) which we did not wish to model. Instead, we simply included a (rather complicated) set of productions that tested the current move and updated the current state appropriately.

Of course, our goal was not to develop a production system model, but rather to develop a neural network model with which we could explore neural mechanisms of control and the effects of damage. Our next step was therefore to exploit the mapping from production systems onto neural networks in order to construct an attractor network model for the task.

Recall that according to the mapping, attributes correspond to attractor networks, values correspond to attractors (stable distributed activity patterns), and production rules correspond to asymmetric associations between attractors. Accordingly, we built attractor networks for each of the attributes listed in Table 1 and set the internal weights so that each network would have a sufficient number of attractors to correspond to all the different values that needed to be represented. For example, we constructed attractor networks for each of the six positions and each of these networks contained four attractor patterns (one corresponding to empty, another to blue, another to red, and another to green). Similarly, we built an attractor network for moves that contained attractor patterns corresponding to all possible moves (ball–target pairs) and attractor networks to represent goals and subgoals. Finally, we built connections between networks to represent the asymmetric associations reflected in the production rules.

Fig. 5 shows a schematic of the resulting model. In keeping with the four types of production rules, we had four types of connections. (1) Connections to excite legal moves and to inhibit illegal moves (from current state to move in the figure). (2) Connections to excite (sub)goal achieving moves and to inhibit other moves (from goal and subgoal to move in the figure). (3) Connections to propose subgoals to remove balls currently blocking goal-achieving moves (from current state [specifically, above-source, in-target, and free-position] to subgoal in the figure). (4) Connections to inhibit goals that had already been achieved (inhibitory connections from current state to goal and subgoal).

Excluding the influence of the goal and subgoal networks on behavior, the operation of the model is fairly simple. The representation of the current configuration excites all legal moves in the move network and moves that involve blocked balls are strongly inhibited by the ball status networks. The legal moves then compete with each other via attractor dynamics in the move network until one wins and the move network converges. Without other sources of input, the move that is selected is random and simply depends on noise. Once a move is selected, the representation of the current state is updated to reflect the move.[4] The new current state then once again votes for legal moves and the process begins again. In short, in the absence of input from goals and subgoals, the model simply performs random search using any legal moves.

The goal and subgoal networks modulate processing in the move network by exciting moves that achieve the current (sub)goal and by inhibiting moves that do not. This modulation biases the competition in the move network so that moves that achieve (sub)goals tend to be selected. If

[4]We do not model updates to the current state using neural networks. As previously discussed, we assume this computation is performed by complex visual processing (at least when moves are physically made) and we are not trying to model the details of this process. We therefore simply wrote Matlab code that updates the current state networks whenever the move network converges on a move.

