Supporting Appendix

Here we give a detailed description of our model and result. The sections are organized as follows: (1) formal definitions of graphs, circuits, and task; (2) formal theorem, according to which internal conflict exists in the boundedly optimal circuit; (3) a high level plan of proof, giving the proof idea mainly in verbal form; (4) formal proof for fanout-free circuits; (5) extension of the proof to limited-sharing circuits; (6) concluding remarks.

1. Definitions

1.1 Graphs. Let a directed graph $G = (V_G, E_G)$ be a set V_G of nodes and a set $E_G \subseteq V_G \times V_G$ of directed edges (ordered pairs of nodes).

A graph is strongly connected if there exists a path from any origin to any target node consisting of a sequence of appropriately directed edges.

A graph is reflexive if each node has a loop, i.e., an edge directed from that node to itself.

Consider G_i graphs where $i \in \{1,...,a\}$. Let G_i's nodes be V_i and edges be $E_i = \bigcup_{v \in V_i} \{v\} \times W_{i,v}$, where $W_{i,v}$ is the set of nodes that have incoming edges from v in G_i. The product graph $G' = G_1 \times \cdots \times G_a$ has nodes $V = V_1 \times \cdots \times V_a$ and edges:

$$E = \bigcup_{(v_1,\ldots,v_a) \in V} \{(v_1,\ldots,v_a)\} \times (W_{1,v_1} \times \ldots \times W_{a,v_a}).$$

We will call $G_1,\ldots,G_a$ the factor graphs of G'.

We will consider graphs in which the number of nodes is an integral power of 2. Within each graph, each node will be labeled with a unique sequence of Boolean values. Furthermore, a node in a product graph G' will be labeled by a concatenation of the labels of its constituents in the factor graphs of G'. Namely, if the nodes $v_1,\ldots,v_a$ taken from the factor graphs $G_1,\ldots,G_a$, respectively, are labeled by the Boolean sequences $l_1,\ldots,l_a$, respectively, then their corresponding node in the product graph G' will be labeled by the Boolean sequence $l_1 l_2 l_3 \ldots l_a$.

1.2 Circuits. We consider here acyclic interconnections of input terminals, gates, and output terminals by means of wires. Each gate computes a function of the

signals on the wires directed into it from input terminals or other gates and places the result on each of the wires directed out of it to other gates or output terminals. As a result, signals are produced on the output terminals, which represent the circuit's response to its inputs. An example of a very simple circuit is given in Figure 1*a*.

We consider circuits in which the gates compute functions that are drawn from a finite set (basis) $\mathcal{B}$ that is complete (meaning that every Boolean function can be computed by some circuit whose gates compute functions from $\mathcal{B}$) and closed (meaning that the result of substituting either Boolean constant for any argument of any function in $\mathcal{B}$ is another function in $\mathcal{B}$). We let wires carry Boolean (two-valued) signals and let gates compute Boolean functions, although generalization to *k*-valued signals and functions is straightforward. We consider both fanout-free circuits (also called formulas), where only one wire comes out of each gate (although any number of wires may be connected to any input terminal), as well as limited-sharing circuits, where the number of wires coming out of each gate is unlimited, but the extent of merging and diverging of paths (sequences of interconnected gates) is limited (see section 5 for more detail).

We assume that there is a function Λ that assigns a cost $\Lambda(b)$ to each function $b \in \mathcal{B}$. The cost of a gate is then defined to be the cost of the function computed by that gate. We also assume that the cost of a function that results from substituting a constant for an argument of a function $b \in \mathcal{B}$ is strictly less than $\Lambda(b)$. This reflects the physical basis of computation, i.e., a function of fewer variables is easier to implement. We define the cost (equivalently, computational complexity) $L(\mathcal{C})$ of a circuit $\mathcal{C}$ to be the sum of $\Lambda(\gamma)$ over all gates γ in $\mathcal{C}$. If $\mathcal{F}$ is a set of Boolean functions, we denote by $L(\mathcal{F})$ the minimum possible cost of a fanout-free circuit (respectively, limited-sharing circuit) computing all of the functions in $\mathcal{F}$.

1.3 Task. We intend to build a circuit for finding shortest-paths on graph *G* (the task environment). The circuit will serve as a brain for the robot described in the main text of this article. At any time step, the circuit will be presented with a pair of current and target node-labels as a sequence of Boolean input values on its input terminals and will be required to produce in response the label of the next node to step onto as a sequence of Boolean output values on its output terminals.

Formally, let $S^t = (c, d)$ be the state of the world at the beginning of time interval *t*, where *c* is the robot's current location and *d* is the target (on graph *G*). The robot will be tested for each pair of origin and target nodes. Every test begins with $t = 0$, c = origin, and d = target. Let $I^t = g(S^t)$ be the inputs into the circuit at the beginning of time interval *t* and $\hat{O}^t = F(I^t)$ be the outputs of the circuit at the end of that time interval. Here $g(S^t) = S^t$, although later we will use $g_i(S^t) \neq S^t$. Finally, $S^{t+1} = h\left(S^t, \hat{O}^t\right)$, where

$$h\left((c,d), \hat{O}^t\right) = \begin{cases} (\hat{O}^t, d) \ \textit{if} \ (c, \hat{O}^t) \in E_G \\ (c,d) \ \textit{otherwise.} \end{cases} \qquad \textbf{[A.1]}$$

That is, the robot moves to the next node named by the circuit, unless there is no edge connecting to that next node from the current location, in which case the robot stays put.

1.4 Performance. Given a reflexive and strongly connected graph $\mathcal{G}$ whose $p = 2^m$ nodes are labeled with binary strings of length m and a circuit $\mathcal{C}$ with $2m$ inputs and m outputs, let the score $P(\mathcal{C}, \mathcal{G})$ of $\mathcal{C}$ in $\mathcal{G}$ be the number of origin-target node-pairs (x, y) in $\mathcal{G}$ such that the circuit $\mathcal{C}$, when receiving the labels of (x, y) at its inputs, produces at its outputs the label of a node *z* that immediately follows *x* on some shortest path from *x* to *y* in $\mathcal{G}$ (or produces the label of *x* if $x = y$). (We do not assume that shortest paths in $\mathcal{G}$ are unique.) We have $0 \leq P(\mathcal{C}, \mathcal{G}) \leq p^2$. We say that $\mathcal{C}$ is perfect for $\mathcal{G}$ if $P(\mathcal{C}, \mathcal{G}) = p^2$. We observe that if $\mathcal{G}$ has unique shortest paths, then all circuits that are perfect for $\mathcal{G}$ compute the same set of functions.

