Columnar Timing Mechanisms in Neural Models of Problem Solving

Patrick Simen, Eric Freedman, Rick Lewis, Thad Polk

August 4, 2003

email: psimen@eecs.umich.edu
Keywords: Timing, Problem Solving, Cortical Column

Abstract

By using a columnar subnetwork of continuous-time sigmoid activation units as a building block, hierarchical networks can be constructed that serve as cognitive models of algorithmic behavior, including goal-directed problem solving. Previous work by Polk et al. [28] demonstrated the power of a hierarchical composition of local attractor networks for modeling problem solving in the Tower of London task, but critical timing issues were addressed by non-neural components. The essential function of the column structure proposed here for addressing timing issues is to produce a controllable propagation delay in the signal from a column's input unit to its output unit. Lateral inhibition between input units and between output units in different columns forms layered decision-making modules. These modules use winner-take-all attractor dynamics to compute the results of simple if-then rules applied to the feedforward outputs of other modules. Strong recurrent excitation in multiple layers of these columnar modules produces activation-based short-term memory that preserves the symbolic results of these computations during algorithmic processing. We show that propagation delay between layers allows the application of timing methods from digital sequential circuit design that solve the timing problems inherent in the existing Tower of London model without relying on discrete-time updating or binary values.

1 Introduction

Computational cognitive models of human behavior in complex cognitive tasks have been difficult to construct in ways that can be plausibly mapped onto the brain. Symbolic models of cognitive tasks typically rely on computational principles common in computer science: for example, buffers may preserve symbolic information until overwritten, list and stack data structures may play a central role, and program state may be updated in a sequence of discrete transitions spaced uniformly in time. Because these principles ensure universal computational power [26] and relatively straightforward construction and interpretation, symbolic cognitive models have formed a conceptual paradigm for cognitive psychologists interested in higher cognition [24]. However, it is often unclear how brain structures could implement these computational principles. Neural models, on the other hand, have rarely been constructed of complex, algorithmic behavior [27, 5], and when they have [10, 28], they have not been easily extensible into general models of complex computation on a par with symbolic systems such as ACT [3].

Our research suggests that the primary obstacle to general symbolic processing functionality by neural cognitive models is the unmet need for a large degree of control over the time course of

neural activation. This may be considered a virtue of neural approaches, however. Symbolic models typically rely on implausible assumptions about the temporal aspects of internal representations that are made possible by the underlying computer architecture: i.e., there is a regular, system-wide clock pulse which updates all binary-valued memory components in the system synchronously, so that symbolic representations can be assumed to activate or inactivate instantaneously, with no 'rise time' or 'decay time'. A host of timing problems is eliminated with this assumption, and discrete mathematics and theoretical computer science can then be applied directly to algorithm design without reference to low-level hardware issues.

Here we present critical elements of an architecture for neural cognitive modeling that abandons some traditional symbolic computing principles - particularly discrete time representation and binary values - while at the same time emulating some of the low-level digital hardware components upon which symbolic systems are based. Some of these elements are present in an existing model of the Tower of London task used in cognitive psychology [28]. We briefly review the principles of this model, which relies on non-neural timing mechanisms for a critical aspect of its operation, and then demonstrate the replacement of this mechanism with neural timing mechanisms.

2 A Limited Neural Production System in a Hybrid Tower of London Model

One way to build neural networks that can carry out search in a problem space is to emulate symbolic production systems [26]: parallel systems of if-then rules in which rules compete for activation based on the contents of a working memory. In Polk et al. [28], we demonstrate the utility of the production analogy. There we describe a feedforward composition of locally recurrent attractor network modules in which each module effectively holds an 'election' between all the candidate representations for which upstream modules are voting (similar to Feldman & Ballard [14]). When feedforward connections from the units highly active in attractor pattern u of upstream network U excite the units in downstream network D which are highly active under the symbolic representation d, the system is said to be implementing the production: *if* $U = u$, *then* $D = d$. The downstream network D may then retain an activation-based short-term memory of the value d after all units in U return to baseline.

We then apply this framework to the Tower of London (TOL) task (similar to the Tower of Hanoi), shown at the bottom of Fig. 7. This task has been used extensively to assess planning impairments and is thought to depend crucially on goal management [30, 33]. It is a variant of the Tower of Hanoi problem and involves moving colored balls on pegs from an initial configuration until they match a goal configuration (Fig. 7). Unlike the Tower of Hanoi problem, 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 (the first peg can hold one ball, the second peg can hold two, and the third peg can hold three). There is typically one red, one green and one blue ball. 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.

Modules in Polk et al. are clusters of fully connected, laterally inhibiting, self-exciting 'grandmother cells' governed by a continuous activation update rule. Considered in isolation from external input, these modules are attractor networks with the desirable property that mutual inhibition within them provides winner-take-all dynamics in many situations [20, 9]. This property makes them suitable as individual decision makers [1, 14], in addition to their potential role as activation-based short-term memories [7].

In the model solver, a set of Sensory modules, one for each position of the gameboard, is

initialized to patterns encoding the color of a ball at that position, if any, and these representations then persist until reinitialized by changes in the environment. They excite the representations of legal moves in a separate Move module devoted to action representations, and inhibit illegal ones. Attractor dynamics within the Move module may then result in the selection of a single action for execution, completing the simulation of a simple production of the form: 'if the red ball is in position X, then place it in position Y'.

2.1 Goal-Directed Behavior

In the Tower of London solver, we dedicate one attractor module to the representation of externally defined goals and another to internally generated subgoals. Activation in the goal modules biases the competition taking place in the Move module, warping the energy landscape [20] of the module by widening and deepening the basin of attraction around desired patterns. This biasing is just another form of production, but the if-condition is semantically special: it represents a desired state of the environment. Further, the biasing strength of such a production is insufficient to activate its then-condition without support from some other module, as in the case of the Sensory module just discussed. Technically then, this Goal $\rightarrow$ Move excitation should be considered only a component of a production of the form: 'if Goal is X and Percept is Y, then Do Z'. By incorporating goal representation modules, the model in Polk et al. [28] solves problems in a fashion similar to normal subjects.

2.2 Remaining Issues

The scheme so far described for building problem-solving neural networks addresses critical issues, but it leaves an important question unresolved: can feedforward compositions of modular attractor networks themselves form recurrent cycles? The system in Polk et al. [28], for example, is actually a hybrid neural/symbolic system. It requires a non-neural component to read the output of a feedforward composition of modules, which emerges in the Move module representing the current action. If a single, clear winner emerges in this network during voting by Sensory and Goal modules, the system takes the prescribed action and then reinitializes several modules in the network to new values. It thereby closes the loop that feeds output information back into the system. However, in the smoothly continuous systems we propose, such a process cannot be instantaneous: new module values will have a nonzero rise time, and old values will have a nonzero decay time. Thus a timing mechanism is required for ensuring proper 'setup' of inputs for the next cycle of computation.

In many situations, however, an action should never emerge, because no atomic action can achieve the current goal. In such a case, the network remains near baseline activation, thereby signaling that a subgoal ought to be generated in order to produce environmental conditions suitable for taking actions to achieve the parent goal. This raises a second question: for how long should an election be allowed to continue? Convergence times within attractor networks are difficult if not impossible to predict in many cases. Further, a locally recurrent network with non-uniform or asymmetric lateral connections may display complex transient behavior before settling toward an equilibrium, as we show in section 4, and we would like our timing mechanisms to generalize to less restrictive assumptions about representation (namely, we would like them to be compatible with distributed representations as would more likely arise in a system that self-organized through Hebbian learning [2, 4]).

A simple solution is simply to wait for some period of time after one pattern of activity has remained approximately unchanged to declare either a winner or convergence to baseline: the waiting period can be determined by the cost of waiting too long to get a true winner relative to

the cost of responding too quickly with a false winner. If we can sample the activation of a module only after a 'safety period' has elapsed, we can reduce the chance of picking a false winner. If we are able to construct a timer which is triggered whenever module activity fails to satisfy the criteria for representing a winner, and which is inactivated by the emergence of a winner, we can produce a convergence-to-baseline detector that is triggered when the timer reaches a threshold duration without being inactivated.

Neural mechanisms for cycle timing and convergence detection are presented in the next two sections. First, two meanings of the word 'timing' should be distinguished according to the distinction between 'closed-loop' and 'open-loop' control in the control systems literature. In the former, some control process operates with feedback from the controlled system, and in the latter without feedback. Cognitive models built with symbolic production systems and sequential logic circuits can each exhibit both kinds of timing. In production systems, closed-loop timing is exhibited when a production sets some variable in working memory which is part of the precondition of a second production, but when the second production must wait to fire until some second precondition is satisfied - in other words, the system is prepared for an action but is waiting for a 'go' signal, a form of feedback. The same is true at the hardware level when a memory cell in RAM is accessed by a read signal and an address signal: all memory locations are prepared to output their contents simultaneously, but the temporal order of the address signals determines the output order. Open-loop timing in production systems is exhibited by any production that, once it fires, initiates some counting process resulting in an action or change to working memory after a duration that is a multiple of the production-firing cycle duration. In hardware, the duration of the clock pulse itself is a form of open-loop timing designed to let the voltages of all physical components settle to equilibrium values before the next cycle of operation [17]. For our purposes, closed-loop timing refers to the ability to control the sequential order of events with trigger signals, and open-loop timing refers to an internal representation of duration. We address closed-loop timing mechanisms in section 3 and open-loop timing mechanisms in section 4. Section 5 brings these mechanisms together to replace the non-neural components in Polk et al.

3 Closed-loop Timing: Latches, Gates and Flip-Flops

In order to integrate closed-loop timing mechanisms into the existing problem-solving model, some details of the model's operation must be presented. We begin with a general discussion of the properties of units employing recurrent self-excitation that will also be essential in the discussion of open-loop timers. These properties also explain the operation of the existing model, as we show. We then integrate closed-loop timing mechanisms into this model, and conclude with a description of the entire model minus the convergence detection mechanism added in section 5.