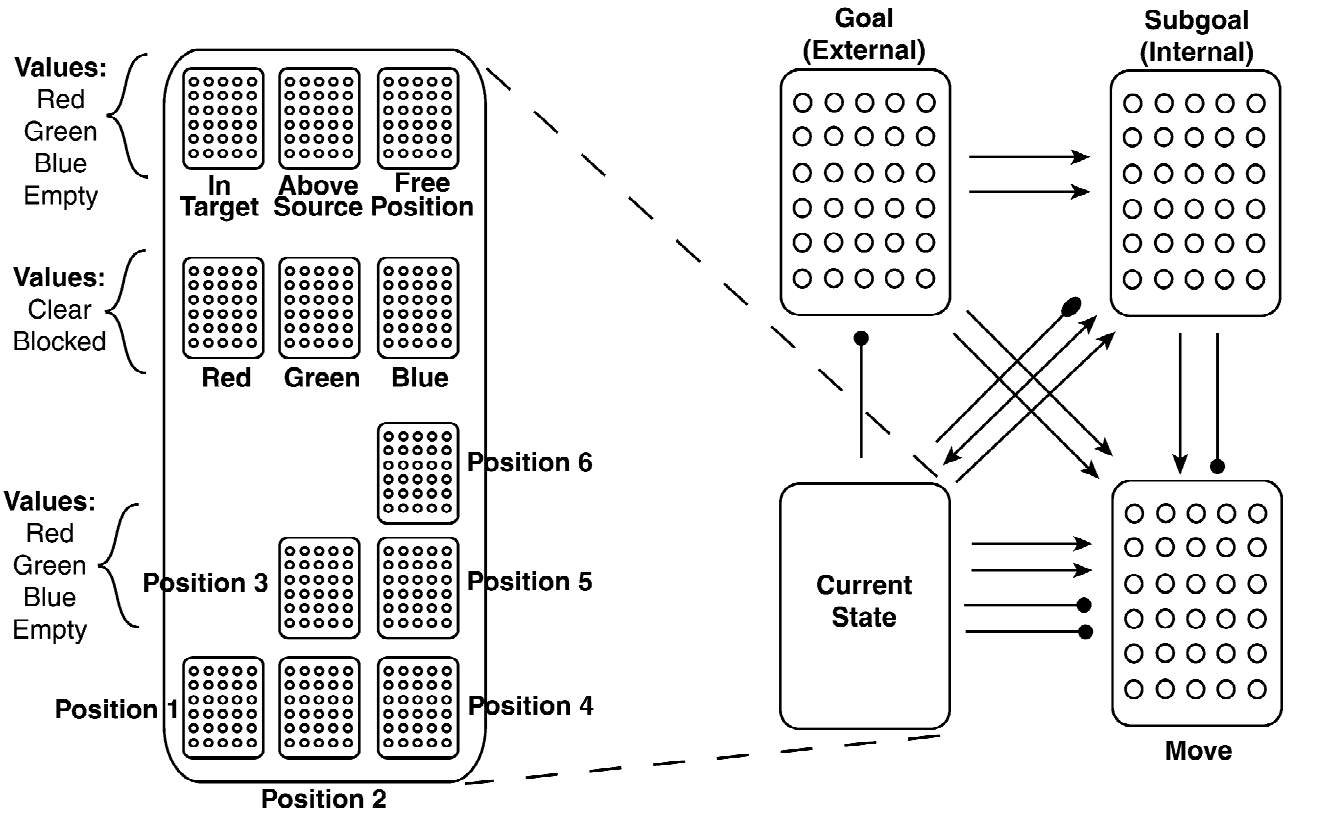

Fig. 5. Schematic of the model showing the modular attractor networks that represent current goals, subgoals, moves and current states. Move and goal representations are conjunctions of ball colors and target positions (e.g. 'move red to position5'). A current state consists of a set of perceptual features specifying the colors of the balls in each gameboard position and the status of each ball in terms of whether it can be moved or whether it is blocked. Finally, the top row of networks represents the color of the ball occupying the target position of the current goal, the color of the ball above the goal ball's current position, and the lowest position on the peg which is neither the source nor the target of the current goal (a peg which is ideal for holding blocking balls).

no legal move can achieve the current (sub)goal, then no move is selected because the current (sub)goal inhibits all moves that do not achieve it. In that case, the move network converges to no move, above-source, in-target, and free-position are updated, and they vote for a new subgoal to remove an obstructing ball.

5.1. Simulation #1: modeling normal performance

Fig. 6 presents data we collected from 42 undergraduate students as well as the behavior of the model on 18 TOL problems of varying difficulty. The human data were collected using the methods (and software) described in Owen et al. [36] and we also used the same 18 problems. The figure plots number of moves as a function of minimum number of moves (a measure of problem complexity). Thus, optimal performance corresponds to the diagonal line in which the number of moves equals the minimum possible number of moves.

Two phenomena are strikingly apparent in the human data. First and foremost, intact participants are exceedingly good at the task. For problems involving fewer than five moves, performance is almost optimal with the mean number of moves very close to the minimum possible. Furthermore, the mean number of moves is never more than two over the minimum possible (and the median is