The score is a tool for finding desirable circuits; i.e., a circuit perfect for *G* always finds the shortest path from any origin to any target on *G* when placed inside the robot described above. For such a circuit,

$$\hat{O}^t = \arg\max_{O^t} U\left(g\left(h\left(S^t, O^t\right)\right)\right), \forall t, S^t, \qquad \textbf{[A.2]}$$

where the utility function $U((c,d))$ is the negative of the distance (i.e. the number of edges on the shortest path) between *c* and *d* on *G* (or any positive affine transformation of it).

$P(\mathcal{C},\mathcal{G})$ provides a natural scale of desirability also under a computational limitation, even though the errors that may exist then may also be "priced" differently, with other scores. Given a computational limitation, we say that a circuit that maximizes $P(\mathcal{C},\mathcal{G})$ subject to the computational limitation is a boundedly optimal circuit.

2. Theorem – Internal Conflict in Boundedly Optimal Circuits

Theorem: There exist graphs *G* and complexity bounds *l* such that, if *C* is the fanout-free circuit that maximizes the score *P*(*C*, *G*) subject to the complexity constraint $L(C) \le l$, then: (*i*) *C* decomposes into subcircuits $C_1,\ldots,C_a$ with inputs $I_i^t = g_i(S^t)$ and outputs $\hat{O}_i^t = F_i(I_i^t)$ for $i \in \{1,...,a\}$. If Φ_i is the set of all input values I_i^t, and Ω_i is the set of all output values O_i^t, then $I^t \in \Phi_1 \times \ldots \times \Phi_a$ and $O^t \in \Omega_1 \times \ldots \times \Omega_a$. (*ii*) There exist graphs $G_1,\ldots,G_a$ such that the behavior of each subcircuit C_i is described most parsimoniously as: $\hat{O}_i^t = \arg\max_{O_i^t} U_i(\bar{\lambda}_i(S^t,O_i^t)), \forall t, S^t$, where $\bar{\lambda}_i(S^t,O_i^t) = h_i(g_i(S^t),O_i^t)$, and h_i and U_i refer to G_i in the same way that *h* and *U* refer to *G*. (*iii*) In the task environment, there is conflict between the subcircuits, i.e.: $U_i(\lambda_i(S^t,\hat{O}_{-j}^t,\hat{O}_j^t)) < U_i(\lambda_i(S^t,\hat{O}_{-j}^t,\tilde{O}_j^t)) \exists t, S^t, \tilde{O}_j^t, \forall i, j \ne i$, where $\lambda_i(S^t,O^t) = g_i(h(S^t,O^t))$. [(*ii*) and (*iii*) correspond to the definition of conflict in the main text (Eqs. **2** & **3**.)]

3. Plan of Proof

3.1 Overview. There are three important graph constructs in the proof. First, a set of *a* strongly connected reflexive graphs (henceforth "factor graphs"), each with *q* nodes. Second, their graph product G'. And third, a graph *G* that is obtained by removing a certain edge, *e*, from G'. *G* is the task environment, the graph on which our circuit will be required to find shortest paths.

We focus on a certain circuit, C'. This circuit is made of *a* subcircuits, each perfect for one of the *a* factor graphs. The subcircuits are wired to C''s terminals such

that C' is perfect for G'. When an input vector representing both the current location and target of a robot on graph G' is fed into C', it is broken into *a* input vectors, each of which represents both a current location and target on one of the factor graphs. These smaller input vectors are fed by wires into their corresponding subcircuits, whose output vectors are then fed into their appropriate locations on C''s output terminals.

We can now see that, when we apply C' to the task environment *G* (rather than G'), it will do well but not perfectly. Most of the time it will take the correct step, because *G* is similar to G'. However, it will err whenever it attempts to cross the edge *e*, which exists in G' but not in *G*; where by "error" we mean a production of an output that is not consistent with any shortest path from origin to target.

The proof now proceeds in two phases. The first phase shows that there exists a computational limitation under which a circuit identical in its behavior to C' is a boundedly optimal circuit for *G*. This limitation is in fact the one which allows no more computational complexity than is necessary for the construction of a circuit identical in its behavior to C'. The bounded optimality of such a circuit is obtained from an interaction of the graph structures and computational limitations. The second phase shows that, based on information theoretic considerations, this boundedly optimal behavior manifests internal conflict. We will now outline these issues in turn.

3.2 Graph structures. The graph structure considerations can be divided into three parts. First, we construct the factor graphs such that the upper bound on the number of errors made by C' on graph *G* is $\leq (q-1)^a$, where *q* is the number of nodes on each factor graph. One method of achieving this is based on a graph-structural element that we will refer to as a "tooth" and will describe in detail in the formal proof section. Second, we consider scenarios where each subcircuit $j \neq i$ of C' receives as an input identical origin and target on its corresponding factor graph. The significance of these scenarios is that in them, subcircuit *i* is given no leeway, no matter where it is with respect to its own target on its own graph. Had it made an error with respect to its task of finding a shortest path on its own factor graph, C' would have also made an error with respect to its task of finding a shortest path on *G*. We will show that there are at least q^{a-1} such scenarios for each subcircuit *i*. Third, we observe that, for a large enough *a*, $(q-1)^a < q^{a-1}$.

From the above three elements, a central piece of the proof follows. If we wish to build a circuit that would do at least as well as a circuit identical in its behavior to C', then it must make no more mistakes than C', and hence the number of mistakes it makes, Q, must satisfy $Q \leq (q-1)^a < q^{a-1}$. Without needing to know on what occasions these errors will be made, we know from this inequality that for each block of this circuit's outputs that corresponds to a block controlled by one of the subcircuits of C', there is at least one scenario where that block must imitate the behavior of that subcircuit of C'. This central piece, in combination with the computational limitations, will be used to prove the bounded optimality of a circuit identical in its behavior to C'.

