neural firing [4]. One reason for this sparse activation may be metabolic cost. For example, a recent biological study on the energy cost of cortical computation [6] concludes that "The cost of a single spike is high, and this limits, possibly to fewer than 1 %, the number of neurons that can be substantially active concurrently". The metabolic cost of the active ('1") state of a neuron is very asymmetric. The production of a spike consumes a substantial amount of energy (about 2.4×10^9 molecules of ATP according to [6]), whereas the energy cost of the no-spike rest state, is substantially less. In contrast to neuronal circuits, computations in feedforward threshold circuits (and many other circuit models for digital computation) have the property that a large portion, usually around 50%, of gates in the circuit output a "1" during any computation. Common abstract measures for the energy consumption of electronic circuits treat the cost of the two output states 0 and 1 of a gate symmetrically, and focus instead on the required number of switchings between these two states (see [5] and its references, as well as [11]). An exception are [14, 4, 1], which provide Shannon-type results for the number of gates that output a "1" in Boolean circuits consisting of gates with bounded fan-in. Circuits of threshold gates (= linear threshold gates = McCulloch-Pitts neurons) are an important class of circuits that are frequently used as simplified models for computations in neural circuits [8, 12, 10, 13]. In this paper we consider how investigations of such abstract threshold circuits can be reconciled with actual activity characteristics of biological neural networks.

In section 2 we give a precise definition of threshold circuits, and also define their *energy complexity*, whose minimization yields threshold circuits that carry out computations with sparse activity: on average few gates output a "1" during a computation. In section 2 we also introduce another novel complexity measure, the *entropy of a computation*. This measure is interesting for many types of circuits, beyond the threshold circuits discussed in this paper. It measures the total number of different patterns of gate states that arise during computations on different circuits inputs. We show in section 3 that the entropy of circuit states defines a coarse lower bound for its energy complexity. This result is relevant for any attempt to simulate a given threshold circuit by another threshold circuit with lower energy complexity, since the entropy of a circuit is directly linked to the algorithm that it implements. Therefore, it is unlikely that there exists a general method permitting any given circuit to be simulated by one with smaller entropy. In this sense the entropy of a circuit defines a hard lower bound for any general method that aims to simulate any given threshold circuit using a circuit with lower energy complexity. However, we will prove in section 3 that there exists a general method that reduces – if this entropy is $O(\log n)$ – the energy complexity of a circuit to a level near the entropy of the circuit. Since the entropy of a circuit is a complexity measure that is interesting in its own right, we also offer in section 4 a first result on the computational power of threshold circuits

[4] According to recent data [7] from whole cell recordings in awake animals the spontaneous firing rates are on average below 1 Hz.

with low entropy. Some open problems related to the new concepts introduced in this article are listed in section 5.

2 Definitions

A threshold gate g (with weights $w_1, \ldots, w_n \in \mathbb{R}$ and threshold $t \in \mathbb{R}$) outputs 1 for any input $X = (x_1, \ldots, x_n) \in \mathbb{R}^n$ if $\sum_{i=1}^n w_i x_i \geq t$, otherwise 0. We write $g(X) = \text{sign}(\sum_{i=1}^n w_i x_i - t)$ where $sign(z) = 1$ if $z \geq 0$ and $sign(z) = 0$ if $z < 0$. As usual we assume that threshold gates operate in discrete time, with unit delays between gates.

For a threshold gate g_i within a feedforward circuit C that receives $X = (x_1, \ldots, x_n)$ as *circuit input*, we write $g_i(X)$ for the output that the gate g_i gives for this circuit input X (although the actual input to gate g_i during this computation will in general consist of just some variables x_i from X, and in addition, or even exclusively, of outputs of other gates in the circuit C).