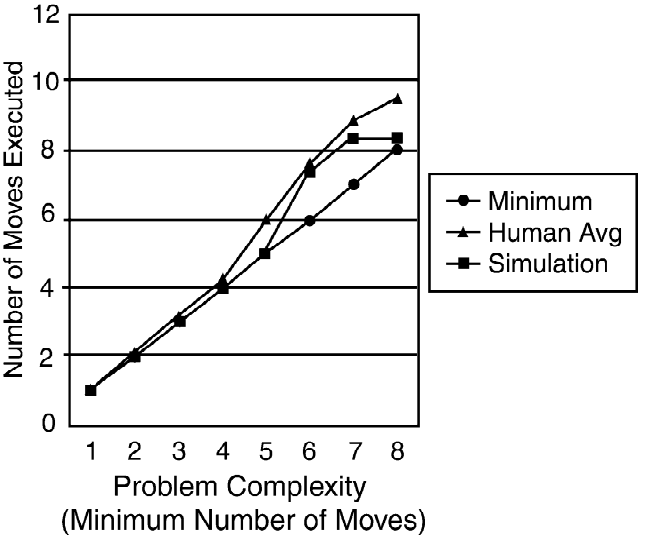

Fig. 6. Optimal performance, average human performance, and intact model performance across 18 Tower of London problems from Owen et al. [36].

always less than one move over minimum, not shown). Clearly, these participants can solve these problems and can do so relatively efficiently. Secondly, there is a clear effect of complexity on the number of extraneous moves. The vast majority of extraneous moves occur on the problems requiring five or more moves; for simpler problems there are very few extraneous moves. This pattern of results is consistent with previous findings in the literature [30,36,49].

As the figure illustrates, the model also exhibits both of these behavioral patterns. Most importantly, the model is capable of solving all the problems relatively efficiently. To our knowledge, no neural network model has ever solved the hardest problems with this level of efficiency. Also, like the human data, the model's efficiency is affected by the complexity of the problem. On simpler problems involving five or fewer moves, the model's behavior is essentially optimal. It only begins to make extraneous moves on the most complex problems involving six or more moves.

5.2. Simulations #2 and #3: modeling prefrontal deficits

Before we can simulate the effects of prefrontal lesions, we must decide which part of the model to damage. Prefrontal patients are often described as having a deficit in top-down control. Their perceptual and motor systems may be relatively intact, but they have difficulty organizing their behavior. They are often strongly influenced by affordances in the environment (e.g. as in utilization behavior) and have difficulty inhibiting behavior that is prepotent in the current context. They are easily distractible by information in the environment and thus have difficulty focusing on internally generated information. The part of the model that most naturally matches these characteristics is the subgoal network. Unlike the current state and move representations, the subgoal network is involved in top-down control rather than perceptual or motor processing. And unlike the goal network, the subgoal network represents internally generated control signals rather than externally provided ones (like affordances and many prepotent behaviors). We therefore chose to simulate prefrontal impairments by damaging the subgoal network in the model. We did so simply by removing a random sample of units from the subgoal network (including the connections to and from those units).

5.2.1. Group×difficulty interaction

Fig. 7 presents the performance of intact participants, prefrontal patients, and the model with varying amounts of damage to the subgoal network. On the left are data from Owen et al. [36] and in the middle are data from Carlin et al. [8]. In both cases prefrontal patients perform worse than the control participants do, but this difference is really only apparent on the harder problems. That is, there is a clear group (patients vs. controls) by difficulty (easy vs. hard) interaction. Consistent with these behavioral results, neuroimaging studies confirm that prefrontal cortex activity is greater on more complex problems [2,37].

The model exhibits a similar pattern of behavior. With increasing amounts of damage the model's performance degrades, but this effect is much more apparent on the harder problems than on the easier problems (right side of figure). The reason the model produces this interaction (and therefore the explanation it provides) is that subgoals become increasingly important in the more complex problems (for evidence consistent with this hypothesis, see Refs. [19,43,49]). In particular, at most three moves on the minimal solution path can achieve the final goals (because there are only three balls). Therefore, other moves on that path must be based on internally generated plans/subgoals