The method of graph construction provided here is simple and has been chosen for a pedagogical purpose. Although it requires the use of a large a, a slightly more elaborate structural requirement on the factor graphs completes the proof with $a = 2$ for arbitrarily large q.

3.3 Computational limitations. In deriving the consequences of the computational limitations we will use the following concept of "reducibility". Let $\mathcal{C}$ be a circuit, with input and output terminals partitioned into blocks $\mathcal{M}_1,\ldots,\mathcal{M}_a$ and $\mathcal{N}_1,\ldots,\mathcal{N}_a$ respectively. Let $\mathcal{F} = \mathcal{F}_1 \bigcup \ldots \bigcup \mathcal{F}_a$ be a partitioned set of Boolean functions of a partitioned set of arguments $\mathcal{I} = \mathcal{I}_1 \bigcup \ldots \bigcup \mathcal{I}_a$, such that, for $1 \leq i \leq a$, the functions in block $\mathcal{F}_i$ depend only on the arguments in block $\mathcal{I}_i$. We say that $\mathcal{C}$ is reducible to $\mathcal{F}$ if, for every $1 \leq i \leq a$, one can substitute constants at the inputs of $\mathcal{C}$ in the blocks $\mathcal{M}_j$ for all $j \neq i$ so that the resulting circuit produces at the outputs in block $\mathcal{N}_i$ the values of the functions $\mathcal{F}_i$ on the arguments $\mathcal{I}_i$ received at the inputs $\mathcal{M}_i$.

Now, let circuit $\mathcal{C}$ compute the set of functions $\mathcal{H}$, where $\mathcal{H} \neq \mathcal{F}$. Our proof method requires that if $\mathcal{C}$ is reducible to $\mathcal{F}$ then $L(\mathcal{F}) < L(\mathcal{H})$. To show this, it suffices to show that if $\mathcal{C}$ is reducible to $\mathcal{F}$ then one can simplify some gates in $\mathcal{C}$ and thereby make it compute $\mathcal{F}$. This can be shown in fanout-free circuits. Since the cost of a function that results from substituting a constant for an argument of a function $b \in \mathcal{B}$ is strictly less than the cost of b, the gates receiving signals from blocks $\mathcal{M}_j$ for all $j \neq i$ and influencing block $\mathcal{N}_i$, for all i, can be simplified.

The requirement above is also satisfied by limited sharing circuits, as will be explained in section 5; and, in general, the proof method provided here will work for any other computational model satisfying this requirement.

3.4. Synthesis. Section 3.2 shows that each block of the boundedly optimal circuit's outputs that corresponds to a block controlled by one of the subcircuits of C' must imitate the behavior of that subcircuit of C' in at least one of the scenarios described in section 3.1. More precisely, there is an assignment of constants to the boundedly optimal circuit's input terminals which correspond to the input terminals of all other subcircuits of C', such that the boundedly optimal circuit's output block in question imitates its counterpart in C'. According to section 3.3, this shows that the boundedly optimal circuit is reducible to a circuit identical in its behavior to C'. Hence, if it behaved any differently than C', it would have been more complex than C', but this is disallowed by the complexity limitation. It follows that the boundedly optimal circuit is the smallest circuit identical in its behavior to C'.

3.5 Inference of internal conflict. The boundedly optimal circuit can be decomposed into agents. The inputs and outputs of each agent correspond to the inputs and outputs of each subcircuit in C', and the "body" of each agent is either a fanout-free subcircuit or, in the case of limited-sharing circuits, a network of gates.

Based on the behavior of each agent, we can build a truth table for that agent, which lists each possible input vector along with the output vector that the agent produces in response to it. We find that the behavior of the agent, or equivalently its truth table, can be most parsimoniously described by assuming that the agent attempts to reach targets on a certain graph (one of the *a* factor graphs); in other words, the agent maximizes a utility function defined as proximity to a given target on a certain graph.

Conflict between the agents emerge when the boundedly optimal circuit attempts to cross the removed edge, *e*. In this case, if any agent had taken any other step than it actually did, a move would have been carried out, bringing all other agents closer to their targets and increasing their utilities. This state of affairs satisfies the definition of mutual conflict. The latter part of the next section will carry out the inference of utilities and conflict more formally.

Remark: Since agents are functional units made of simpler elements, they satisfy the biological definition of modules (1,2). Thus, the result also demonstrates the

emergence of modules from first principles. However, more importantly here, it shows that modules emerge that are in conflict with one another, even in a system designed for a single purpose.

4. Formal Proof for Fanout-Free Circuits

We begin by constructing *a* directed graphs. Each of these graphs has $q = 2^n$ nodes so that its nodes can be labeled with binary strings of length *n*. The parameters *a* and *q* will be chosen to satisfy the inequality

$$(q-1)^a < q^{a-1}. \qquad \textbf{[A.3]}$$

If $q \geq 4$ is chosen arbitrarily, this inequality will be satisfied for all sufficiently large *a*.

The construction that follows requires only that each of the *a* graphs be a reflexive and strongly connected graph with unique shortest paths in which there is an edge (v, w) and an additional node *u* such that the only nonloop directed into *u* is the edge (v, u) and the only nonloop directed out of *u* is the edge (u, w). We describe these three edges as forming a tooth (Fig. 3). Any set of *a* strongly connected reflexive graphs each having at least one tooth may be used. However, for definiteness, let each of the *a* graphs be a labeled version of a common graph G_0 in which every edge is a part of such a tooth.

Let G_0 be a reflexive graph comprising a directed cycle of 2^{n-1} nodes and, for each directed edge (v, w) on this cycle, an additional node *u*, and two additional edges (v, u) and (u, w) (Fig. 4). This gives a total of $q = 2^n$ nodes and $5q/2$ edges ($q/2$ in the cycle, another *q* additional edges, and *q* loops). Due to rotational symmetry, the nodes of this graph can be labeled with binary strings of length *n* in $q!/(q/2) = 2(q-1)!$ ways. Let $G_1, \ldots, G_a$ denote *a* labeled versions of G_0. We form the product graph $G' = G_1 \times \cdots \times G_a$. The labellings of $G_1, \ldots, G_a$ give rise to $\big(2(q-1)!\big)^a$ labellings of the product graph G'.