We define the *energy complexity* of a circuit C consisting of threshold gates $g_1, \ldots, g_m$ to be the expected number of 1's that occur in a computation, for some given distribution Q of circuit inputs X, i. e.

$$EC_Q(C) := E[\sum_{i=1}^m g_i(X)],$$

where the expectation is evaluated with regard to the distribution Q over $X \in \mathbb{R}^n$ (or $X \in \{0,1\}^n$). Thus, for the case where Q is the uniform distribution over $\{0,1\}^n$, we have $EC_{uniform} := \frac{1}{2^n} \sum_{X \in \{0,1\}^n} \sum_{i=1}^m g_i(X)$.
In some cases it is also interesting to consider the *maximal* energy consumption of a circuit for any input X, defined by

$$EC_{\max}(C) := \max(\sum_{i=1}^m g_i(X) : X \in \mathbb{R}^n).$$

We define the *entropy* of a (feedforward) circuit C to be

$$H_Q(C) := -\sum_{A \in \{0,1\}^m} P_C(A) \cdot \log P_C(A),$$

where $P_C(A)$ is the probability that the internal gates $g_1, \ldots, g_m$ of the circuit C assume the state $A \in \{0,1\}^m$ during a computation of circuit C (for some given distribution Q of circuit inputs $X \in \mathbb{R}^n$). We often write $H_{\max}(C)$ for the largest possible value that $H_Q(C)$ can assume for any distribution on a given set of circuit states A. If $MAX(C)$ is defined as the total number of different circuit states that circuit C assumes for different inputs $X \in \mathbb{R}^n$, then one has $H_Q(C) = H_{\max}(C)$ if Q is such that these $MAX(C)$ circuit states all occur with the same probability, and $H_{\max}(C)$ is then equal to $\log_2 MAX(C)$. Thus

$2^{H_{\max}(C)}$ is the maximal number of circuit states that a circuit C assumes for arbitrary inputs X.

We write $size(C)$ for the number m of gates in a circuit C, and $depth(C)$ for the length of the longest path in C from an input to its output node (which is always assumed to be the node g_m).

3 Construction of Threshold Circuits with Sparse Activity

Obviously, the number of 1's in a computation limits the number of states that the circuit can assume:

$$\begin{aligned} H_Q(C) &\leq \log(\#\ of\ circuit\ states\ A\ that\ C\ assumes) \\ &\leq \log \sum_{j=0}^{EC_{\max}(C)} \binom{\text{size}(C)}{j} \\ &\leq \log(\text{size}(C)^{EC_{\max}(C)}) = EC_{\max}(C) \cdot \log \text{size}(C) \end{aligned}$$

(for sufficiently large values of $EC_{\max}(C)$ and $size(C)$; log always stands for $\log_2$ in this paper). Hence

$$EC_{\max}(C) \geq H_Q(C)/\log \text{size}(C)\ . \tag{1}$$

In fact, this argument shows that

$$EC_{\max}(C) \geq H_{\max}(C)/\log \text{size}(C)\ . \tag{2}$$

Every Boolean function $f : \{0,1\}^n \to \{0,1\}$ can be computed by a threshold circuit C of depth 2 that represents its disjunctive normal form, in such a way that for every circuit input X at most a single gate on level 1 outputs a 1. This circuit C has the property that $EC_{\max}(C) = 2$ and $H_Q(C) = \log(\text{size}(C) - 1)$ for a suitable distribution Q of circuit inputs. Hence it is in some cases possible to achieve $EC_{\max}(C) < H_Q(C)$, and the factor $\log \text{size}(C)$ in (1) and (2) cannot be eliminated or significantly reduced.

Threshold circuits that represent a Boolean function f in its disjunctive normal form allow us to compute any Boolean function with a circuit C that achieves $EC_{\max}(C) = 2$. However these circuits C have in general exponential size in n. Therefore, the key question is whether one can also construct polynomial size circuits C with small EC_Q or $EC_{\max}$. Because of the a-priori bounds (1) and (2), this is only possible for those functions f that can be computed with a low entropy of circuit states. The following results show that, on the other hand, the existence of a circuit C that computes f with $H_{\max}(C) = O(\log n)$ is sufficient to guarantee the existence of a circuit that computes f with low energy complexity.