3.1 Neural Activation and Positive Feedback

First we state the neural activation function for single units. The activation of unit i, $V_i \in [0,1]$, is determined by a standard nonlinear differential equation that is taken to model the firing rate of a population of neurons, possibly averaged over time so that more recent firing contributes more to the average than firing that occurred longer ago:

$$\frac{dV_i}{dt} = -V_i + \frac{1}{1 + e^{-\lambda(NetIn_i - \theta_i)}} \tag{1}$$

where $NetIn_i = \sum_{j=1}^{n} w_{ij} V_j$, and w_{ij} is the synaptic weight on the connection from unit j to unit i. [9] (See also Wilson & Cowan [35].) A small random noise term is also often added to V_i.

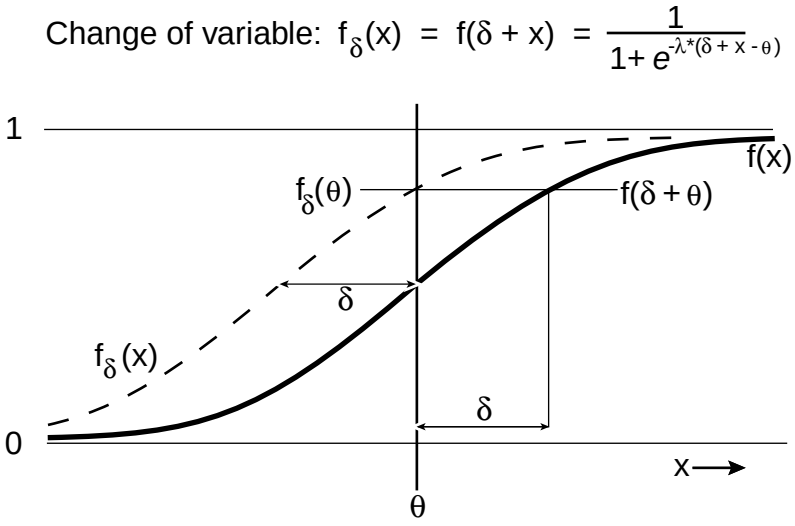


Figure 1: The effective activation function.

Positive feedback within individual units plays an important role in what follows. We define a *self-exciting unit* to be a unit i whose output value V_i is weighted by a nonzero synaptic strength w_{ii} and added to the $NetIn_i$ term of its own activation function. Non-self-exciting units have $w_{ii} = 0$.

We now state a fact which is useful for characterizing the behavior of a self-exciting unit under external excitation or inhibition. First we note that the sigmoid function $f(x) = \frac{1}{1+e^{-\lambda(x)}}$ is strictly less than $f(x+\delta)$, for $\delta > 0$. Similarly, $f(x) > f(x+\delta)$ for $\delta < 0$. This follows by the monotonicity of $f(x)$. Now we define the *effective activation function* $f_\delta(x)$ to be $f(x+\delta)$. This notation simply denotes a change of variable on the x axis. Thus $f_\delta(x)$ is $f(x)$ shifted leftward by δ when $\delta > 0$ and rightward by $|\delta|$ when $\delta < 0$. See Fig. 1.

The following lemma allows us to use a graphical 'cobweb diagram' method to characterize both the equilibrium points and the temporal behavior of self-exciters. Equation (1) is not analytically solvable due to the form of its nonlinearity, so we have no closed-form expression for the value of a unit's activation at any given time. The following tool will, however, allow us to compare the rates at which two self-exciters will approach their equilibrium points (which always exist in the case of constant external input, as the cobweb diagram will indicate). This will prove useful when we construct systems that require one unit to approach its attractor more slowly than another unit. The analysis depends on separating the contribution of a self-exciting unit's activation to its own net input (the *internal input*: $Int_i = w_{ii} \cdot V_i$) from the contributions by other units (the *external input*: $Ext_i = NetIn_i - Int_i$).

Lemma 3.1 *The time rate of change of V_i for a self-exciting unit with constant net external input Ext_i from other units is equal to the size of a 'stair step' formed between the sigmoid curve $f^i_{Ext_i}(Int_i)$ and the positive part of the line $\frac{1}{w_{ii}} \cdot Int_i$ (hereafter called the* reference line), starting at the value $f^i_{Ext_i}(Int_i(t_0))$. *(See Fig. 2).*

Proof: Each stair step is formed by tracing from $f^i_{Ext_i} = V_i$ horizontally to the reference line, and then vertically to $f^i_{Ext_i}(Int_i)$. The significance of the reference line is that

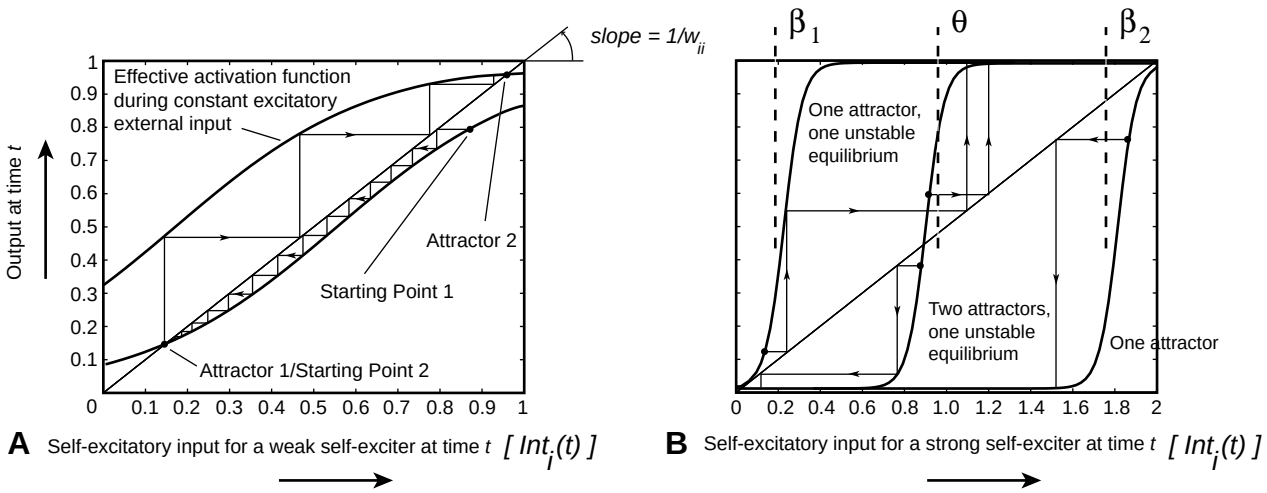


Figure 2: Cobweb analysis of weak and strong self-exciting units.

it translates the output of unit i into the weighted input it experiences due to its recurrent self-excitatory connection. An iterated system of stair steps gives a cobweb diagram which describes the evolution of the related discrete-time difference equation $V_i(t+1) = V_i(t) + (f^i_{Ext_i}(Int_i(t)) - V_i(t)) = f^i_{Ext_i}(w_{ii}V_i(t))$. In this case, to determine the value of V_i at the next time step, we simply need to run the currrent value of V_i weighted by w_{ii} through the effective activation function. The stair step magnitude is $f^i_{Ext_i}(w_{ii}V_i) - V_i$, and the discrete-time system increases or decreases rapidly in regions with large steps upward or downward respectively. The stair step magnitude is also the righthand side of the continuous time differential equation $\frac{dV_i}{dt} = -V_i + f^i_{Ext_i}(w_{ii}V_i)$, so when the stair step is large (small), $\frac{dV_i}{dt}$ is large (small), and the continuous-time system also increases or decreases rapidly in regions with large steps upward or downward respectively.

It will be useful to divide the family of parameterizations of self-exciting units into two classes as follows:

Def. 3.1 *A weakly self-exciting unit i has $w_{ii} \leq 4/\lambda$, or $1/w_{ii} \geq \lambda/4$. A strongly self-exciting unit has $w_{ii} > 4/\lambda$, or $1/w_{ii} < \lambda/4$. We now characterize the stable and unstable equilibria of these two classes.*

Theorem 3.1 *For a weakly self-exciting unit i, there is a single asymptotically stable equilibrium value of V_i when Ext_i is held constant.*

Proof: We examine the case in which Ext_i is held equal to 0, since the result holds for all values of θ_i, and thus is independent of leftward and rightward shifts. The slope of the tangent to the sigmoid component of the activation function is given by the following:
$\frac{df^i(NetIn_i)}{dNetIn_i} = \frac{\lambda e^{-\lambda(NetIn_i - \theta_i)}}{(1+e^{-\lambda(NetIn_i - \theta_i)})^2}$
Evaluating this derivative at $NetIn_i = \theta_i$ gives the slope of the tangent at the inflection point, $\frac{df^i(\theta_i)}{dNetIn_i} = \lambda/4$. There can be only one intersection of activation sigmoid and

reference line in this case, because, by the definition of weak self-excitation, the reference line is at least as steep as the tangent to $f^i(NetIn_i)$ at $NetIn_i = \theta_i$ – see Fig. 2, part A.

For a weakly self-exciting unit with constant external input Ext_i, the rate of change for values of V_i above the intersection of $f^i_{Ext_i}$ and the reference line is negative, and for values below the intersection is positive. The size of a stair step is itself a differentiable function of $NetIn_i$ since $f^i_{Ext_i}(NetIn_i)$ and $1/w_{ii} \cdot NetIn_i$ are both differentiable, and the stair step size is the difference between those two quantities. Thus stair step size is a smoothly varying quantity. The step size must also approach 0 as $t \to \infty$ and V_i approaches the intersection, and it must always maintain the same sign. Thus V_i must approach the intersection value.

The following result for strongly self-exciting units is presented without proof, since the diagram in Fig. 2, part B, makes the point clear.