The graph G_0 is clearly strongly connected and, in fact, has unique shortest paths between all origins and targets. Furthermore, each edge (u, w) from a node *u* outside the cycle to a node *w* on the cycle belongs only to shortest paths originating at *u* and having targets distinct from *u*. Therefore, each such edge belongs to at most $q - 1$ shortest paths. For $1 \le i \le a$, let $e_i = (u_i, w_i)$ be such an edge in G_i. Let *G* be the graph obtained from G' by deleting the edge $e = ((u_1, \ldots, u_a), (w_1, \ldots, w_a))$. The edges e_i can be chosen in $(q/2)^a$ ways, giving a total of $(2(q-1)!)^a (q/2)^a = q!^a$ labeled graphs *G*.

(While the above construction requires Eq. **A.3** and a large *a*, a slightly more elaborate construction of the graphs G_i dispenses with these requirements and complete the proof with $a = 2$ for arbitrarily large *q*, as said.)

Let F' be the set of functions computed by circuit C' that is a perfect circuit for graph G' and that has minimum cost $L(C') = L(F')$. Let *C* be a fanout-free circuit that maximizes $P(C, G)$ subject to complexity constraint $L(C) \le L(F')$. We will show that *C* computes the same functions F' as C'.

The input terminals of C' can be partitioned into blocks $M'_1, \ldots, M'_a$ such that in each block $M'_{1 \le i \le a}$, half of the terminals receive the part of the origin (or current location) on G' that corresponds to G_i, and the other half receive the part of the target on G' that corresponds to G_i (recall that $G' = G_1 \times \ldots \times G_a$). The output terminals of C' can be partitioned into blocks $N'_1, \ldots, N'_a$ such that in each block $N'_{1 \le i \le a}$, the terminals produce the part of the next node to step onto on G' that corresponds to G_i. Let us also partition the input and output terminals of *C* into blocks $M_1, \ldots, M_a$ and $N_1, \ldots, N_a$ respectively, such that blocks M_i and N_i of *C* correspond to blocks M'_i and N'_i of C'.

We will now show that the circuit *C* is reducible to F'.

Since *C* is chosen to maximize $P(C, G)$ subject to $L(C) \le L(F') = L(C')$, we must have

$$P(C, G) \ge P(C', G). \qquad \textbf{[A.4]}$$

For convenience, let $Q(\mathcal{C},\mathcal{G}) \equiv p^2 - P(\mathcal{C},\mathcal{G})$. That is, $Q(\mathcal{C},\mathcal{G})$ represents the number of errors of $\mathcal{C}$ in $\mathcal{G}$. Thus Eq. **A.4** can be rewritten as:

$$Q(C,G) \le Q(C',G). \qquad \textbf{[A.5]}$$

Since G is a subgraph of G', any shortest path in G is at least as long as the unique corresponding shortest path in G'. Thus C' fails to find a shortest path in G only when the shortest path in G' involves the deleted edge $e = ((u_1,\ldots,u_a),(w_1,\ldots,w_a))$, and this happens only when, for each $1 \le i \le a$, the shortest path in G_i involves the edge (u_i, w_i). Since the only nonloop edge directed into u_i is directed out of v_i, and because G_i contains an edge (v_i, w_i), the shortest path in G_i for an origin-target pair involves the edge (u_i, w_i) only when the origin is the node u_i and the target is distinct from u_i. Since there are just $q - 1$ such origin-target pairs in G_i, there are just $(q-1)^a$ origin-target pairs in G for which C' fails to find a shortest path. Thus, $Q(C',G) = (q-1)^a$, and eqs. **A.5** and **A.3** imply

$$Q(C,G) < q^{a-1}. \qquad \textbf{[A.6]}$$

We say that an assignment of constants to the inputs in blocks M_j for all $j \ne i$ is diagonal if the origin equals the target in each of these $a - 1$ blocks. Thus, there are q^{a-1} diagonal assignments for each i. We say that a diagonal assignment is bad if the resulting circuit does not compute at the outputs in N_i the functions F_i' on the arguments I_i at the inputs M_i. Each bad diagonal assignment increases $Q(C,G)$ by at least 1 (i.e., contributes at least one mistake to $Q(C,G)$). Therefore Eq. **A.6** implies that, for each i, there is at least one diagonal assignment that is not bad. These assignments show that C is reducible to F'.

That C in fact computes F' now follows from the proposition below.

Proposition A.1: Let $\mathcal{C}$ be a fanout-free circuit, with inputs and outputs partitioned into blocks $\mathcal{M}_1,\ldots,\mathcal{M}_a$ and $\mathcal{N}_1,\ldots,\mathcal{N}_a$, respectively. Let $\mathcal{F} = \mathcal{F}_1 \bigcup \ldots \bigcup \mathcal{F}_a$ be a partitioned set of Boolean functions of a partitioned set of arguments $\mathcal{I} = \mathcal{I}_1 \bigcup \ldots \bigcup \mathcal{I}_a$, such that, for $1 \le i \le a$, the functions in block $\mathcal{F}_i$ depend only on the arguments in block $\mathcal{I}_i$. If no function in $\mathcal{F}$ is constant, $\mathcal{C}$ is reducible to $\mathcal{F}$, and $L(\mathcal{C}) \le L(\mathcal{F})$, then $\mathcal{C}$ computes $\mathcal{F}$.