Theorem 1. *Assume that a Boolean function $f : \{0,1\}^n \to \{0,1\}$ can be computed by some polynomial size threshold circuit C with $H_{\max}(C) = O(\log n)$. Then f can also be computed by some polynomial size threshold circuit C' with*

$$EC_{\max}(C') \leq H_{\max}(C) + 1 = O(\log n). \tag{3}$$

Furthermore, if Q is any distribution of inputs $X \in \{0,1\}^n$, then it is possible to construct a polynomial size threshold circuit C'' with

$$EC_Q(C'') \leq \frac{H_Q(C)}{2} + 1 = O(\log n). \tag{4}$$

Remark 1. The proof below shows that the following more general statements hold for any function f and any distribution Q:

If f can be computed by some arbitrary (feedforward) threshold circuit C, then f can also be computed by a threshold circuit C' with $size(C') \leq 2^{H_{\max}(C)}$, $depth(C') \leq size(C)+1$, $H_{\max}(C') \leq H_{\max}(C)$, and $EC_{\max}(C') \leq H_{\max}(C)+1$.

Furthermore, f can also be computed by a threshold circuit C'' with $size(C'') \leq 2^{H_{\max}(C)}$, $depth(C'') \leq size(C) + 1$, $H_Q(C'') \leq H_Q(C)$, and $EC_Q(C'') \leq \frac{H_Q(C)}{2} + 1$.

Remark 2. The assumption $H_{\max}(C) = O(\log n)$ is satisfied by standard constructions of threshold circuits for many commonly considered functions f. Examples are all symmetric functions (hence in particular PARITY of n bits), COMPARISON of binary numbers, and BINARY ADDRESSING (routing) where the first k input bits represent an address for one of the 2^k subsequent input bits (thus $n = k + 2^k$). In fact, to the best of our knowledge there is no function known which can be computed by polynomial size threshold circuits, but not by polynomial size threshold circuits C with $H_{\max}(C) = O(\log n)$.

Proof of Theorem 1 The proof is split up into a number of Lemmata (Lemma 1 – 6). The idea is first to simulate in Lemma 1 the given circuit C by a threshold decision tree (i.e., by a decision tree T with threshold gates at its nodes, see Definition 1) that has at most $2^{H_{\max}(C)}$ leaves. Then this threshold decision tree is restructured in Lemma 3 in such a manner that every path in the tree from the root to a leaf takes at most log(# of leaves) times, hence in this case at most $H_{\max}(C)$ times, the right branch at an internal node. Obviously such an asymmetric cost measure is of interest when one wants to minimize an asymmetric complexity measure such as EC, which assigns different costs to gate outputs 0 and 1. Finally, we show in Lemma 5 that the computations of the resulting threshold decision tree can be simulated by a threshold circuit where some gate outputs a "1" whenever the simulated path in the decision tree moves into the right subtree at an internal node of the tree. The proof of this Lemma 5 has to take into account that the control structures of decision trees and circuits are quite different: A gate in a decision tree is activated only when the computation path happens to arrive at the corresponding node of the decision tree, but a gate in a threshold circuit is activated in *any* computation of that circuit. Hence a

threshold decision tree with few threshold gates that output "1" does not automatically yield a threshold circuit with low energy complexity. However, we show that all gates in the simulating threshold circuit that do not correspond to a node in the decision tree where the right branch is chosen, receive an additional input with a strongly negative weight (see Lemma 4), so that they output a "0" when they get activated.

Finally, we show in Lemma 6 that the threshold decision tree can be restructure alternatively, so that the *average* number of times when a computation path takes the right subtree at a node remains small (instead of the maximal number of taking the right subtree). This manouvre yields the proof of the second part of the claim of Theorem 1.

Definition 1 *A threshold decision tree(called a linear decision tree in [2]) T is a binary tree in which each internal node has two children, a left and a right one, and is labeled by a threshold gate that is applied to the input $X \in \{0,1\}^n$ for the tree. All the leaves of threshold decision trees are labeled by 0 or 1. To compute the output of a threshold decision tree T on an input X we apply the following procedure from the root until reaching a leaf: we go left if the gate at a node outputs 0, otherwise we go right. If we reach a leaf labeled by $l \in \{0,1\}$, then l is the output of T for input X.*

Note that the threshold gates in a threshold decision tree are only applied to input variables from the external input $X \in \{0,1\}^n$, not to outputs of preceding threshold gates. Hence it is obvious that computations in threshold decision trees have a quite different structure from computations in threshold circuits, although both models use the same type of computational operation at each node.