Theorem 3.2 *For a strongly self-exciting unit i, when Ext_i is held constant, there are either two stable equilibria (attractors) and an unstable one, one stable and one unstable, or a single stable equilibrium. These situations are illustrated in Fig. 2. All stable equilibria are asymptotically stable.*

There exist two values of Ext_i, β_1 and β_2, at which a strong self-exciter's equilibria bifurcate. For constant $Ext_i \in (-\infty, \beta_1)$, (i.e., when the effective activation sigmoid is far enough to the left) there is a single stable equilibrium at $\frac{1}{w_{ii}} \cdot Int_i = \frac{1}{1+e^{-\lambda(Ext_i + w_{ii}V_i - \theta_i)}} = f^i_{Ext}(w_{ii}V_i) \approx 1$. Similarly, for $Ext_i \in (\beta_2, \infty)$, (i.e., the effective activation sigmoid is far enough to the right) there is a single stable equilibrium at $\frac{1}{w_{ii}} \cdot Int_i = f^i_{Ext}(w_{ii}V_i) \approx 0$.

For $Ext_i \in [\beta_1, \beta_2]$, there is a single unstable equilibrium, and one stable equilibrium at $\frac{1}{w_{ii}} \cdot Int_i = f^i_{Ext}(w_{ii}V_i) \approx 1$, and another stable equilibrium at $\frac{1}{w_{ii}} \cdot Int_i = f^i_{Ext}(w_{ii}V_i) \approx 0$. See Fig. 2. When Ext_i is exactly β_1 or β_2, then there are exactly two intersection points of the reference line and effective activation sigmoid. At the bifurcation point $Ext_i = \beta_1$, there is a stable equilibrium near $V_i = 1$, and an unstable equilibrium closer to $V_i = 0$ resulting from the collision of a stable equilibrium and an unstable equilibrium as Ext_i approaches β_1 from above. The other bifurcation point is similar.

As for the relation between the equilibria of weak and strong self-exciters, when $f^i_\delta(w_{ii} \cdot V_i)$ at the intersection points $f^i_\delta(w_{ii} \cdot V_i) = 1/w_{ii} \cdot V_i$ is plotted as a function of both δ and w_{ii}, a fold catastrophe can be seen to occur at $w_{ii} = 4/\lambda$, which is the value that separates the weak from the strong.

To recap the results of this discussion, the activation of a self-exciter under constant external input always approaches the intersection of the effective activation curve and reference line whenever the slope of the reference line is greater than the slope of the activation curve at the point of their intersection. The behavior of the system therefore depends critically on the ratio of λ to w_{ii} and on θ_i. Weakly self-exciting units have activation curves that intersect the reference line at only one point, because the activation curve has slope $\lambda/4$ at the point of inflection, and for a weak self-exciter, this slope is shallower than that of the reference line. Strong self-exciters may have one, two, or three intersections, depending on the value of θ_i and the level of external input Ext_i.

As discussed in section 2, productions are modeled as feedforward excitation or inhibition projecting from one module to another (connections are allowed to be bidirectional, but the point is that they are asymmetrically weighted in general). The weights on these connections capture preference information, and their effects on a downstream module can be best understood now as shifting the effective activation curves of downstream units. With this formalism, several critical

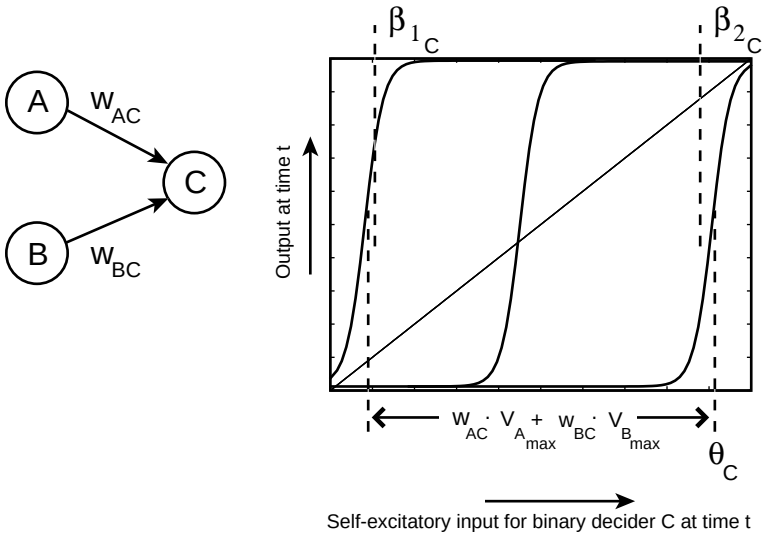


Figure 3: Parameterization for a binary decider, C.

behaviors can be gotten from a module: 1) it can be made to respond strongly to a conjunction of inputs, and then return to baseline when its conjunctive 'feature' is no longer present; 2) it can respond to a feature conjunction, but hold on to its representation indefinitely; 3) it can be made to ramp up slowly toward its attractor value. We discuss special cases of the first two behaviors next.

3.2 Binary Deciders

Consider a system of two upstream units, A and B, and one downstream unit C, with feedforward excitation from A and B to C. It is often useful to have C respond as the neural equivalent of an AND gate in digital logic: when A and B are both highly active, C should be highly active. If either A or B are inactive, C should be inactive. C should never linger at values that are far from 0 or far from 1.

The following parameterizations, illustrated in Fig. 3, give this behavior. C should be strongly self-exciting, and θ_C should be greater than β_{2_C} as defined above. This means that without external input, the effective activation sigmoid intersects C's reference line only once, near $V_C = 0$. The connection strengths from A and B to C should sum to a value sufficient to shift C's activation curve leftward so that its point of inflection occurs at a value less than β_1. Thus when A and B are highly active, C's effective activation curve will have a single intersection with its reference line at a value near $V_C = 1$. Without a drop in A or B, C will eventually become highly active. The time that C takes to become active will depend on how far left of β_{1_C} its activation function is shifted, because this shift determines the size of the first stair steps leading upwards from small values of V_C toward larger values in the cobweb diagram. For left shifts to points arbitrarily close to β_{1_C} and less than β_{1_C}, the cobweb analysis indicates that C's ramp-up time will be arbitrarily long. (Strictly speaking, the time to approach an attractor is always infinite, since there are always infinitely many steps on the way to an attractor. By 'ramp-up time' we refer instead to the time required to approach within some nonzero-diameter neighborhood of an attractor, which happens in finite time.) Thus faster decisionmaking requires the sum of input from A and B to shift C's effective activation sigmoid farther to the left (that is, to excite C more strongly).

With strong $A \rightarrow C$ and $B \rightarrow C$ connections (denoted w_{CA} and w_{CB} respectively), however, there is the potential problem that strong activation in only one of A or B will be sufficient to activate C. In order for C to retain the property of being a conjunction detector, the connection strength from each of A and B to C must be less than $\theta_C - \beta_{1_C}$. Thus a requirement for faster conjunction detection requires both that θ_C be larger and that the feedforward connection strengths be larger and roughly equal, each less than $\theta_C - \beta_{1_C}$, and with sum greater than $\theta_C - \beta_{1_C}$.

Finally, we note that the connection strengths used in this discussion are approximations. The unit A in the simulated inverter example will never quite reach an activation of 1. Weights and θ's must therefore be set with a margin for error that is discovered through trial and error when building a system (in our experience, this is quite easy to do).

3.3 Latches, Gates, Flip-Flops and Sources

In some cases, we may wish to have C act as a conjunction detector as just discussed, but also to maintain an activation-based memory of the conjunction that persists after the conjunction feature disappears. By again relying on strong self-excitation and setting θ_C to be a value in the interval $[\beta_{1_C}, \beta_{2_C}]$, C will maintain a value near 0 or near 1 under small external input. Net excitatory inputs of value greater than $\theta_C - \beta_{1_C}$ will shift the effective sigmoid left past C's 'activation threshold', and if held on sufficiently long for V_C to rise past 0.5, V_C will continue to rise to a value near 1 even after the excitatory input is reduced to 0: that is, it will latch on to a 1. This is because a return to net 0 external input will return the effective activation sigmoid to its true position, which, by equation (1), has its point of inflection centered directly above the value θ_C , and the value of the sigmoid at this point is 0.5. In order to reset C, inhibitory input with absolute value greater than $\beta_{2_C} - \theta_C$ will shift the effective activation sigmoid right to the point that V_C begins to fall off. If inhibitory input is held on until V_C decreases below 0.5, V_C will continue to fall to a value near 0.

C thus has approximately the properties of a latch as it is defined in digital logic design [17]. Latches use feedback to store a bit until reset to store a different value, and thus play a critical role in almost all digitial logic devices that have memory. They may suffer from an undesirable oscillatory behavior called a 'race condition' when their outputs are fed through combinational logic elements back into their inputs [17], thus necessitating the use of a clock pulse. The clock pulse, in combination with a copy of the latch, allows the inputs of a latch to respond to its own outputs only from their values in the duplicate latch at the end of the previous clock pulse. Thus, at the end or beginning of a clock pulse, depending on the particular technology, the output of the latch is deterministic and predictable. Such a clocked latch is termed a flip-flop.

In Polk et al. [28] and the neural model shown in Fig. 7, modules collect and integrate the outputs of other modules in order to compute their own outputs. But the components of a circuit may require the synchronous arrival of signals from multiple upstream modules in order to compute correctly. Timing problems like these are typically handled in digital logic design by buffering values in flip-flops whose values are updated at each clock pulse, or by using AND gates that prevent updating until the arrival of a specific 'enable' signal (which also typically arrives at effectively the same time as a clock pulse). Here, we use a latch/gate pair, the first component of which uses strong self-excitation to buffer its values until inhibited strongly by an external signal. Modules are latches if $\theta_i \in (\beta_{1_i}, \beta_{2_i})$ for every unit i in the module: that is, if the true activation function has two stable attractors and one unstable equilibrium.

We define a *gate* to be a copy of an upstream module (which is typically a latch), with one-to-one feedforward connections from upstream units to their corresponding downstream units. A second

set of inputs to the downstream module is a blanket inhbition of all units that reduces all activity in the gate to baseline. The purpose of a gate is to allow the effective inhibition of a module without wiping out its activation-based memory. This allows a module to vote in downstream elections at only the appropriate times, but allows preservation of the symbolic content of the module itself by isolating it in the meantime from the gating inhibition. We point out that while neural latches are analogous to latches in digital systems, gates as we have defined them are distinctly different than digital components with the same name (AND, NAND, OR gates, etc.). We use this term for our neural component because it is a common term for mechanisms with the same function in the cognitive neuroscience literature [15, 7, 12].