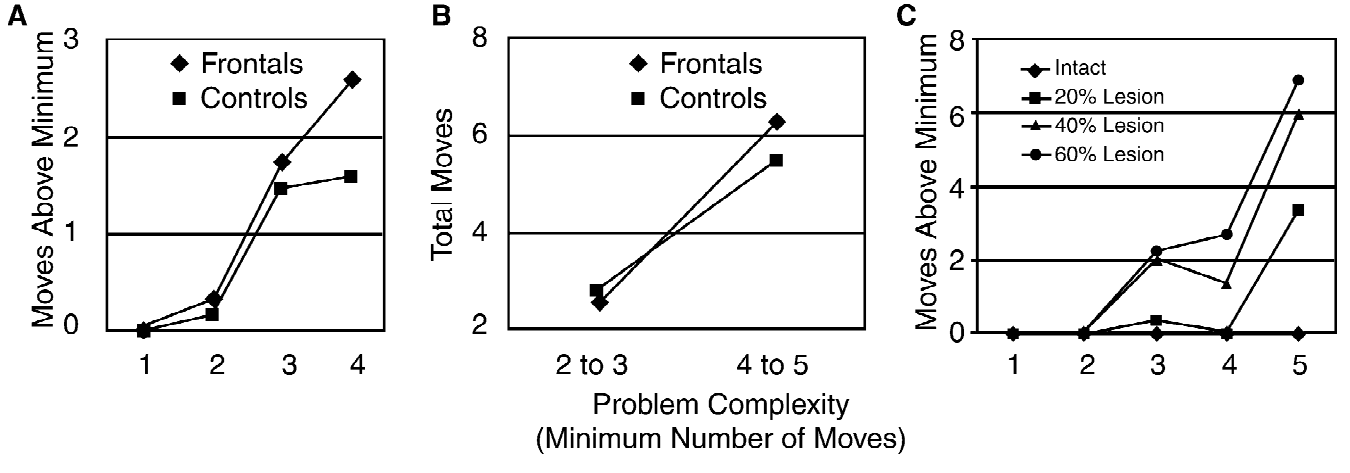

Fig. 7. (A) Tower of London performance of patients with prefrontal damage and intact control participants from Owen et al. [36]. The figure shows the number of moves above minimum and illustrates that prefrontal patients show larger impairments on harder problems than on easier problems (a group×difficulty interaction). (B) Similar data from Carlin et al. [8] also demonstrating a group×difficulty interaction. (C) Performance of the neural network model with varying amounts of damage to the subgoal network. Like the human data, damaging the model leads to disproportionate difficulty with the hardest problems.

rather than on externally provided constraints. For example, an eight-move problem requires at least five such moves whereas a three-move problem requires at most one.

5.2.2. Failure to inhibit prepotent responses

We also tested the effect of damage on the ability of the model to solve problems that require prepotent (goal-achieving) moves to be inhibited (this test was inspired by a similar simulation in Ref. [14]). Fig. 8 shows the results. On the left of the figure is the model's performance on a problem in which the prepotent move is correct. Notice that damage had no effect in this case. The reason is that behavior could be controlled by the intact goal network (which biases the system toward prepotent, goal-achieving moves). On the right of the figure is the model's behavior on the same problem, but with the initial and goal states reversed. In this problem, however, there is a prepotent move that is actually incorrect (that is, it is not on the minimal solution path). In this case, increasing amounts of damage lead to increasingly severe impairments in performance. The model therefore predicts an interaction between problem type (prepotent correct vs. incorrect) and subject group (controls vs. prefrontal patients).

The success of the damaged model to simulate the behavior of prefrontal patients on the Tower of London task leads to the hypothesis that the role of DLPFC in these kinds of tasks is to represent internally-generated subgoals that bias competition among options.

6. General discussion

Our three main theoretical points can be summarized as follows: firstly, there is a natural mapping from certain key aspects of goal-driven production systems onto attractor neural networks. The mapping is directly supported by properties of attractor networks that are both well understood computationally and broadly consistent with what is known about cortical processing. Secondly, this mapping makes it possible to develop neural models of complex problem solving tasks such as the Tower of London. A key demonstration of the efficacy of this mapping is the ability of the model to account for the problem solving behavior of normal subjects on a range of Tower of London tasks, including fairly challenging 6–8-step problems that were solved reasonably well by both subjects and model. Thirdly, the mapping and the Tower of London model in particular led to a hypothesis about the role of DLPFC in problem solving: DLPFC supports the representation of internally generated subgoals that modulate among choices. This is a novel hypothesis that has not been directly tested empirically, but the correspondence of the damaged model's behavior with that of frontal patients suggests that the hypothesis is a viable one, at least in this task domain.