Proof: For $1 \le i \le a$, let $\mathcal{C}_i$ be the fanout-free subcircuit of $\mathcal{C}$ computing the outputs in block $\mathcal{N}_i$. Then,

$$L(\mathcal{F}) = \sum_{1 \le i \le a} L(\mathcal{F}_i) \le \sum_{1 \le i \le a} L(\mathcal{C}_i) = L(\mathcal{C}) \le L(\mathcal{F}). \qquad \textbf{[A.7]}$$

Here the equalities reflect the additivity of cost for fanout-free circuits, the inequalities $L(\mathcal{F}_i) \le L(\mathcal{C}_i)$ reflect the reducibility of $\mathcal{C}$ to $\mathcal{F}$, and the last inequality holds by hypothesis. Suppose, to obtain a contradiction, that $\mathcal{C}$ at outputs $\mathcal{N}_i$ does not compute the functions $\mathcal{F}_i$ of inputs $\mathcal{I}_i$. Then there is a path in $\mathcal{C}_i$ from some input in a block $\mathcal{M}_j$ with $j \ne i$ to some output in block $\mathcal{N}_i$. This path must contain at least one gate, since no function in $\mathcal{F}_i$ is constant. If constants are substituted for inputs in block $\mathcal{M}_j$, at least the first gate on this path can be replaced by a gate with a strictly smaller cost. Thus we have $L(\mathcal{F}_i) < L(\mathcal{C}_i)$, and the first inequality in Eq. **A.7** can be strengthened to strict inequality, resulting in a contradiction [$L(\mathcal{F}_i) < L(\mathcal{C}_i)$ for some i, $L(\mathcal{F}_i) \le L(\mathcal{C}_i)$ for all other i, and $L(\mathcal{C}) \le L(\mathcal{F})$]. □

We have now shown that if C maximizes $P(C,G)$ subject to $L(C) \le L(F')$, then C computes the same functions F' as C'. This behavior of C satisfies the requirements for conflict as follows. We know that the graph G specifies a set of computationally-unlimited "best behaviors" B (produced by circuits C_B) that maximize the score on G equally well, and that each such set of best behaviors specifies the graph G. The former is obvious, and the latter follows from the fact that we can construct G from any B by inferring that G has an edge from node c to node $d \ne c$ if and only if the output of some

circuit in C_B is d for the input (c,d). It follows that there is a one-to-one correspondence between graphs G and sets of best behaviours, in other words, between graphs G and tasks. Information theory shows that, because of this one-to-one correspondence, describing G and the objective of finding shortest-paths on it provides a description of the task that is, for any descriptive formalism, asymptotically in q^a, a most parsimonious description.

More specifically, by a "most parsimonious description" we mean a description by a bit string using the minimum possible number of bits (3). For the graph G we may consider a description comprising (*a*) a description of the general scheme of construction (such as is presented in prose in the third through fourth paragraphs of this section, using a fixed number $O(1)$ of bits), (*b*) a description of the values of q and a [using $O(\log q) + O(\log a)$ bits], and (*c*) a description of the labeling of the nodes and the choice of the edge to be deleted [using $\log_2(q!^a) = aq\log_2 q + O(aq)$ bits]. The total length of this description is $aq\log_2 q + O(aq)$ bits. Such a description is asymptotically most parsimonious, since a description of one of $q!^a$ different labeled graphs cannot, on the average, use fewer than $aq\log_2 q + O(aq)$ bits.

A similar argument applies to the description of the behavior of each subcircuit. In this case, though, we wish to describe for each subcircuit the exact behavior that it exhibits (whereas the description of the collective's task specified a set of equally-desirable behaviors). Here we use the fact that each graph G_i has unique shortest paths. We know that the structure of a graph with unique shortest paths is determined by the behavior of a circuit that finds shortest paths in it, and vice versa. Therefore, describing G_i and the objective of finding shortest-paths on it is a most parsimonious description of the behavior of C_i (for any descriptive formalism, asymptotically in q). In other words, the utility function U_i provides a most parsimonious description of the behavior of C_i. (While there may be other equally parsimonious descriptions, there are none that are more parsimonious, which gives us license to use U_i as our description of interest.)

Compare, for instance, the utility-based description of a subcircuit with a description that lists an output sequence for each input sequence (i.e., a description consisting of the truth-table itself). For the former, we need to describe the structure of

the graph. According to a calculation similar to the one above, this can be done in $O(q \log_2 q)$ bits of information. For the latter, $O(q^2 \log_2 q)$ bits are needed, since for each of q^2 pairs of origin and target nodes we need to list an output node which can be specified with $\log_2 q$ bits of information. In other words, specifying the principle of operation is much more efficient than listing all possible responses $(q \log q << q^2 \log q)$.

Finally, in the task environment, U_i gives conflict as stated in the theorem. Whenever $(u_1,\ldots,u_a)$ is the current location and $(w_1,\ldots,w_a)$ is the next node on the shortest path to the target in G', each C_i produces w_i as its output. The output of the collective is then $(w_1,\ldots,w_a)$, which is illegal on *G*, and therefore no step is taken. In this situation, if any subcircuit had taken any other step than it actually did, the move could have been carried out. Then all other subcircuits would have been one step closer to their targets and would have increased their utilities. Thus, the subcircuits prevent each other from maximizing their respective utilities and, by definition, all subcircuits are in mutual conflict. This completes the proof of theorem. □

Remark: We observe that a situation similar to that of "Buridan's ass" occurs as a special case of this result. If we take $q = 4$ (Fig. 5) and $a = 5$, then there are $2^a - 2 = 30$ paths of length 2 between the endpoints of the deleted edge $e = ((u_1,\ldots,u_a),(w_1,\ldots,w_a))$. The circuit *C*, given origin $(u_1,\ldots,u_a)$ and target $(w_1,\ldots,w_a)$, is unable to choose among these alternatives, attempts to follow the deleted edge *e* (i.e., produces the illegal move) and remains frozen at $(u_1,\ldots,u_a)$. With a change of interpretation of the nodes of the graph, where nodes represent behavioral states rather than spatial locations (as described in the main text), the illegal move represents an attempt to enter behavioral states that cannot be entered simultaneously, such as approach and avoidance (Fig. 2).