Race conditions somewhat similar to those in digital logic can occur in the neural systems we propose here. The model we discuss below uses the symbolic contents of a latch module called 'Subgoal' to compute a new symbol to replace the old one in Subgoal. This is implemented in a loop which routes output from Subgoal through a feedforward logic circuit back into its own input. But in practice, the process of computing a new subgoal breaks down just as it begins to take effect. Since the logic for computing a new subgoal depends on the identity of the old one, as soon as the new subgoal unit begins to activate, it begins instructing the logic circuit to compute still another new subgoal. To prevent this, an analogue of digital flip-flops is used, consisting of a latch-gate-latch-gate sequence (in Fig. 7, the sequence Subgoal-SubgoalGate-SubgoalBuffer-SubgoalBufferGate). The trigger for a new subgoal computation prevents signal propagation through the loop until the next trigger signal arrives. We discuss the source of that signal in section 5.

We mention one more class of module that proves useful in practice: a source module, in which $\theta < \beta_1$, and in which there is only one unit. Without external inhibition, a source module will ramp up to maximal activation. Sources serve to inject energy into a neural circuit, and are analogous to permanent connections to high voltage sources in digital systems. Their use is illustrated in the model below.

3.4 Hybrid TOL Model

We gave a brief overview of the TOL model in section 2. We now explain how the different types of components just described contribute to it. The structure of the model is illustrated in Fig. 7. Modules are indicated as winner-take-all modules, latches, sources, delays, or any combination of these (except for source/latches, since sources are latches by definition). Delays will be discussed in the next section.

Excluding the influence of goals on behavior, the operation of the model is fairly simple. One Sense module is devoted to each position on the gameboard, with units for representing the values 'red', 'green', 'blue' and 'empty'. A simulated environment clamps these modules to the appropriate values to model perception. Each Sense module is a latch (although constant environmental input makes this unnecessary). The units in the Move module (also a latch) encode possible moves as conjunctions of one of three colors and one of six positions. This 3x6 matrix of move encodings is repeated in several modules in the system. (We note that a permanent, dedicated conjunctive representation of this form is not a scheme that will scale up well to larger problem spaces as it is wasteful of neural resources. The most obvious solution to this problem appears to be online binding of conjunction components, but this is a form of the variable binding problem and is beyond the scope of this paper.)

The representation of the current configuration (the Sense modules) excites all legal moves in the Move module with the same degree of preference and inhibits illegal moves. Constructing connections that encode this excitation and inhibition is a straightforward matter. For example, the unit encoding 'red' in module Sense1 will excite Move units encoding moves of the red ball to

any other position on the board, but will inhibit units encoding moves of any ball to position 1, since that space is occupied. The 'empty' unit in module Sense4 will vote for moves of any ball to position 4, and will do so more strongly than the 'empty' unit in module Sense3.

Binary deciders are used in determining which balls are blocked from moving by balls stacked on top of them. This information requires input from two Sense modules: the Sense module corresponding to the ball that may be blocked, and the module corresponding to the position above it. Thus for each of the blockable positions 2, 4 and 5, there is one module for each of the three colors. For example, if Sense2 represents 'red', then the Red2Block module will be activated to indicate that the red ball may be blocked in position 2. This module excites the RedBlocked module, but not enough to activate it. The 'green' and 'blue' units in Sense3 also excite RedBlocked. RedBlocked is a neural AND gate that only responds to the conjunction of Sense2= 'red' and (Sense3 = 'green' or Sense3 = 'blue').

Moves that involve blocked balls are strongly inhibited by the Blocked modules. The possible legal moves then compete with each other in the Move module via attractor dynamics until one is selected (i.e., convergence to an attractor pattern is detected using a threshold on Euclidean distance between previous Move module activity and current activity - this convergence detection is the element of the hybrid model which is not implemented neurally). Without other sources of input, the move that is finally selected is random and simply depends on noise. The EnergyRegulator module also supplies diffuse excitation to the Move module as part of a negative feedback mechanism that ensures the winner-take-all property of the Move module (that is, if no winner emerges, EnergyRegulator boosts all Move unit excitations, but does so by ramping up slowly enough to ensure that the runner-up representation does not also activate along with the winner).

Once a move is selected, it excites a corresponding unit in the MoveGate module, which is not a latch. The activity of this module can be thought of as a motor command issued to the peripheral motor system. It also serves to extinguish the move selected in the Move module, in order to allow for later move elections to take place, as well as serving to extinguish other short-term memories related to selecting subgoals that are no longer relevant. Once MoveGate is active above a threshold value of activation of 0.9, the simulated environment is updated, causing the representation of the configuration to change accordingly, and the new configuration once again votes for any legal moves. In short, in the absence of goal direction, the model simply performs random search using any moves that are legal in the current configuration.

Goals as they are represented in the system correspond to placing individual balls in specific locations (e.g., getting the blue ball onto the bottom of the third peg). Thus goals are encoded in the same fashion as moves. The base level goals of the system are represented in the latch modules Goal1 through Goal6. The goal configuration of the gameboard is represented there with six latches, as in the Sense system, by initializing each to the color of the ball that occupies the corresponding position. Only one component of this goal configuration is worked on at a time, and this is decided by attractor dynamics within the latch module GoalDecide. Here, weights on the connections from the Goal modules to GoalDecide were easy to set with a fixed preference ordering that guaranteed a unique winner at all times. The preference scheme favors moving balls to lower positions on pegs than to higher positions, since a ball moved to its final position at the bottom of a peg will never have to be moved thereafter to achieve a complete solution. The output of GoalDecide is fed to GoalGate, which attempts in turn to guide Move selection when the system is not already working on some other goal, as we discuss next.

The currently active goal guiding behavior, if any exists, is represented in the latch module labeled 'Subgoal' (the name derives from the fact that this module also represents subgoals generated by impasses in the problem solving process). The Subgoal module modulates processing in the Move module by exciting moves that will achieve the current goal and inhibiting moves that

won't. This modulation biases the competition in the Move module so that moves that will achieve the current goal will tend to be selected. If no legal move will achieve the current goal, then no move is selected (because the current goal will inhibit all legal moves in that case). For some neural systems, it is possible to dispense with feedforward inhibition by manipulating the θ parameter of equation (1) so that patterns not supported by external excitation will fail to activate. Here, however, external inhibition is critical, because the Move module needs to be able to select a move even when the Subgoal module sends no signals (either because Subgoal is damaged, or because it has no information to contribute).

The model also uses latch modules to encode information that is relevant to the current controlling goal, specifically: what is above the ball that the goal refers to ('AboveSource'), the color of the ball, if any, in the target position ('InTarget'), and the lowest free position on the peg that is neither the source nor the target of the current goal ('FreePosition'). This information is crucial for generating subgoals when necessary to get blocking balls out of the way without disrupting progress toward the current goal, and is computed in a manner similar to that of determining which balls are blocked, by using a cascade of binary deciders that receive input from the Subgoal module and the Sense modules.

The selection of a new goal in the Subgoal module can occur in one of two ways. First, if the current goal has been achieved, then it is inhibited by the Sense modules (e.g., Sense1 = 'red' inhibits the goal 'Red to 1'), and the next most important base level goal still unachieved is retrieved from the GoalGate module. This happens because all units in Subgoal inhibit all units in GoalGate, so that GoalGate cannot vote for working on a base level goal once progress has begun on some other goal. Once Subgoal has baseline activation in all units, however, GoalGate is able to become active for long enough to install a new base level goal in Subgoal. Second, if the current goal cannot be directly achieved because of some obstruction (either a ball in the target position or a ball above the ball that we want to move), then a new goal to remove the obstruction will be proposed via input from 'Abovesource', 'InTarget', and 'FreePosition'.

The following algorithm that the hybrid model carries out is a simplified form of that implemented in the hybrid system of Polk et al., which used extra heuristics to deal with difficult cases:

1. set a goal to move a ball to its final goal position (with preference, in decreasing order, for moves to position 4, then 5, then 2, then 6, then 3, then 1)

2. if the current goal is not achieved because some ball is blocking the ball to be moved, set a subgoal to move the blocking ball to the lowest position on the peg which is neither the source nor the target of the current goal - if there are two such pegs, pick one at random - then return to step 2; otherwise, make the desired move;

3. as soon as a move is made, return to step 1.

This algorithm relies on non-neural control code that detects approximate convergence to baseline in the Move module as the trigger for generating a subgoal. Replacing this control functionality with a neural mechanism is the main focus of the next two sections.

4 Open-loop Timing Mechanisms

Open-loop iming is critical in most digital circuits. In 'synchronous' circuits, which currently predominate in hardware design, a series of flip-flops is used to propagate bits of data toward their final targets, one flip-flop per clock pulse, so that they arrive at a logic gate only when it is

appropriate for them to do so. The unvarying duration of individual clock pulses is designed to be long enough to prevent transient, analog voltages within logic gates and flip-flops from affecting computations, and otherwise to be as short as possible. The only way for software running on such systems to measure duration is to count clock pulses. In this section we address open-loop timing in our neural systems and its role in convergence detection and 'glitch' protection. The neural mechanism we propose plays the same role as a digital clock pulse for preventing transients from affecting behavior, but it differs in the way in which cognitive models using it may encode duration. Further, unlike a sequential circuit clock pulse which controls the rest of the circuit, the mechanism we propose is selectively enabled or disabled by the circuit, which can operate without it under some conditions.

4.1 Measuring and Encoding Duration

As in digital systems, counting is still an option for continuous systems to measure duration, because oscillators can be used to implement a clock pulse with a specific frequency. Continuous approximations to finite automata can then be used to count pulse sequence length, or some continuous quantity can be increased by each pulse in order to measure elapsed time.