The depth of a threshold decision tree is the maximum number of nodes from the root to a leaf. We assign binary strings to nodes of T in the usual manner:

- $\hat{g}_\varepsilon$ denotes the root of the tree (where ε is the empty string)
- For a binary string s, let $\hat{g}_{s\circ 0}$ and $\hat{g}_{s\circ 1}$ be the left and right child of the node with label $\hat{g}_s$, where $\circ$ denotes concatenation of strings.

For example, the ancestors of a node $\hat{g}_{1011}$ are $\hat{g}_\epsilon$, $\hat{g}_1$, $\hat{g}_{10}$ and $\hat{g}_{101}$. Let S_T be the set of all binary strings s that occur as indices of nodes $\hat{g}_s$ in a threshold decision tree T. Then all the descendants of node $\hat{g}_s$ in T can be represented as $\hat{g}_{s\circ *}$ for $s \circ * \in S_T$.

The given threshold circuit C can be simulated in the following way by a threshold decision tree:

Lemma 1. *Let C be a threshold circuit computing a function $f : \{0,1\}^n \to \{0,1\}$ with m gates. Then one can construct a threshold decision tree T with at most $2^{H_{\max}(C)}$ leaves and depth$(T) \leq m$ which computes the same function f.*

Proof Assume that C consists of m gates. We number the gates $g_1, \ldots, g_m$ of C in topological order. Since g_i receives the circuit input X and the outputs of

g_j only for $j < i$ as its inputs, we can express the output $g_i(X)$ of g_i for circuit input $X =< x_1, \ldots, x_n >$ as $g_i(X) = sign(\sum_{j=1}^{n} w_j^i x_j + \sum_{j=1}^{i-1} w_{g_j}^i g_j(X) + t_i)$, where $w_{g_j}^i$ is the weight which g_i applies to the output of g_j in circuit C.

Let S be the set of all binary strings of length up to $m-1$. We define threshold gates $\hat{g}_s : X \to \{0,1\}$ for $s \in S$ by $\hat{g}_s(X) = sign(\sum_{j=1}^{n} w_j^{|s|+1} x_j + t_s)$ with $t_s = \sum_{j=1}^{|s|} w_{g_j}^{|s|+1} s_j + t_{|s|+1}$, where s_j is the j-th bit of string s and $|s|$ is the length of s. Obviously these gates $\hat{g}_s$ are variations of gate g_i with different built-in assumptions s about the outputs of preceding gates.

Let T be the threshold decision tree consisting of gates $\hat{g}_s$ for $s \in S$. That is, gate $\hat{g}_\epsilon = g_1$ is placed at the root of T. We let the left child of $\hat{g}_s$ be $\hat{g}_{s \circ 0}$ and the right child of $\hat{g}_s$ be $\hat{g}_{s \circ 1}$. We let each $\hat{g}_s$ with $|s| = m-1$ have a leaf labeled by 0 as left child and a leaf labeled 1 as right child. Since $\hat{g}_s$ computes the same function as $g_{|s|+1}$ if the preceding gates g_i output s_i for $1 \leq i \leq |s|$, T computes the same function f as C. We then remove all leaves from T for which the associated paths correspond to circuit states $A \in \{0,1\}^m$ that do not occur in C for any circuit input $X \in \{0,1\}^n$. This reduces the number of leaves in T to $2^{H_{\max}(C)}$. Finally, we iteratively remove all nodes without children, and replace all nodes below which there exists just a single leaf by a leaf. In this way we arrive again at a binary tree. □

We now introduce a cost measure $cost(T)$ for trees T, that like the energy complexity for circuits, measures for threshold decision trees how often a threshold gate outputs a 1 during a computation:

Definition 2 *We denote by $cost(T)$ the maximum number of times where a path from the root to a leaf in a binary tree T goes to the right. If T is a leaf, then $cost(T) = 0$.*

We will show later, in Lemma 5, that one can simulate any threshold decision tree T' by a threshold circuit $C_{T'}$ with $EC_{\max}(C_{T'}) \leq cost(T') + 1$. Hence it suffices for the proof of Theorem 1 to simulate the threshold decision tree T resulting from Lemma 1 by another threshold decision tree T' for which $cost(T')$ is small. This is done in Lemma 4, where we will construct a tree T' that reduces $cost(T')$ down to another cost measure $rank(T)$. This measure $rank(T)$ always has a value $\leq \log(\#$ of leaves of $T)$ according to Lemma 2, hence $rank(T) \leq H_{\max}(C)$ for the tree T constructed in Lemma 1.