5. Extending the proof to limited-sharing circuits

The most important assumption in the foregoing result is the restriction to fanout-free circuits. This assumption can be weakened as follows. Say that a circuit $\mathcal{C}$ (whose outputs are partitioned into blocks $\mathcal{N}_1,\ldots,\mathcal{N}_a$) is nonsharing if for every gate Γ in $\mathcal{C}$ there is at most one $1 \le i \le a$ for which there exists a path from Γ to some output in $\mathcal{N}_i$. Clearly, every fan-out free circuit is a nonsharing circuit, but the converse is false (in particular, when $a = 1$, a nonsharing circuit is not restricted at all). Even this weaker assumption can dramatically affect the number of gates needed to compute various sets of functions. This is illustrated by the result (4) that, for almost all Boolean functions f of n Boolean arguments, the number of gates needed to compute two copies of f on disjoint sets of arguments is asymptotic to the number needed to compute f once (whereas, for nonsharing circuits, computing two copies of f requires exactly twice as many gates).

The restriction to nonsharing circuits can be weakened still further. Let us define a notion of limited-sharing circuits in such a way that every nonsharing circuit is a limited-sharing circuit, but the converse is false. The definition of a limited-sharing circuit assumes that the inputs are partitioned into blocks $\mathcal{M}_1,\ldots,\mathcal{M}_a$, and the outputs are partitioned into blocks $\mathcal{N}_1,\ldots,\mathcal{N}_a$. Let $\mathcal{C}$ be such a circuit. Assign to every wire w in $\mathcal{C}$ the set $J_w \subseteq \{1,...,a\}$ of indices i such that w lies on some path from some input in $\mathcal{M}_i$ to some output in $\mathcal{N}_i$. Let Γ be a gate in $\mathcal{C}$, and let $w_1,\ldots,w_t$ be the wires fed by Γ. We say that Γ is a limited-sharing gate if

$$\sum_{1\le s\le t}(\#J_{w_s}-1) \le (\#\bigcup_{1\le s\le t}J_{w_s})-1\,. \qquad \textbf{[A.8]}$$

Finally, we say that $\mathcal{C}$ is a limited-sharing circuit if every gate in $\mathcal{C}$ is a limited-sharing gate.

Let a path in a circuit be a sequence of interconnected gates. Because of the unlimited fanout, paths can merge and diverge. One can think of limited-sharing as a limitation on the extent of repeated merging and diverging of paths, each of which ends

at the same block where it started (there is no limitation on merging and diverging of other, interblock paths).

The restriction to limited-sharing circuits will only be useful in the presence of certain additional assumptions concerning the gates used to build circuits, the way in which the complexity of circuits is assessed, and the functions computed by circuits. We assume that all circuits are built from gates in a set $\mathcal{B}$ of allowable gates. Assume furthermore that:

(1) If Γ is a gate in $\mathcal{B}$ depending on $k \geq 1$ arguments, then the result of substituting a constant (0 or 1) for some argument of Γ is a gate also in $\mathcal{B}$ and depending on $k - 1$ arguments.

(2) If Γ is a gate in $\mathcal{B}$, then result of complementing an argument of Γ, or the result of complementing the value of Γ, is a gate also in $\mathcal{B}$.

We say that a set $\mathcal{F}$ of Boolean functions is nondegenerate if no function in $\mathcal{F}$ is the complement of another function in $\mathcal{F}$, no function in $\mathcal{F}$ is the complement of one of its arguments, and no function in $\mathcal{F}$ is a constant function.

Lemma A.2: All gates depending on at most one argument can be eliminated from a circuit computing a nondegenerate set of functions without increasing its cost.

Proof: We begin by eliminating constant gates. Such a gate cannot feed an output, since the functions produced at the outputs are nondegenerate. If such a gate feeds another gate Γ, then Γ can be simplified by assumption **(1)**. By repetition of this procedure, all constant gates can be eliminated.

We finish by eliminating complement gates. If such a gate feeds another gate Γ, then it can be eliminated by modifying Γ according to assumption **(2)**. If such a gate feeds an output, then it cannot be fed by an input, since the functions produced at the outputs are nondegenerate. Thus it must be fed by another gate Γ. This gate Γ cannot feed an output, since a nondegenerate set of functions cannot contain two complementary functions. Thus the complement gate can be eliminated by complementing the value of the gate Γ according to assumption **(2)**, and complementing

the appropriate arguments of any gates fed by Γ, again according to assumption **(2)**. By repetition of this procedure, all complement gates can be eliminated. □

In what follows, we assume that the cost $L(\mathcal{C})$ of a limited-sharing circuit $\mathcal{C}$ is the sum of the costs of the gates in $\mathcal{C}$. From assumptions **(1)** and **(2),** it follows that the cost of a gate depending on no arguments (a constant gate) or one argument (a complement gate) is irrelevant, because such a gate can be removed. Finally, we assume that the cost of a gate depending on $k \geq 2$ arguments is $k - 1$. If $\mathcal{F}$ is a partitioned set of Boolean functions of a partitioned set of arguments, we let $L(\mathcal{F})$ denote the minimum possible cost of a limited-sharing circuit computing the functions in $\mathcal{F}$.

Proposition A.3: Let $\mathcal{C}$ be a limited-sharing circuit, with inputs and outputs partitioned into blocks $\mathcal{M}_1,\ldots,\mathcal{M}_a$ and $\mathcal{N}_1,\ldots,\mathcal{N}_a$, respectively. Let $\mathcal{F} = \mathcal{F}_1 \cup \ldots \cup \mathcal{F}_a$ be a partitioned set of Boolean functions of a partitioned set of arguments $\mathcal{I} = \mathcal{I}_1 \cup \ldots. \cup \mathcal{I}_a$, such that, for $1 \leq i \leq a$, the functions in block $\mathcal{F}_i$ depend only on the arguments in block $\mathcal{I}_i$. If $\mathcal{F}$ is nondegenerate, $\mathcal{C}$ is reducible to $\mathcal{F}$, and $L(\mathcal{C}) \leq L(\mathcal{F})$, then $\mathcal{C}$ computes $\mathcal{F}$.

Proof: Since $\mathcal{C}$ is reducible to a nondegenerate set of functions, it must itself compute a set of nondegenerate functions, so we may assume by lemma A.2 that every gate in $\mathcal{C}$ depends on at least two arguments.