Other options exist in real nervous systems. At very fast time scales (tens of milliseconds or less), the relative spacing of individual action potentials can be used to measure time [19]. Precise propagation delays along axons can be used, for example, to measure the arrival time of an auditory signal from the left ear against that of a signal from the right ear in the auditory cortex of a barn owl in order to localize a sound source with great accuracy [8]. At longer time scales, however, some other method is necessary. Counting action potentials is a possibility, as is counting lower-frequency oscillations of a population's recent average firing rate. Non-counting methods for measuring and encoding duration include the use of a monotonically increasing quantity, or 'ramp', that reaches a threshold value at the end of an interval. This is arguably the simplest non-counting method possible.

We choose a combination of ramps and counting for modeling the timing of durations ranging from roughly half a second to minutes (the time scale of cognitive operations [26]) for the following reason: if we were to count simulated action potentials with finite automata, intervals of many seconds would require automata with thousands of states or else depend on neurons with extremely low firing rates. If the oscillator cycles occurred on a slower time scale than individual spikes, then they would most likely either have ramps already built in to each cycle or be 'relaxation oscillations' with abrupt transitions between otherwise slowly changing values [23]. A brief examination of the positive feedback analysis of section 2 shows that we already have ramps for free, and that their durations can be easily modulated through a simple change in synaptic strength. Further, a combination of counting with these methods is straightforward to implement with a sequence of ramps, each one of which times the duration of one 'click' and initiates the subsequent ramp. We refer to a sequence of modules that implements this ramp sequence as a timer.

One means for duration encoding with synaptic weights is illustrated by the following system of three units, shown in Fig. 4. A latch unit labeled 'Start' that encodes a start signal sends excitation to a second, weakly self-exciting unit, the Ramp unit, which in turn sends excitation to a latch unit labeled 'Finish'. The self excitation strength of Ramp should be nearly $4/\lambda$, so that the reference line has approximately the same slope as the activation sigmoid at its point of inflection. The 'bias' term θ_{Ramp} is set so that without external excitation, the inflection point of the true activation sigmoid is to the right of the reference line and the single intersection of the ramp unit's reference line and its sigmoid is near 0. The Start $\rightarrow$ Ramp connection strength should be such that when Start is highly active, Ramp's effective activation sigmoid is shifted left so that its inflection point

is to the left of the reference line. In this case, Ramp will ramp up to a high activity at a rate that is determined by the size of the first stair steps in the cobweb diagram formed after Start excitation. The connection strength from Ramp to Finish should be such that Finish is not activated except by maximal Ramp activation. The effect is that information from Start propagates to Finish after a delay imposed by Ramp. The length of that delay depends on the size of the first stair steps in Ramp's cobweb diagram. Larger initial steps occur when Ramp's effective activation curve sits farther to the left, implying faster increase. Thus the connection strength from Start to Ramp encodes the propagation delay. The time course of activation for this system, shown in Fig. 4, parts A & B, shows that Ramp gets its name from its gradual, nearly linear increase in strength.

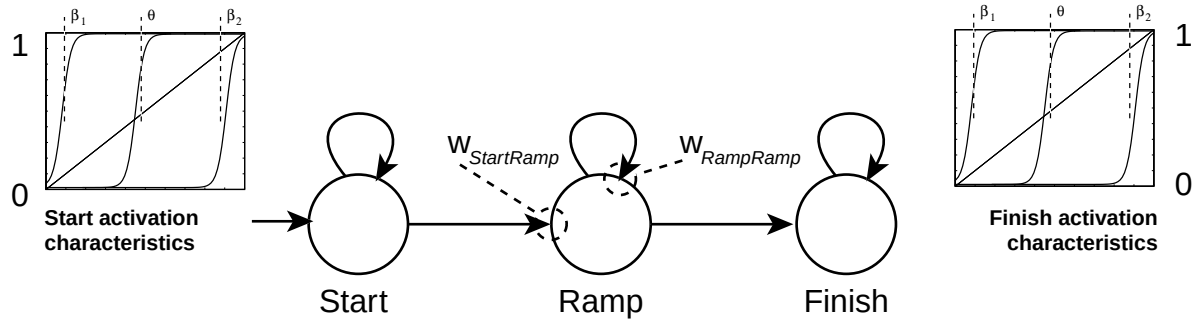
The scheme can be extended to one in which Ramp is a strong self-exciter, as in Jones [22]. In this case, the Start $\rightarrow$ Ramp connection strength should be such that Ramp's effective activation curve is shifted leftward slightly more than $\theta_{Ramp} - \beta_{1_{Ramp}}$. If Ramp is at a low activation when this occurs, a bottleneck appears at the beginning of Ramp's increase, which is eventually escaped, producing a timecourse that looks somewhat like a two-component linear ramp. See Fig. 4, parts C & D.

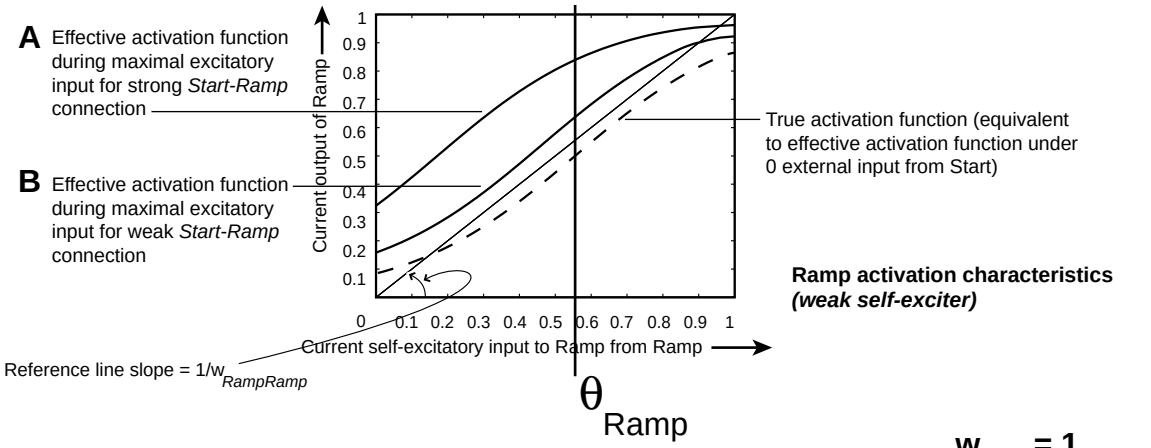
At this point we should note that the ramp-up dynamics depend in the first case on at least part of the actual activation function being approximately linear. In the second case, the dynamics depend on it having a gradual saturation nonlinearity, or knee shape, at the bottom. Neither assumption requires the particular form of equation (1), but any activation function which has neither property is unsuitable for this form of timing. Finally, we note that we can also chain several ramp units in sequence to create a delay circuit that performs more like counting. Such a scheme allows us to avoid relying on arbitrarily precise connection strengths for timing very long intervals, which otherwise would require the proximity of the Ramp sigmoid to its reference line to be arbitrarily small. Any noise in the system would then disrupt the timing mechanism severely.

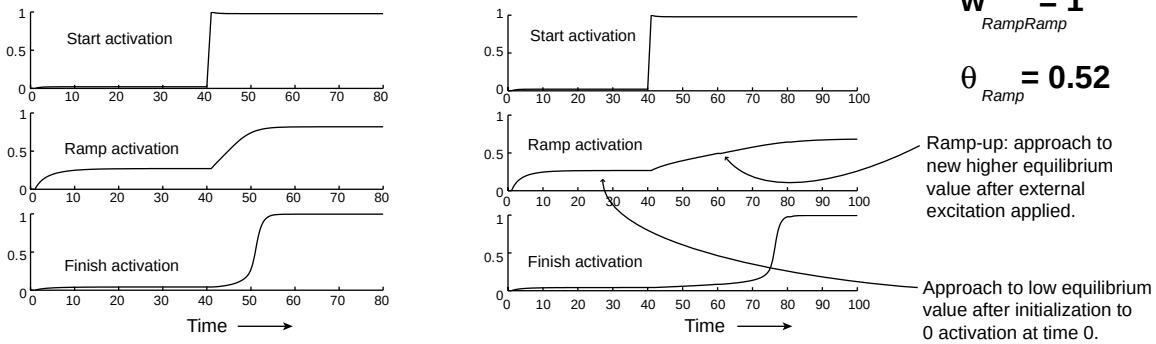
4.2 Glitches and Glitch Protection

We noted in our discussion of the role of clock pulses in digital sequential circuits that a critical function of the clock pulse is to prevent transient voltages from propagating through the system in a way that causes erroneous computation. Such transients are called 'glitches', and while they tend to dissipate within a system because of time delays imposed by individual logic components, it is possible for them to disrupt computation to the point that the outputs of a system are incorrect [17]. A glitch-propagation problem similar to that of digital circuits can occur in our neural systems.

Consider the associative memory system illustrated in Fig. 5. In this system, three latch units, A, B and C, form an associative memory storing the patterns $AB\neg C$ and $\neg AB\neg C$. The notation $AB\neg C$ stands for patterns of activity in which $V_A \approx 1, V_B \approx 1, V_C \approx 0$. Whenever V_C is set to a value near 1, B will become active, which in turn deactivates C. Whenever A is active, C becomes active, and the previous process repeats. Thus for the input patterns $A\neg B\neg C$, $AB\neg C$, $A\neg BC$ and $A\neg B\neg C$, the associated stored pattern $AB\neg C$ is retrieved. For patterns $\neg ABC$, $\neg A\neg BC$, and $\neg AB\neg C$, the pattern $\neg AB\neg C$ is retrieved. For input pattern $\neg A\neg B\neg C$, that pattern itself, corresponding to baseline activation in all units, is retrieved. For input pattern $A\neg B\neg C$ however, there is transient activation of the pattern $A\neg BC$, which is not a stored pattern, before $AB\neg C$ is retrieved.