Definition 3 *The rank of a binary tree T is defined inductively as follows:*

- *If T is a leaf then $rank(T) = 0$.*
- *If T has subtrees T_l and T_r then*

$$rank(T) = \begin{cases} rank(T_l), & \text{if } rank(T_l) > rank(T_r) \\ rank(T_r) + 1, & \text{if } rank(T_l) = rank(T_r) \\ rank(T_r), & \text{if } rank(T_l) < rank(T_r) \ . \end{cases}$$

Lemma 2. *Let T be any binary tree. Then $rank(T) \leq \log(\#$ of leaves of $T)$.* □

Lemma 3. *Let T be a threshold decision tree computing a function $f : \{0,1\}^n \to \{0,1\}$. Then f can also be computed by a threshold decision tree T' which has the same depth and the same number of leaves as T, and which satisfies $cost(T') = rank(t)$.*

Proof Let T consist of gates g_s for $s \in S_T$. We define T^s as the subtree of T whose root is g_s. Let T_l^s(respectively, T_r^s) denote the left(right) subtree below the root of T^s. We modify T inductively by the following procedure, starting at the nodes g_s of largest depth. If $cost(T_l^s) < cost(T_r^s)$, we replace g_s by its complement, and swap the left subtree and the right subtree. The complement of g_s is here another threshold gate g that outputs 1 if and only if g_s outputs 0. Such gate g exists since $\sum_{i=1}^n w_i x_i < t \Leftrightarrow \sum_{i=1}^n (-w_i) x_i > -t \Leftrightarrow \sum_{i=1}^n (-w_i) x_i \geq t'$ for another threshold t' (which always exists if the x_i assume only finitely many values). Let $\hat{T}^s$ be the threshold decision tree which is produced from T^s by this procedure. By construction it has the following properties:

- If the children of g_s both are both leaves, then we have $cost(\hat{T}^s) = 1$.
- Otherwise,

$$cost(\hat{T}^s) = \begin{cases} cost(\hat{T}_l^s), & \text{if } cost(\hat{T}_l^s) > cost(\hat{T}_r^s) \\ cost(\hat{T}_r^s) + 1, & \text{if } cost(\hat{T}_l^s) = cost(\hat{T}_r^s) \\ cost(\hat{T}_r^s), & \text{if } cost(\hat{T}_l^s) < cost(\hat{T}_r^s) \end{cases},$$

  where $\hat{T}^s$ has subtrees $\hat{T}_l^s$ and $\hat{T}_r^s$.

Since this definition coincides with the definition of the rank, we have constructed a tree T' with $cost(T') = rank(T)$. This procedure preserves the function that is computed, the depth of the tree, and the number of leaves. □

We now show that the threshold decision tree that was constructed in Lemma 3 can be simulated by a threshold circuit with low energy complexity. As a preparation we first observe in Lemma 4 that one can "veto" any threshold gate g through some extra input. This will be used in Lemma 5 in order to avoid the event that gates in the simulating circuit that correspond to gates in an inactive path of the simulated threshold decision tree increase the energy complexity of the resulting circuit.