Let us construct from $\mathcal{C}$ a circuit $\mathcal{C}'$ that computes the functions $\mathcal{F}$. We will show that if $\mathcal{C}$ itself did not compute $\mathcal{F}$, then we would have the strict inequality $L(\mathcal{C}') < L(\mathcal{C})$, which would yield the contradiction:

$$L(\mathcal{F}) \leq L(\mathcal{C}') < L(\mathcal{C}) \leq L(\mathcal{F}). \qquad \textbf{[A.9]}$$

(The leftmost inequality holds because $\mathcal{C}'$ computes $\mathcal{F}$, and the rightmost holds by hypothesis.)

The circuit $\mathcal{C}'$ will be obtained from $\mathcal{C}$ by replacing each wire w in $\mathcal{C}$ by a "cable" Γ_w that contains a wire w^i for each index i in J_w. The wire w^i will carry that Boolean function of the inputs in block $\mathcal{M}_i$ that is carried by w when the inputs in blocks $\mathcal{M}_j$ for $j \neq i$ are set to the constants that, by the definition of reducibility, cause $\mathcal{C}$ to compute the functions $\mathcal{F}_i$. The circuit $\mathcal{C}'$ will also contain a "module" M_g for every gate g in $\mathcal{C}$. If $v_1, \ldots, v_k$ are the wires feeding g and $w_1, \ldots, w_t$ are the wires fed by g, then M_g will compute the signals on the wires in $\bigcup_{1 \leq s \leq t} \{w_s^i : i \in J_{w_s}\}$ from those on the wires in $\bigcup_{1 \leq h \leq k} \{v_h^i : i \in J_{v_h}\}$.

If a wire w in $\mathcal{C}$ is fed by an input in block $\mathcal{M}_i$, then J_w is either ϕ or $\{i\}$, so the only possible wire in the cable Γ_w is w_i, and this wire can be fed by the corresponding input in $\mathcal{C}'$. If a wire w in $\mathcal{C}$ feeds an output in block $\mathcal{N}_i$, then $J_w = \{i\}$, so the corresponding output in $\mathcal{C}'$ can be fed by the wire w_i in the cable Γ_w.

Suppose now that g is a gate in $\mathcal{C}$ fed by wires $v_1, \ldots, v_k$ and feeding wires $w_1, \ldots, w_t$. For each i in $\bigcup_{1 \leq s \leq t} J_{w_s}$, the module M_g will contain a gate g^i computing the signals on the wires w_s^i in the cables Γ_{w_s} for those s for which $i \in J_{w_s}$. The gate g^i will be fed by the wires v_h^i in the cables Γ_{v_h} for those h for which $i \in J_{v_h}$. For those h for which $i \notin J_{v_h}$, g^i will be fed by the constants carried by the wires v_h in $\mathcal{C}$ when the inputs in blocks $\mathcal{M}_j$ for $j \neq i$ are set to the constants that, by the definition of reducibility, cause $\mathcal{C}$ to compute the functions $\mathcal{F}_i$.

Without any assumptions about the functions that $\mathcal{C}$ computes, we now show that in the conversion of $\mathcal{C}$ to $\mathcal{C}'$, no additional cost need be paid, and therefore, $L(\mathcal{C}') \leq L(\mathcal{C})$. For this purpose, we invent a virtual currency that can be obtained by selling gates in $\mathcal{C}$ and can be used to buy gates in $\mathcal{C}'$, and can be transferred along the wires of a circuit. We show that the conversion can be carried out for free, so that $L(\mathcal{C}') \leq L(\mathcal{C})$.

Let the "balance of payments" at each locality be the amount of payment flowing in, minus the amount of payment flowing out, minus any cost (or negative cost)

used in the conversion of that locality. That is, the balance of payments is the amount of currency remaining in the locality after the conversion. At each input, the balance of payments is simply zero minus the sum of payments flowing out of that input (outflow). (Zero represents the fact that no external investment is made). At each output, the balance of payments is the sum of payments flowing into that output (inflow). At each gate *g* in $\mathcal{C}$, the balance is inflow minus outflow, minus the cost of all the gates in the module M_g in $\mathcal{C}'$, plus the cost of the gate *g* in $\mathcal{C}$ (because *g* is replaced by M_g, we save the cost of *g*).

Assume that a payment of $\#J_w - 1$ units flows along each wire *w*, from the input or gate feeding *w* to the gate or output fed by *w*. If $J_w = \phi$, a payment of 1 unit is given to the input or gate feeding *w*.

Notice that, under this assumption, the balance of payments at each input is nonnegative, even though no external investment is made. This is because a wire *w* fed by an input in block $\mathcal{N}_i$ has J_w equal to ϕ or $\{i\}$. Furthermore, since a wire *w* feeding an output in block $\mathcal{N}_i$ has $J_w = \{i\}$, the balance of payments at an output is 0.

We now consider the balance of payments at a gate *g* in $\mathcal{C}$, fed by wires $v_1, \ldots v_k$ and feeding wires $w_1, \ldots, w_t$. We define $U = \bigcup_{1 \le s \le t} J_{w_s}$. Note that, by the definition of *J*, $\bigcup_{1 \le h \le k} J_{v_h} = U$. We know that:

(a) The inflow at *g* is $\sum_{1 \le h \le k} \left(\#J_{v_h} - 1\right)$.

(b) The outflow at *g* is $\sum_{1 \le s \le t} \left(\#J_{w_s} - 1\right)$. Since *C* is a limited-sharing circuit (every gate satisfies (A.8)), the outflow is $\le \#U - 1$.

(c) Since the cost $\Lambda(f)$ of gate *f* is $k - 1$ by definition, the cost of the module M_g, namely $\sum_{f \in M_g} \Lambda(f)$, equals the total number of wires entering the module minus the number of gates in the module:

$$\sum_{f \in M_g} \Lambda(f) = \left(\sum_{1 \le h \le k} \#J_{v_h}\right) - \#U.$$

(d) The cost of gate *g* (to be saved) is $\Lambda(g) = k - 1$.

It follows that the balance of payments at each gate is nonnegative, since:

$$\sum_{1\le h\le k}\left(\#J_{v_h}-1\right)-\left(\#U-1\right)-\left(\left(\sum_{1\le h\le k}\#J_{v_h}\right)-\#U\right)+k-1=0\,.$$

We now suppose that $\mathcal{C}$ does not compute $\mathcal{F}$, and obtain a contradiction in the form of the strict inequality $L(\mathcal{C}')<L(\mathcal{C})$. For this purpose, it suffices to show that at some input or gate, the balance of payments is strictly positive.