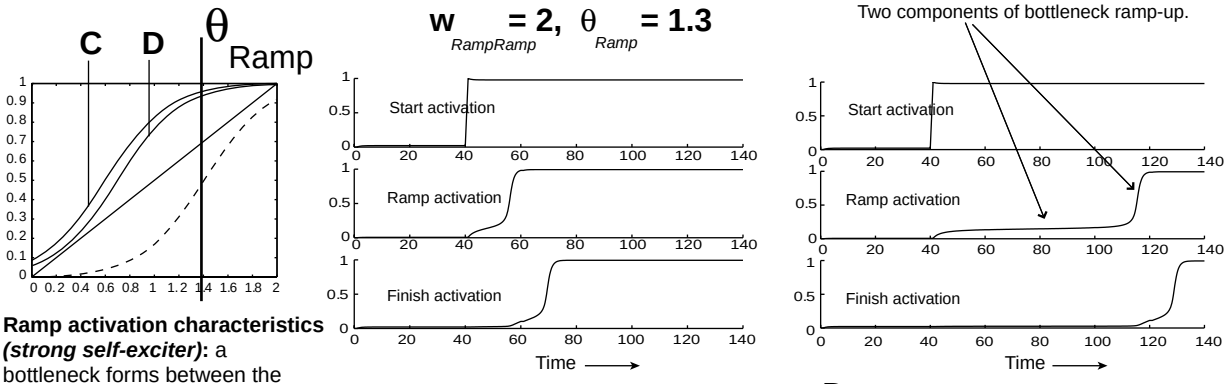






A *StartRamp* connection weight = 0.08. Start is set to 1 at time 40.

B *StartRamp* connection weight = 0.03. Start is set to 1 at time 40.



Ramp activation characteristics *(strong self-exciter)*: a bottleneck forms between the reference line and activation function to produce delay

C *StartRamp* connection weight = 0.6. Start is set to 1 at time 40.

D *StartRamp* connection weight = 0.58. Start is set to 1 at time 40.

Figure 4: The effects of Ramp unit parameterizations on propagation delay.

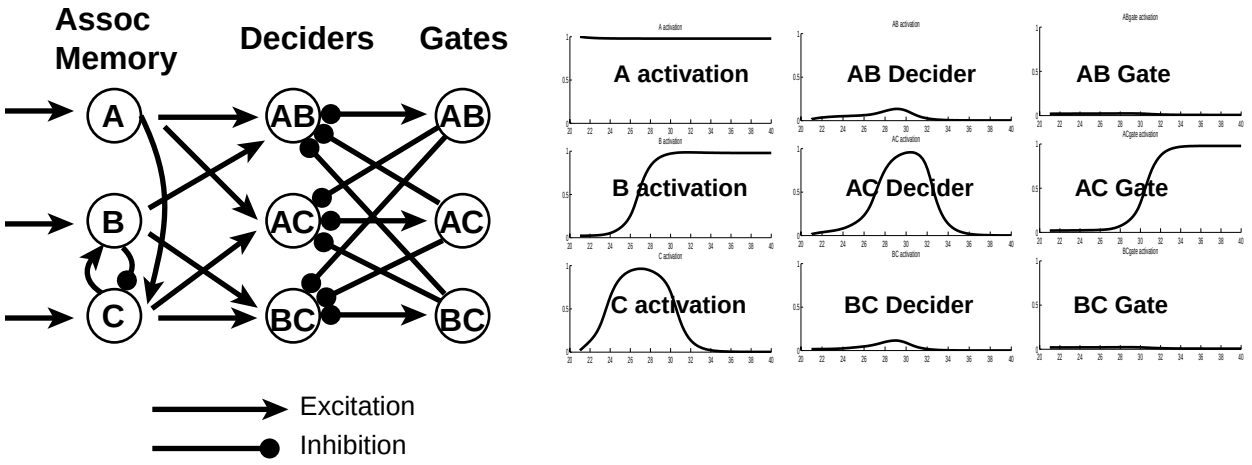


Figure 5: A glitch: premature propagation of the computation of an attractor module.

Now consider a set of three binary deciders that detects conjunctions of two features from this memory: AB, BC, or AC. Considering the associative memory in isolation, it may seem that there is no need for detection of conjunctions involving C, since C can only be active transiently, but keep in mind that external inputs to the associative memory can overwhelm internal mutual inhibition and force retrieval of C. The deciders pass their computations on to a module that initiates one of a set of processes depending on the identity of the retrieved conjunction. We make the assumption that a process, once begun, may continue after the disappearance of the triggering stimulus, and may sometimes need to continue to completion before responding to the next stimulus. A case like this is illustrated in the convergence detection circuit discussed in section 5. To model such a case, the binary deciders are inhibited by the latch supporting the response process. Thus we have the circuit shown in Fig. 5. In this case, fast propagation of information from the associative memory to the binary decider results in a response process appropriate for stimulus conjunction $A\neg BC$, rather than the proper response to the equilibrium pattern $AB\neg C$, and once initiated, the system becomes insensitive to the proper conjunction. This is an example of a glitch resulting in erroneous computation. Sophisticated response-process interrupt circuitry could be developed for cases like this, but a simpler solution is simply to delay propagation of the associative memory signal by an interval sufficient to let the associative memory settle into equilibrium. This is precisely what happens in digital logic, where the clock pulse effectively forces a long enough delay period within each component for equilibrium to occur [17].

For this purpose, we propose a propagation delay device with a modifiable delay which we build in to the structure of all modules. However, unlike the equivalent situation in digital circuits, we do not attempt to calculate or enforce a delay period sufficient for glitch protection in all cases. Convergence time in a random associative memory is difficult to predict in general, and any fixed time delay is liable to force the system to slow down unnecessarily in many situations. Furthermore, glitches may very likely play a role in modeling human errors in psychological tasks. We therefore use a trial-and-error process of setting delay characteristics for each module individually (this was easy to do in constructing the TOL model, since most modules required no delay at all).

4.3 Delay in Columnar Networks

A distinctly columnar approach to neural network construction has appeared in work that uses structured columns in the visual system to perform useful computations [16] as well as work that

demonstrates the ability of visual cortex to self-organize into columns [11]. Wilson & Cowan [35] show that different areas of cortex may be modeled by equations similar to (1) and an extremely simple columnar structure can give rise to forms of activity characteristic of prefrontal, primary visual and thalamic neurons. There has also been widespread use of model frontostriatal loops in neural networks [15, 13, 6, 21], in much of which the model loop circuits are topologically similar to columns.

We use the column concept to address our need for a means of encoding variable propagation delays within modules. The columnar version of a module is schematically depicted in Fig. 6. Instead of a single, fully connected recurrent network, a module now consists of two identical copies of such a network. One copy functions as the input interface to the module, and the other functions as the output interface. Each input layer unit $Input_i$ sends a feedforward excitatory connection to its counterpart $Output_i$ in the output layer. The $Output_i$ unit is inhibited by a self-exciting unit $Delay_i$ that also receives inhibitory input from $Input_i$. The $Delay_i$ unit serves to prevent rapid transmission to $Output_i$ of large jumps in the activation of $Input_i$. The rate of transmission from $Input_i$ to $Output_i$ is determined by how strongly $Input_i$ inhibits $Delay_i$. Variable delay characteristics derive from the fact that strong inhibition from *Input* shifts the $Delay_i$ effective activation sigmoid to the right. Typically, θ_{Delay_i} causes the activation curve to sit far enough to the left so that it has a single intersection with the reference line very near $V_{ZeroLatch_i} = 1$ when $Input_i \approx 0$. When $Input_i$ sends inhibition to $Delay_i$, that shifts it back rightward. In the case of a weakly self-exciting $Delay_i$, the shift should be sufficient to move the inflection point of the curve to a position close to, but to the right of, the reference line (see Fig. 2, part A). In the case of a strongly self-exciting *Delay*, the shift should be sufficient to shift the effective activation sigmoid by an amount slightly greater than $\beta_{2_{Delay}} - \theta_{Delay}$. In both cases, a large number of small stair steps near the beginning of the trajectory indicates slow decay of $Delay_i$, and therefore slow transmission of changes from low activation to high activation in $Input_i$ to $Output_i$. Slow transmission of high to low changes in *Input* can be accomplished by making sure that the true activation sigmoid for *Delay* also sits close to the reference line.

5 Neural Convergence Detection and Subgoal Generation

The only remaining issue not addressed in section 3 in constructing a fully neural cognitive model of the TOL task was the need to detect convergence to equilibrium of the Move module. The hybrid model in Polk et al. simply waits until a Move unit is active above some threshold and is relatively unchanging in value, and until all other units are inactive before it considers a move to have been selected. If no move is selected after a duration exceeding some threshold, the symbolic control code determines that the model has reached an impasse in the solution process, and it generates a subgoal to move a blocking ball out of the way (a blocking ball is the only source of impasses in this task). When multiple moves emerged in the Move module, which was rarely, the control code simply reinitialized the Move module and let a new election take place.

In cases in which the Subgoal module is inhibiting moves that would not help achieve the current goal, connection strengths are such that the Move module will never select a winner. In this case, a timer circuit will be triggered by lack of activity in the Move module. This is the purpose of the 'NoMove' timer chain in Fig. 7.

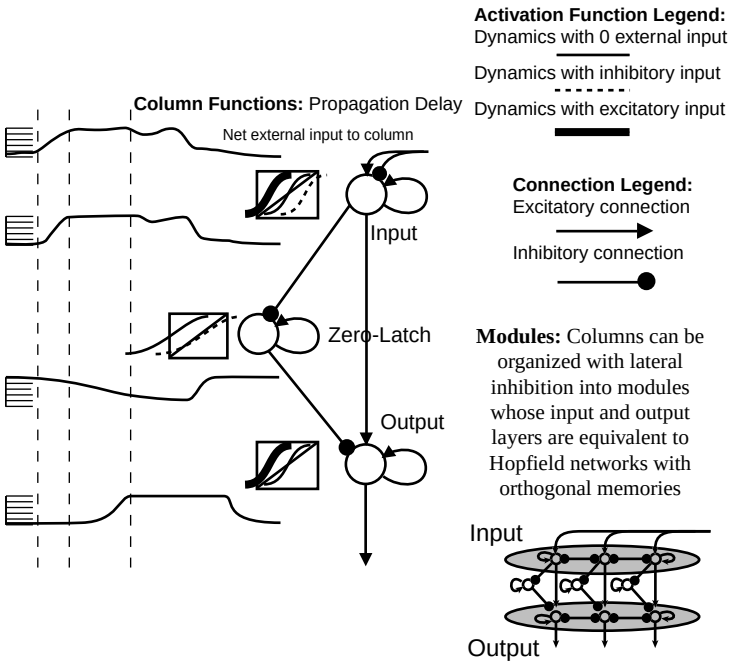


Figure 6: A model cortical column.