Lemma 4. *Let $g(x_1, \ldots, x_n) = sign(\sum_{i=1}^n w_i x_i - t)$ be a threshold gate. Then one can construct a threshold gate g' using an additional input x_{n+1} which has the following property:*

$$g'(x_1, \ldots, x_n, x_{n+1}) = \begin{cases} 0, & \text{if } \; x_{n+1} = 1 \\ g(x_1, \ldots, x_n), & \text{if } \; x_{n+1} = 0 \end{cases}.$$

Proof We set $w_{n+1} := -(\sum_{i=1}^n |w_i| + |t| + 1)$. Apart from that g' uses the same weights and threshold as g. It is obvious that the resulting gate g' has the desired property. □

Lemma 5. *Let T be a threshold decision tree which consists of k internal nodes and which computes a function f. Then one can construct a threshold circuit C_T with $EC_{\max}(C_T) \leq cost(T) + 1$ that computes the same function f. In addition C_T satisfies $depth(C_T) \leq depth(T) + 1$ and $size(C_T) \leq k + 1$.*

Proof We can assume without loss of generality that every leaf with label 1 in T is the right child of its parent (if this is not the case, swap this leaf with the right subtree of the parent, and replace the threshold gate at the parent node like in the proof of Lemma 3 by another threshold gate that always outputs the negation of the former gate; this procedure does not increase the cost of the tree, nor its depth or number of internal nodes). Let now $g_s(X) = sign(\sum_{j=1}^{n} w_j^s x_j - t_s)$ be the threshold gate in T at the node with label $s \in S_T$. Let w_{n+1}^s be the weight constructed in Lemma 4 for an additional input which can force gate g_s to output 0. Set $W := \max\{|w_{n+1}^s| : s \in S_T\}$.

The threshold circuit C_T that simulates T has a gate g_s' for every gate g_s in T, and in addition an OR-gate which receives inputs from all gates g_s' so that g_s has a leaf with label 1 (according to our preceding remark this leaf is reached whenever the gate g_s at node $s \in S_T$ gets activated and g_s outputs a 1). We make sure that any gate g_s' in C_T outputs 1 for a circuit input X if and only if the gate g_s in T gets activated for this input X, and outputs 1. This implies that only gates g_s' in C_T can output 1 that correspond to gates g_s in T with output 1 that lie on the single path of T that gets activated for the present circuit input X. Hence this construction automatically ensures that $EC_{\max}(CT) \leq cost(T) + 1$ (where the "+1"arises from the additional OR-gate in C_T).

In order to achieve this objective, g_s' gets additional inputs from all gates $g_{\tilde{s}}'$ in C_T so that $\tilde{s}$ is a proper prefix of s. The weight for the additional input from $g_{\tilde{s}}'$ is $-W$ if $\tilde{s} \circ 0$ is a prefix of s, and W otherwise. In addition the threshold of g_s' is increased by $l_s \cdot W$, where l_s is the number of $1's$ in the binary string s. In this way g_s' can output 1 if and only if *g_s outputs 1 for the present circuit input X, and all gates $g_{\tilde{s}}$ of T for which g_s lies in the right subtree below $g_{\tilde{s}}$ output 1, and all gates $\hat{g}_{\tilde{s}}$ of T for which g_s lies in the left subtree below $g_{\tilde{s}}$ output 0.* Thus g_s' outputs 1 if and only if the path leading to gate g_s gets activated in T and g_s outputs 1. □

The *proof of the first claim of Theorem 1* follows now immediately from the Lemmata 1–5. Note that the number k of internal nodes in a binary tree is equal to (# of leaves)-1, hence $k \leq 2^{H_{\max}(C)} - 1$ in the case of the decision tree T resulting from applications of Lemma 1 and Lemma 3. This yields $size(C_T) \leq 2^{H_{\max}(C)}$ for the circuit C_T that is constructed in Lemma 5 for this tree T.

The *proof of the second claim of Theorem 1* follows by applying the subsequent Lemma 6 instead of Lemma 3 to the threshold decision tree T resulting from Lemma 1.

Lemma 6. *Let T be a threshold decision tree computing $f\colon \{0,1\}^n \to \{0,1\}$. Then for any given distribution Q of circuit inputs, there exists a threshold decision tree T' computing f such that the expected number of 1's with regard to Q is at most $H_Q(C)/2$.*

Proof Let $P(s)$ be the probability (with regard to Q) that gate g_s outputs 1. We construct T' by modifying T inductively (starting at the nodes of the largest depth m in T) through the following procedure: If $P(s) > 1/2$, replace g_s by a threshold gate which computes its negation and swap the left and right subtree below this node.