Assume that there is some wire *v* in $\mathcal{C}$ with $\#J_v\ge 2$. The wire *v* cannot feed an output, since a wire *w* feeding an output in block $\mathcal{N}_i$ has $J_w=\{i\}$. It follows that *v* feeds a gate *g*. Furthermore, we can choose *v* such that *g* feeds no wire *w* with $\#J_w\ge 2$. This we obtain by starting with some wire *v*; then, if $\#J_w\ge 2$, we choose *w* to be our focal wire *v*, and so on. Now, let $v_1,\ldots,v_{k-1}$ be the wires feeding *g* other than the wire *v*. We know that:

(a) The inflow of *g* is: $\#J_v+\left(\sum_{1\le h\le k-1}\#J_{v_h}\right)-k.$

(b) The outflow from *g* is nonpositive, since no wire *w* fed by *g* has $\#J_w\ge 2$.

(c) The cost of M_g is: $\sum_{f\in M_g}\Lambda(f)=\left(\sum_{1\le h\le k-1}\#J_{v_h}\right)+\#J_v-\#U\le\sum_{1\le h\le k-1}\#J_{v_h}$, since $J_v\subseteq U$.

(d) The cost of the gate *g* in $\mathcal{C}$ is: $\Lambda(g)=k-1$.

Since $\#J_v+\left(\sum_{1\le h\le k-1}\#J_{v_h}\right)-k-\left(\sum_{1\le h\le k-1}\#J_{v_h}\right)+k-1=\#J_v-1>0,$ we find that the balance of payments at *g* is strictly positive.

Suppose next that no wire *v* in $\mathcal{C}$ has $\#J_v\ge 2$, but that there is some wire *w* in $\mathcal{C}$ with $\#J_w=\phi$. If this wire is fed by a gate *g* for which some wire *v* feeding *g* has $\#J_v=\phi$, we choose *v* as our focal wire *w*, etc. Thus we may assume that *w* is either fed by an input, or fed by a gate that is not fed by any wire *v* with $\#J_v=\phi$. If *w* is fed by an input, then the balance of payments at that input is 1, and thus strictly positive. Suppose then that *w* is fed by a gate *g* fed by wires $v_1,\ldots,v_k$. Then:

(a) The inflow to *g* is zero, since every wire *v* feeding *g* has $\#J_v=1$.

(b) The outflow from g is strictly negative, since no wire w fed by g has $\#J_w \geq 2$, and at least one such wire has $\#J_w = \phi$.

(c) Since in the conversion of wires to cables, no cable receives more than one wire, the cost of M_g is $\sum_{f \in M_g} \Lambda(f) = k - \#U$.

(d) The cost of the gate g in $\mathcal{C}$ is: $\Lambda(g) = k - 1$.

Since $\#U \geq 1$, (d) $\geq$ (c), and therefore (a) – (b) – (c) + (d) > 0. The balance of payments at g is again strictly positive.

It remains to examine the case where every wire w in $\mathcal{C}$ has $\#J_w = 1$. If $\mathcal{C}$ does not compute $\mathcal{F}$, but $\mathcal{C}$ is reducible to $\mathcal{F}$, then there must exist some path π from some input in a block $\mathcal{M}_i$ to some output in a block $\mathcal{N}_i$ with $j \neq i$. The wire v of π fed by the input must have $J_v = \{i\}$, and the wire w of π feeding the output must have $J_w = \{j\}$. Thus there must exist some gate g on π that has $J_x = \{i\}$ for the wire x of π feeding g, but $J_y = \{j\}$, with $j \neq i$, for the wire y of π fed by g. Now,

(a) There is no inflow to g, since all wires have $\#J_w = 1$.

(b) For the same reason, there is no outflow from g.

(c) The cost of M_g is $\sum_{f \in M_g} \Lambda(f) = k - \#U$, as in the previous case.

(d) The cost of the gate g in $\mathcal{C}$ is: $\Lambda(g) = k - 1$.

Now notice that $\#U \geq 2$, since the distinct indices i and $j \neq i$ both appear in U. Thus (c) < (d), and the balance of payments is again strictly positive.

Since the balance of payments is nonnegative at every input, output or gate in $\mathcal{C}$, and since it is strictly positive in at least one input or gate if $\mathcal{C}$ does not compute $\mathcal{F}$, we may conclude that $L(\mathcal{C}') < L(\mathcal{C})$. This contradiction shows that $\mathcal{C}$ computes $\mathcal{F}$. □

Proposition A.3 now extends the proof of conflict to limited-sharing circuits, since it is a limited-sharing version of proposition A.1 in section 4.

6. Concluding Remarks

The proof outlined above accepts arbitrarily large circuits made of any kind/s of gates, including threshold gates, as long as the gates are simple relative to the circuit as a whole (and therefore many gates are needed for the circuit's construction). These facts reflect well the properties of neurons and brain. The two main limitations on the analogy between circuits and brains are: first, the limitation on the extent on merging and diverging of paths; and second, that circuits are acyclic, i.e., exclude loops. Thus, the model described raises interesting technical questions for future research, primarily, what further weakenings of the circuit structural requirements and the graph structural requirements are possible, questions that aim at the theoretical prevalence of conflict. Presently, we take the above as a proof of principle for internal conflict, and use the wealth of biological evidence as a guide for its prevalence.

1. Hartwell, L. H., Hopfield J. J., Leibler S. & Murray, A. W. (1999) *Nature* **402,** C47–C52.

2. Wagner, W. & Wagner G. P. (2003). *J. Cult. Evol. Psychol.* **1,** 135-166.

3. Li, M. & Vitànyi, P. M. B. (1997) *An Introduction to Kolmogorov Complexity and its Applications* (Springer, New York).

4. Uhlig, D. (1974) *Mathematicheskie Zametki*, **15**(6), 937–944; trans. *Mathematical Notes*, **15**(6), 558–562.