The NoMove timer chain starts with a single-unit source module that has very small θ, so that it will tend to activate without external inhibition. It ramps up slowly, and when it finally activates, it prevents any move from emerging and begins a timed process for generating a subgoal. The purpose of this recurrent inhibition is to make the decision to generate a subgoal final, and to stop any later Move activity that would disrupt the generation process or cause moves that would make the ultimately generated subgoal inappropriate for the resulting configuration of the gameboard.

Once activation reaches the final timer module in the NoMove timer chain, the Generate module is activated. This module allows information about the current goal to flow through SubgoalBuffer-Gate and chokes off information about the current goal from flowing through SubgoalGate. The purpose of this is to prevent a race condition as discussed in section 3, and therefore the Generate signal should be seen as the analogue of a single digital clock pulse for the neural flip-flop formed by the chain of modules: Subgoal, SubgoalGate, SubgoalBuffer, SubgoalBufferGate. At this point, a cascade of binary deciders uses information about the current subgoal and the current Sense configuration to determine which ball, if any, is above the ball to be moved under the current goal, which ball is in the target position of the ball to be moved, and which peg is able to hold moved blockers. These three values then vote for appropriate subgoals in the Subgoal module.

The successful generation of a subgoal will be signalled by the elimination of the old Subgoal pattern, followed by the activation of a new one. This process is detected in the sequence of modules Compare and GenerateSuccess. Compare detects activation in Subgoal, but only after it has been enabled by CompareEnable. CompareEnable itself cannot be active until the first goal pattern in Subgoal has been extinguished. This subgoal generation process is ended as soon as a successful generation is detected by Compare, which simply responds to any significant activation in the Subgoal module. If a second subgoal-selection timer expires, then this indicates that there is no appropriate subgoal to generate. In this case, one of the heuristics implemented in Polk et

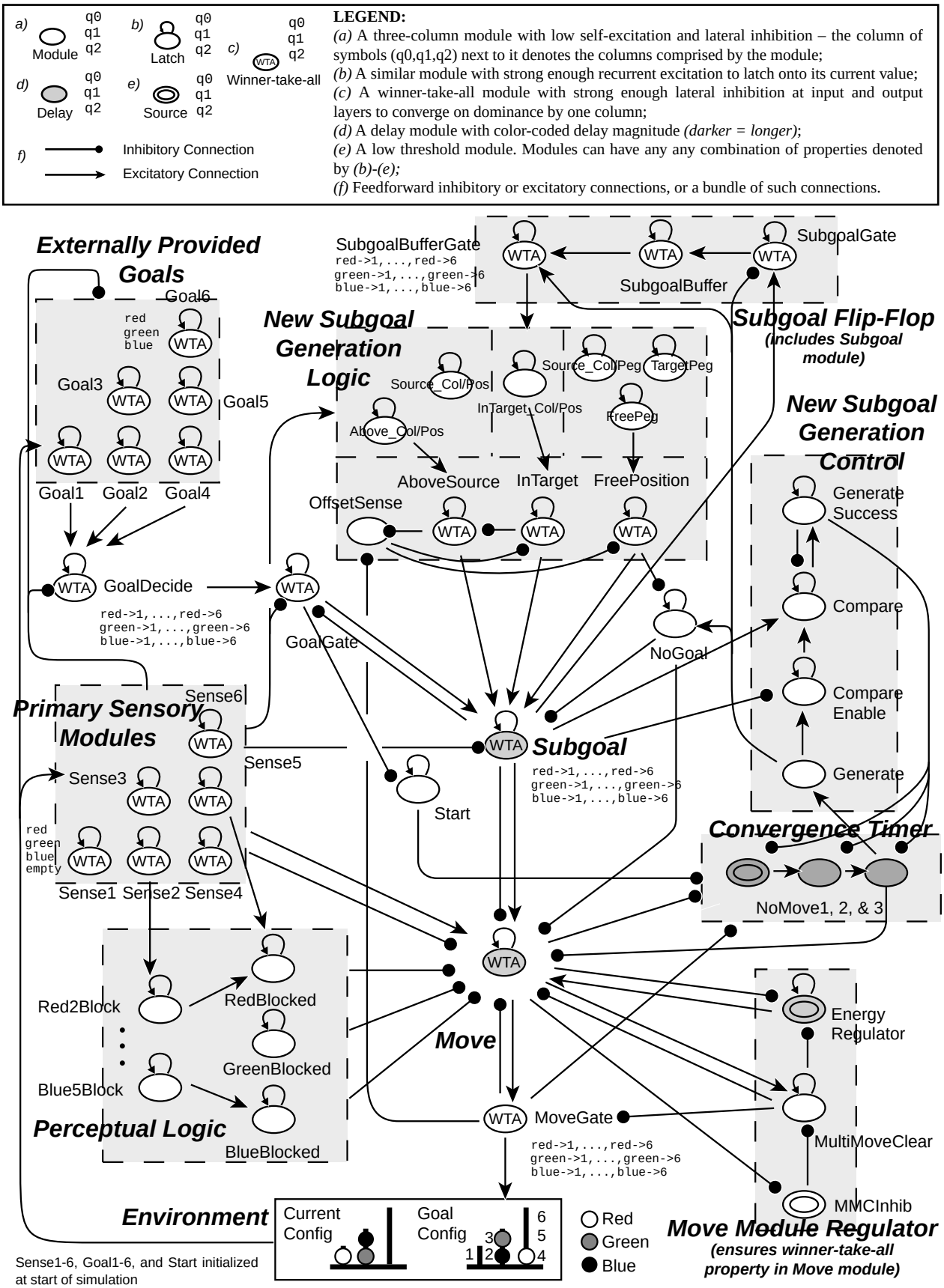


Figure 7: A completely neural model of the Tower of London task.

al. [28] should be used to select a subgoal according to different logic than that encoded in the binary decider cascade. However, at the time of writing, we have not yet completed the neural implementation of this heuristic (which is to try to move the ball to the shortest peg instead of its previous target - due to the constraints imposed by the peg lengths, moving a ball to the first peg is typically the optimal step in getting it to its ultimate destination after an impasse).

With these mechanisms, the symbolic components of the model described in section 3 can be replaced, resulting in a completely neural problem solver.

6 Performance of the Model

Here we show the performance of the model in one problem, simulated in Matlab with the Runge-Kutta(4,5) ordinary differential equation solver. Timecourses of activation in most of the model's components are shown in Fig. 8. The problem, shown at the bottom of Fig. 7, requires five moves for solution and therefore requires that some balls be moved to positions other than their final, goal positions. Thus it requires the internal generation of subgoals for efficient solution. The GoalDecide module can be seen to hold an election for the first goal to control behavior. The goal 'Red to 4', which is the most important unachieved goal according to the preference scheme for moving balls to the lowest possible positions, wins at approximately time 25 (labeled A in the figure), and this information is transmitted through the GoalGate module to the Subgoal module, which responds to it at label B. (In the meantime, in order to prevent the premature generation of a subgoal in response to Move module inactivity, the Start module inhibits the NoMove timer system.) The Sense modules, like the Goal modules, are initialized at the beginning of the simulation and excite potentially legal moves at the same time as the Blocked modules compute which balls are blocked. Finally, a winner, 'Red to 4' is selected at time point C, and the corresponding unit in MoveGate is caused to rise to threshold, achieving the move and wiping out the move-generating command in Move. At this point, the simulated environment causes an update of the Sense modules (point D), which in turn extinguish any goal or subgoal activation pattern in the Goal system or Subgoal which represent goals to create the current environmental configuration (point E). This allows the next most preferred goal to be retrieved and worked on, as can be seen in Subgoal at point F. At no point is the clock circuit involved.

Now the next goal, 'Blue to 2', which is unachievable, has been selected, and this in turn generates a subgoal to remove an obstacle. Once a subgoal is selected ('Green to 5', since Green is in the target position of the blue ball, at time point G), the first element of the NoMove timer sequence begins to ramp up, and finally maximal activation reaches the last timer in the sequence at time H (this also happens for the previous goal). This activates the Generate module for generating a subgoal. This in turn enables testing for generation success by activating CompareEnable while also allowing the information about the current goal to filter into the subgoal computation modules through SubgoalBufferGate, at time I. Generate activation chokes off SubgoalGate so that information about the new subgoal cannot propagate until the next generation process, at time J, and also wipes out the current Subgoal pattern. Finally, the subgoal generation logic computes that the ball above the green source ball is blue, at time K, and that the lowest position on a peg which is neither the source nor the target of the goal is position 5 at time L, and Subgoal responds to this voting at time M. The model continues on in this way until eventually solving the problem in 5 moves, as is shown in the sequence of gameboard configurations selected by the model.