6.1. Relationship to other work

Dehaene and Changeux [14] (henceforth DC97) also

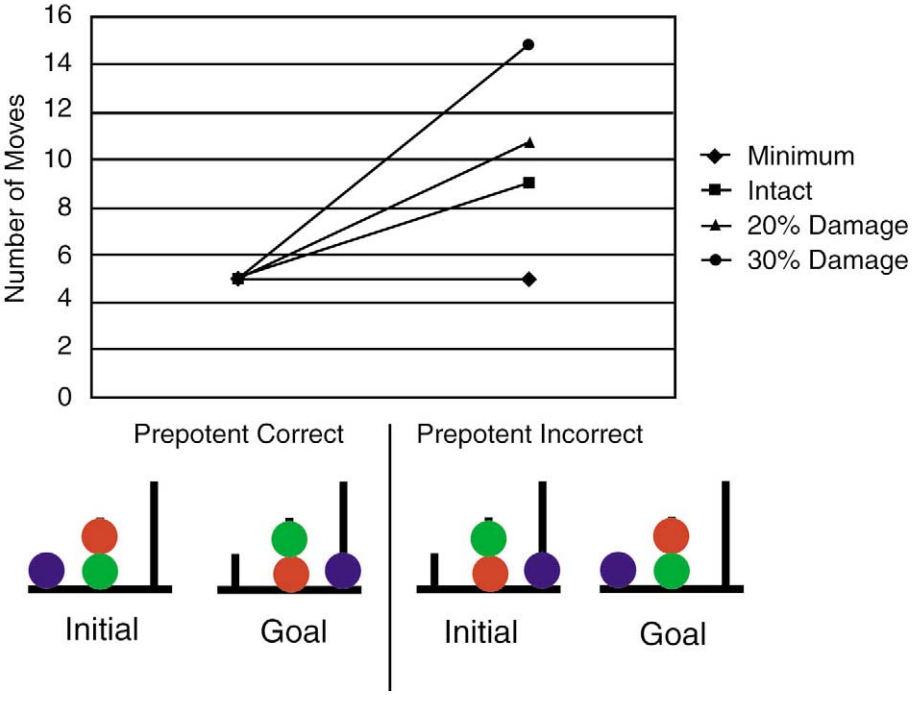

Fig. 8. The effects of damage on the model's ability to inhibit prepotent moves that are incorrect. The figure shows the number of moves made by the model under different amounts of damage for two different problems. The two problems involve the same moves, but in reverse order (the initial and goal states have been reversed). For the problem on the left (prepotent correct), all moves that would place balls in their final positions are correct (are on a minimal solution path). For the problem on the right (prepotent incorrect), a suboptimal move is available which would achieve a goal, but which would have to be undone in order to achieve a solution. Damage to the model leads to disproportionate difficulty with the problem on the right in which a prepotent move must be inhibited.

proposed a neural network model of the Tower of London, and it is instructive to consider the relationship between their model and the attractor model described here. We wish to focus on two distinguishing features of the DC97 model. Firstly, it is comprised of a hierarchy of processing levels that correspond to plan, operation, and gesture levels. (An operation is a single move in the problem composed of two gestures: pointing to the source and destination location). Secondly, it is an architecture that implements a simple greedy search algorithm with a look-ahead mechanism. At each point, if a move is available that puts a ball into a goal position, that move is selected; else one is chosen randomly. If a chosen move turns out to increase the number of balls in a goal position, the current problem state is deliberately stored so that it may be returned to. If a move turns out to decrease the number of balls in a goal position, the path is abandoned by returning to a previously remembered state.

There are several important theoretical and empirical differences between the DC97 model and the work described in this paper. Firstly, the model described here proposes that the generation and maintenance of subgoal representations is a critical part of problem solving in tasks such as TOL, and furthermore ascribes this function of subgoal representation to DLPFC. Secondly, the model is based on a task-independent and problem solving method-independent mapping of goal-driven production systems onto attractor networks. Thirdly, the model makes accurate predictions about the specific number of moves on individual problems, including relatively difficult problems involving up to eight moves. Fourthly, the model yields graded effects of damage, such that slight damage to the subgoal network produces less of an effect than greater damage.