Let $cost_Q(s)$ be the expected number of times where one goes to the right in the subtree of T' whose root is the node labeled by s. By construction we have $P(s) \leq 1/2$ for every gate g_s in T'. Furthermore we have:

- If $|s| = m - 1$ then $cost_Q(s) = P(s)$.
- If $0 \leq |s| < m - 1$, then $P(s) \leq 1/2$ and $cost_Q(s) = P(s) + P(s)cost_Q(s \circ 1) + (1 - P(s))cost_Q(s \circ 0)$.

One can prove by induction on $|s|$ that $cost_Q(s) \leq H_Q(s)/2$ for all $s \in S_{T'}$, where $H_Q(s)$ is the entropy of states of the ensemble of gates of T' in the subtree below gate g_s.

For the induction step one uses the convexity of the log-function, which implies that $P(s) = -P(s) \cdot (-1) = -P(s) \cdot \log \frac{P(s)+(1-P(s))}{2} \leq$
$-P(s)\left(\frac{\log(P(s))+\log(1-P(s))}{2}\right)$, and the fact that $P(s) \leq 1 - P(s)$ to show that

$$\begin{aligned} cost_Q(s) &\leq P(s) + P(s) \cdot \frac{H_Q(s \circ 1)}{2} + (1 - P(s)) \cdot \frac{H_Q(s \circ 0)}{2} \\ &\leq -P(s) \cdot \left(\frac{\log P(s) + \log(1 - P(s))}{2}\right) + \\ &\quad P(s)\frac{H_Q(s \circ 1)}{2} + (1 - P(s)) \cdot \frac{H_Q(s \circ 0)}{2} \\ &\leq -\frac{P(s)}{2}\log P(s) - \frac{(1 - P(s))}{2}\log(1 - P(s)) + P(s)\frac{H_Q(s \circ 1)}{2} \\ &\quad +(1 - P(s))\frac{H_Q(s \circ 0)}{2} \leq \frac{H_Q(s)}{2} \ . \end{aligned}$$

□

Remark 3. The results of this section can also be applied to circuits that compute arbitrary functions $f : D \to \{0, 1\}$ for some arbitrary finite set $D \subseteq \mathbb{R}^n$ (instead of $\{0, 1\}^n$). For domains $D \subseteq \mathbb{R}^n$ of *infinite* size a different proof would be needed, since then one can no longer replace any given threshold gate by another threshold gate that computes its negation (as used in the proofs of Lemma 3, Lemma 5, and Lemma 6).

4 On the Computational Power of Circuits with Low Entropy

The concepts discussed in this article raise the question which functions $f : \{0, 1\}^n \to \{0, 1\}$ can be computed by polynomial size threshold circuits C with

$H_{\max}(C) = O(\log n)$. There is currently no function f in P (or even in NP) known for which this is provably false. But the following result shows that if all functions that can be computed by polynomial size threshold circuits of bounded depth can be computed by a circuit C of the same type which satisfies in addition $H_{\max}(C) = O(\log n)$, then this implies a collapse of the depth hierarchy for polynomial size threshold circuits.

Theorem 2. *Assume that a function $f : \{0,1\}^n \to \{0,1\}$ (or $f : \mathbb{R}^n \to \{0,1\}$) can be computed by a threshold circuit C with polynomially in n many gates and $H_{\max}(C) = O(\log n)$. Then one can compute f with a polynomial size threshold circuit C' of depth 3.*

Proof According to Lemma 1 there exists a threshold decision tree T with polynomially in n many leaves and $depth(T) \leq size(C)$. Design (similarly as in [2]) for each path p from the root to a leaf with output 1 in T a threshold gate g_p on layer 2 of C' that outputs 1 if and only if this path p becomes active in T. The output gate on layer 3 of C' is simply an OR of all these gates g_p. □

5 Discussion

In this article we introduced an energy complexity measure for threshold circuits that reflects the biological fact that the firing of a neuron consumes more energy than its non-firing. We also have provided methods for restructuring a given threshold circuit with high energy consumption by a threshold circuit that computes the same function, but with brain-like sparse activity. Theorem 1 implies that the computational power of such circuits is quite large. The resulting circuits with sparse activity may help us to elucidate the way in which circuits of neurons are designed in biological systems. In fact, the structure of computations in the threshold circuits with sparse activity that were constructed in the proof of Theorem 1 is reminiscent of biological results on the structure of computations in cortical circuits of neurons, where there is concern for the selection of different pathways ("dynamic routing") in dependence of the stimulus [9]. In addition our constructions provide first steps towards the design of algorithms for future extremely dense VLSI implementations of neurally inspired circuits, where energy consumption and heat dissipation become critical factors.