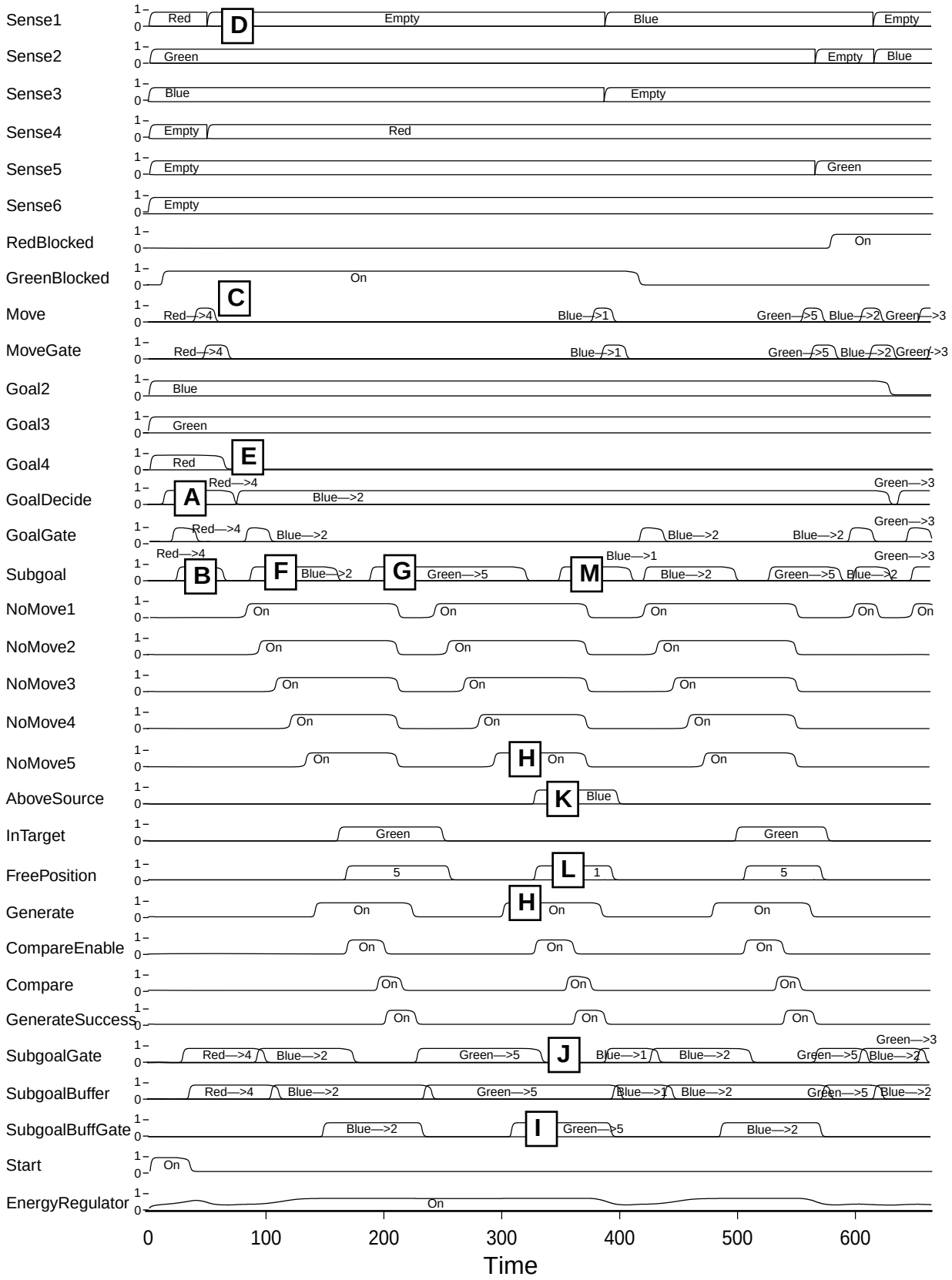


Figure 8: Time courses of activation in most modules of the neural Tower of London problem solver.

7 Conclusion

We obtain continuous neural systems compatible with the principles of Polk et al. [28] that also have the required convergence detection and sequencing functions by employing neural mechanisms at five levels of hierarchical composition. At the lowest level is the firing rate model of a unit or population given in eq. 1. At the next level is the column mechanism disscussed in section 4. Columns are themselves composed through lateral, inhibitory connections into modules, which are in many respects identical to the attractor networks used in Polk et al [28]. Columns turn out to be essential for controlling the rate of signal propagation in order to support sequencing and convergence detection. Modules can themselves be composed with feedforward connections to implement a neural form of productions. The highest layer of organizational abstraction allows construction of complete cognitive models with characteristic combinations of storage and gating modules. All of these mechanisms are consistent with known cortical and subcortical organization, since they require only structured vertical arrangements of densely connected neurons with lateral connections to other columns, along with long-distance, vertical projection axons [34].

A host of issues remains unaddressed in this paper. For example, how do these complex circuits self-organize? How do they learn what the appropriate delays are? How do they acquire a notion of goals online? How do they generate plans which can be stored, and then executed only at a later time? Can they implement goal stack structures?

We conclude with speculations on some issues, and pointers to previous work on others. As for self-organization, preliminary work suggests that Hebbian learning is the key [18], and that the column structure automatically provides a means for sequential organization if the *Output* unit of one column can be held on at the same time as the *Input* unit of another column. A form of Hebbian learning may be suitable for learning the *Input* to *Delay* connection strength within a module in order to acquire a duration encoding. Much research into the neural basis for reward prediction and anticipation suggests a prominent role for temporal difference learning methods in acquiring reward predictors [29, 25], which may provide a means for learning what goals ought to be in new tasks. Finally, we have implemented goal stack structures [31] and list structures [32] in other networks of this type, and in future work we intend to incorporate them in the architecture described here.

References

[1] S. Amari. Dynamics of pattern formation in lateral-inhibition type neural fields. *Biological Cybernetics*, 1977.

[2] S. Amari. Neural theory of association and concept-formation. *Biological Cybernetics*, 1977.

[3] J. Anderson and C. Lebiere. *The atomic components of thought.* Lawrence-Erlbaum Associates, 1998.

[4] J. A. Anderson. Cognitive and psychological computation with neural models. *IEEE Transactions on Systems, Man and Cybernetics*, 1983.

[5] M. Arbib. *The Handbook of Brain Theory and Neural Networks.* MIT Press, 2003.

[6] G. S. Berns and T. J. Sejnowski. A computational model of how the basal ganglia produce sequences. *Journal of Cognitive Neuroscience*, 1998.

[7] T. S. Braver, D. M. Barch, and J. D. Cohen. Cognition and control in schizophrenia: a computational model of dopamine and prefrontal function. *Biological Psychiatry*, 2000.

[8] C. E. Carr and M. Konishi. Axonal delay lines for time measurement in the owls brainstem. *Proc Natl Acad Sci USA*, 1988.

[9] M. Cohen and S. Grossberg. Absolute stability of global pattern formation and parallel memory storage by competitive neural networks. *IEEE Transactions on Systems, Man and Cybernetics*, 1983.

[10] S. Dehaene and J. Changeux. A hierarchical neuronal network for planning behavior. *Proceedings of the National Academy of Science, USA*, 1997.

[11] Ch. Von der Malsburg. Self-organization of orientation sensitive cells in the striate cortex. *Kybernetik*, 1973.

[12] M. Djurfeldt, O. Ekeberg, and A. Graybiel. Cortex-basal ganglia interaction and attractor states. *Neurocomputing*, 2001.

[13] P. F. Dominey and M. A. Arbib. A cortico-subcortical model for generation of spatially accurate sequential saccades. *Cerebral Cortex*, 1992.

[14] J. A. Feldman and D. H. Ballard. Connectionist models and their properties. *Cognitive Science*, 1982.

[15] M. J. Frank, B. Loughry, and R. C. O'Reilly. Interactions between frontal cortex and basal ganglia in working memory: a computational model. *Cognitive, Affective and Behavioral Neuroscience*, 2001.

[16] S. Grossberg. How does the cerebral cortex work? development, learning, attention, and 3d vision by laminar circuits of visual cortex. *Behavioral and Cognitive Neuroscience Reviews*, in press.

[17] J. Hayes. *Introduction to digital logic design.* Addison-Wesley, 1993.

[18] D. O. Hebb. *The organization of behavior.* Wiley, 1949.

[19] J. J. Hopfield and C. D. Brody. What is a moment? transient synchrony as a collective mechanism for spatiotemporal integration. *Proceedings of the National Academy of Sciences, USA*, 2000.

[20] J.J. Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences, USA*, 1984.

[21] J.C. Houk and S.P. Wise. Distributed modular architectures linking basal ganglia, cerebellum and cerebral cortex: their role in planning and controlling action. *Cerebral Cortex*, 1995.

[22] M. Jones. *Temporal information and adaptive rationality.* PhD thesis, University of Michigan, 2003.

[23] D. W. Jordan and P. Smith. *Nonlinear Ordinary Differential Equations.* Oxford University Press, 1999.

[24] R. Lachman, J. Lachman, and E. Butterfield. *Cognitive Psychology and Information Processing: An Introduction.* Lawrence Erlbaum Associates, 1979.

[25] P. R. Montague, P. Dayan, and T. J. Sejnowski. A framework for mesencephalic dopamine systems based on predictive hebbian learning. *Journal of Neuroscience*, 1996.

[26] A. Newell. *Unified Theories of Cognition.* Harvard University Press, 1990.

[27] R. O'Reilly and Y. Munakata. *Computational Explorations in Cognitive Neuroscience.* MIT Press, 2000.

[28] T. A. Polk, P. A. Simen, R. L. Lewis, and E. G. Freedman. A computational approach to control in complex cognition. *Cognitive Brain Research*, 2002.

[29] W. Schultz, P. Apicella, E. Scarnati, and T. Ljungberg. Neuronal activity in monkey ventral striatum related to the expectation of reward. *Journal of Neuroscience*, 1992.

[30] T. Shallice. Specific impairments in planning. *Philosophical Transactions of the Royal Society of London, Ser. B*, 1982.

[31] P. A. Simen, T. A. Polk, R. L. Lewis, and E. G. Freedman. Goal management in a recurrent neural network. In *Proceedings of the 6th Joint Conference on Information Sciences, 2002.*

[32] P. A. Simen, T. A. Polk, R. L. Lewis, and E. G. Freedman. Universal computation by networks of model cortical columns. In *Proceedings of the International Joint Conference on Neural Networks, 2003.*

[33] G. Ward and A. Allport. Planning and problem-solving using the five-disc tower of london task. *Q. J. Exp. Psychol. Sect. a: Hum. Exp. Psychol.*, 1997.

[34] E.L. White. *Cortical circuits: Synaptic organization of the cerebral cortex, structure, function, and theory.* Birkhauser, 1989.

[35] H. R. Wilson and J. D. Cowan. A mathematical theory of the functional dynamics of cortical and thalamic nervous tissue. *Kybernetik*, 1973.