This comparison brings into focus an important point: the empirical success of the current model—in particular, its ability to handle difficult problems in the same number of steps as subjects—is a direct function of the model's support for both subgoal representations and knowledge-intensive problem solving methods. One way to characterize the model is that it provides an architecture for implementing subgoal-mediated problem solving methods. The method is represented by specific associations (corresponding to productions) that encode knowledge about what subgoals to set and what operations to pursue to accomplish those subgoals. Again, this is in contrast to the DC97 model, which can be seen as a task-independent but method-specific architecture for implementing greedy lookahead.

Our approach thus follows work in modern cognitive architectures in carefully separating the knowledge implicated in specific problem solving methods from the architectural mechanisms that support its use. The success of the current model on the most difficult TOL problems is a small indicator of the utility of this approach in general, and of the proposed mapping to attractor networks in particular.

We hasten to add, however, that in other ways, the DC97 model is both more comprehensive and more complex than the model proposed here. It addresses multiple levels of control, and includes mechanisms for look-ahead and explicit state evaluation. These are important functions not yet realized in our current model.

The work presented here also builds on other computational theories of prefrontal cortex function, most notably the models of Cohen and co-workers [9,10,29]. Using models of tasks such as Stroop that directly pit automatic against controlled processes, they have argued persuasively that an important computational function of prefrontal cortex is the modulation of competing posterior representations by explicit representations of current task goals. The model proposed here is entirely in this spirit, and it begins to demonstrate that modulating attractor networks can be assembled in a way that scales to problem solving tasks that are more complex than has previously been attempted. Furthermore, the precise way in which these networks are assembled can be guided by independent work on cognitive architectures that already have a long history in modeling problem solving.

Although we have focused here on problem solving in complex tasks, dorsolateral prefrontal cortex has been hypothesized to be involved in a wide variety of cognitive functions, including attention [3,6,10,12,35], various working memory functions [17,18,26,39], coordination of multiple tasks [46], representation of task context [7,11], voluntary action [38], goal management [19,29], inhibition of irrelevant or inappropriate responses [15], encoding of abstract rules [48], and others. It remains to be seen how the proposed model will fit into a more comprehensive theory that accounts for this multiplicity of prefrontal functions and possible functional specialization within prefrontal cortex. One common thread that seems to run through many of these functions, however, is that PFC represents behavioral intentions that persist and guide processing for stretches of time on the order of several seconds to tens of seconds. A number of researchers have argued for this kind of unifying principle (e.g. Refs. [29,38]), and it is wholly consistent with the model presented here. What we hope to do with the current work is begin to push models based on this idea in the direction of flexible, programmable neural network architectures that can be applied to tasks of greater and greater complexity.

References

[1] J.R. Anderson, C. Lebiere, The Atomic Components of Thought, Erlbaum, Mahwah, NJ, 1998.
[2] S.C. Baker, R.D. Rogers, A.M. Owen, C.D. Frith, R.J. Dolan, R.S.J. Frackowiak, T.W. Robbins, Neural systems engaged by planning: a PET study of the Tower of London task, Neuropsychologia 34 (1996) 515–526.
[3] M.T. Banich, M.P. Milham, R.A. Atchley, N.J. Cohen, A. Webb, T. Wszalek, A.F. Kramer, Z.P. Liang, V. Barad, D. Gullett, C. Shah, C. Brown, Prefrontal regions play a predominant role in imposing an

attentional 'set': evidence from fMRI, Cogn. Brain Res. 10 (2000) 1–9.
[4] R.S. Bapi, D.S. Levine, Modeling the role of frontal lobes in sequential task performance 1. Basic structure and primacy effects, Neural Networks 7 (1994) 1167–1180.
[5] S. Becker, M. Moscovitch, M. Behrmann, S. Joordens, Long-term semantic priming: a computational account and empirical investigation, J. Exp. Psychol.: Learn. Mem. Cognit. 23 (1997) 1059–1082.
[6] C.J. Bench, C.D. Frith, P.M. Grasby, K.J. Friston, E. Paulesu, R.S.J. Frackowiak, R.J. Dolan, Investigations of the functional anatomy of attention using the Stroop test, Neuropsychologia 31 (1993) 907–922.
[7] T.S. Braver, D.M. Barch, J.D. Cohen, Cognition and control in schizophrenia: a computational model of dopamine and prefrontal function, Biol. Psychiatry 46 (1999) 312–328.
[8] D. Carlin, J. Bonerba, M. Phipps, G. Alexander, M. Shapiro, J. Grafman, Planning impairments in frontal lobe dementia and frontal lobe lesion patients, Neuropsychologia 38 (2000) 655–665.
[9] J.D. Cohen, T.S. Braver, R.C. Oreilly, A computational approach to prefrontal cortex, cognitive control and schizophrenia: recent developments and current challenges, Phil. Trans. R. Soc. Lond. Ser. B: Biol. Sci. 351 (1996) 1515–1527.
[10] J.D. Cohen, K. Dunbar, J.L. McClelland, On the control of automatic processes: a parallel distributed processing account of the Stroop effect, Psychol. Rev. 97 (1990) 332–361.
[11] J.D. Cohen, D. Servan-Schreiber, Context, cortex, and dopamine: a connectionist approach to behavior and biology in schizophrenia, Psychol. Rev. 99 (1992) 45–77.
[12] R. Cooper, T. Shallice, Contention scheduling and the control of routine activities, Cogn. Neuropsychol. 17 (2000) 297–338.
[13] G.S. Cree, K. McRae, C. McNorgan, An attractor model of lexical conceptual processing: simulating semantic priming, Cogn. Sci. 23 (1999) 371–414.
[14] S. Dehaene, J.P. Changeux, A hierarchical neuronal network for planning behavior, Proc. Natl. Acad. Sci. USA 94 (1997) 13293–13298.
[15] R. Dias, T.W. Robbins, A.C. Roberts, Dissociation in prefrontal cortex of affective and attentional shifts, Nature 380 (1996) 69–72.
[16] M.J. Farah, R.C. O'Reilly, S.P. Vecera, Dissociated overt and covert recognition as an emergent property of a lesioned neural network, Psychol. Rev. 100 (1993) 571–588.
[17] S. Funahashi, C.J. Bruce, P.S. Goldman-Rakic, Mnemonic coding of visual space in the monkeys dorsolateral prefrontal cortex, J. Neurophysiol. 61 (1989) 331–349.
[18] J.M. Fuster, G.E. Alexander, Neuron activity related to short-term memory, Science 173 (1971) 652–654.
[19] V. Goel, J. Grafman, Are the frontal lobes implicated in planning functions: interpreting data from the Tower of Hanoi, Neuropsychologia 33 (1995) 623–642.
[20] V. Goel, S.D. Pullara, J. Grafman, A computational model of frontal lobe dysfunction, working memory and the Tower of Hanoi task, Cogn. Sci. 25 (2001) 287–313.
[21] J. Hertz, A. Krogh, R. Palmer, Introduction to the Theory of Neural Computation, Addison-Wesley, Redwood City, CA, 1991.
[22] G.E. Hinton, T. Shallice, Lesioning an attractor network: investigations of acquired dyslexia, Psychol. Rev. 98 (1991) 74–95.
[23] J.J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, Proc. Natl. Acad. Sci. USA Biol. Sci. 79 (1982) 2554–2558.
[24] J.J. Hopfield, Neurons with graded response have collective computational properties like those of two-state neurons, Proc. Natl. Acad. Sci. USA Biol. Sci. 81 (1984) 3088–3092.
[25] M. Jones, T.A. Polk, An attractor network model of serial recall, Cogn. Syst. Res. 3 (2002) 45–55.
[26] D.Y. Kimberg, M.J. Farah, A unified account of cognitive impairments following frontal lobe damage: the role of working memory in complex, organized behavior, J. Exp. Psychol. Gen. 122 (1993) 411–428.
[27] R.L. Lewis, Cognitive modeling, symbolic, in: R. Wilson, F. Kyle (Eds.), The MIT Encyclopedia of the Cognitive Science, MIT Press, Cambridge, MA, 1999.
[28] D.E. Meyer, D.E. Kieras, A computational theory of executive cognitive processes and multiple-task performance. 1. Basic mechanisms, Psychol. Rev. 104 (1997) 3–65.
[29] E.K. Miller, J.D. Cohen, An integrative theory of prefrontal cortex function, Annu. Rev. Neurosci. 24 (2001) 167–202.
[30] R.G. Morris, S. Ahmed, G.M. Syed, B.K. Toone, Neural correlates of planning ability: frontal lobe activation during the Tower of London test, Neuropsychologia 31 (1993) 1367.
[31] M.C. Mozer, M. Behrmann, On the interaction of spatial attention and lexical knowledge: a connectionist account of neglect dyslexia, J. Cogn. Neurosci. 2 (1990) 96–123.
[32] A. Newell, Production systems: models of control structures, in: W.C. Chase (Ed.), Visual Information Processing, Academic Press, New York, 1973.
[33] A. Newell, Physical symbol systems, Cogn. Sci. 4 (1980) 135–183.
[34] A. Newell, Unified Theories of Cognition, Harvard University Press, Cambridge, MA, 1990.
[35] D.A. Norman, T. Shallice, Attention to action: willed and automatic control of behaviour, in: R. Davidson, G. Schwartz, D. Shapiro (Eds.), Consciousness and Self-regulation: Advances in Research and Theory, Plenum, New York, 1986, pp. 1–18.
[36] A.M. Owen, J.J. Downes, B.J. Sahakian, C.E. Polkey, T.W. Robbins, Planning and spatial working memory following frontal lobe lesions in man, Neuropsychologia 28 (1990) 1021–1034.
[37] A.M. Owen, J. Doyon, M. Petrides, A.C. Evans, Planning and spatial working memory: a positron emission tomography study in humans, Eur. J. Neurosci. 8 (1996) 353–364.
[38] R.E. Passingham, The Frontal Lobes and Voluntary Action, Oxford University Press, Oxford, 1993.
[39] M. Petrides, The role of the mid-dorsolateral prefrontal cortex in working memory, Exp. Brain Res. 133 (2000) 44–54.
[40] D.C. Plaut, J.L. McClelland, M.S. Seidenberg, K. Patterson, Understanding normal and impaired word reading: computational principles in quasi-regular domains, Psychol. Rev. 103 (1996) 56–115.
[41] D.C. Plaut, T. Shallice, Deep dyslexia: a case-study of connectionist neuropsychology, Cogn. Neuropsychol. 10 (1993) 377–500.
[42] T.A. Polk, C. Behensky, R. Gonzalez, E.E. Smith, Rating the similarity of simple perceptual stimuli: asymmetries induced by manipulating exposure frequency, Cognition 82 (2002) B75–B88.
[43] T.M. Rushe, R.G. Morris, E.C. Miotto, J.D. Feigenbaum, P.W.R. Woodruff, R.M. Murray, Problem-solving and spatial working memory in patients with schizophrenia and with focal frontal and temporal lobe lesions, Schizophr. Res. 37 (1999) 21–33.
[44] T. Shallice, Specific impairments of planning, Phil. Trans. R. Soc. Lond. Ser. B: Biol. Sci. 298 (1982) 199–209.
[45] H.A. Simon, The patterned matter that is mind, in: D. Steier, T. Mitchell (Eds.), Mind Matters: Contributions to Cognitive and Computer Science in Honor of Allen Newell, Erlbaum, Hillsdale, NJ, 1996.
[46] E.E. Smith, J. Jonides, Storage and executive processes in the frontal lobes, Science 283 (1999) 1657–1661.
[47] J. Tanaka, M. Giles, S. Kremen, V. Simon, Mapping attractor fields in face space: the atypicality bias in face recognition, Cognition 68 (1998) 199–220.
[48] J.D. Wallis, K.C. Anderson, E.K. Miller, Single neurons in the prefrontal cortex encode abstract rules, Nature 411 (2001) 953–956.
[49] G. Ward, A. Allport, Planning and problem-solving using the five-disc tower of London task, Q. J. Exp. Psychol. Sect. a: Hum. Exp. Psychol. 50 (1997) 49–78.