The new concepts and results of this article suggest a number of interesting open problems in computational complexity theory. At the beginning of section 3 we showed that the energy complexity of a threshold circuit that computes some functions f cannot be less than the a-priori bound given by the minimal circuit entropy required for computing such a function. This result suggests that the entropy of circuit states required for various practically relevant functions should be investigated. Another interesting open problem is the tradeoff between energy complexity and computation speed in threshold circuits, both in general and for concrete computational problems. Finally, we consider that both the energy complexity and the entropy of threshold circuits are concepts that are of interest in their own right. They give rise to interesting complexity classes that have not

been considered previously in computational complexity theory. In particular, it may be possible to develop new lower bound methods for circuits with low entropy, thereby enlarging the reservoir of lower bound techniques in circuit complexity theory.

6 Acknowledgments

We would like to thank Michael Pfeiffer, Pavel Pudlak and Robert Legenstein for helpful discussions, Kazuyuki Amano and Eiji Takimoto for their advice, and Akira Maruoka for making this collaboration possible. This work was partially supported by the Austrian Science Fund FWF, project # P15386, project # S9102-N04, and projects # FP6-015879 (FACETS) and FP6-2005-015803 (DAISY) of the European Union.

References

1. O. V. Cheremisin. (2003). On the activity of cell circuits realising the system of all conjunctions. *Discrete Mathematics and Applications*, 13(2):209–219.
2. H. D. Gröger and G. Turán. (1991). On linear decision trees computing Boolean functions. *Lecture Notes in Computer Science*, 510:707–718.
3. A. Hajnal, W. Maass, P. Pudlak, M. Szegedy, and G. Turan.(1993) Threshold circuits of bounded depth. *Journal of Computer and System Sciences*, 46:129–154.
4. 0. M. Kasim-Zade. (1992). On a measure of the activeness of circuits made of functional elements (Russian). *Mathematical problems in cybernetics*, 4:218–228, "Nauka", Moscow, see Math. Reviews MR1217502 (94c:94019).
5. G. Kissin. (1991). Upper and lower bounds on switching energy in VLSI. *J. of Assoc. for Comp. Mach.*, 38:222–254.
6. P. Lennie. (2003). The cost of cortical computation. *Current Biology*, 13:493–497.
7. T. W. Margrie, M. Brecht, and B. Sakmann. (2002). In vivo, low-resistance, whole-cell recordings from neurons in the anaesthetized and awake mammalian brain. *Pflugers Arch.*, 444(4):491–498.
8. M. Minsky. and S. Papert. (1988). *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, MA.
9. B. A. Olshausen, C. H. Anderson, and D. C. V. Essen. (1995). A multiscale dynamic routing circuit for forming size- and position-invariant object representations. *J. Comput. Neurosci.*, 2(1):45–62.
10. I. Parberry. (1994). *Circuit Complexity and Neural Networks*. MIT Press.
11. J. H. Reif and A. Tyagi. (1990) Energy complexity of optical computations. In *Proceedings of the Second IEEE Symposium on Parallel and Distributed Processing (December 1990)*, 14–21.
12. V. P. Roychowdhury, K. Y. Siu, and A. Orlitsky. (1994). *Theoretical Advances in Neural Computation and Learning*. Kluwer Academic, Boston.
13. K. Y. Siu, V. Roychowdhury, and T. Kailath. (1995). *Discrete Neural Computation; A Theoretical Foundation*. Information and System Sciences Series. Prentice-Hall.
14. M. N. Weinzweig. (1961). On the power of networks of functional elements, (Dokl.Akad.Nauk SSSR 139(1961),320-323 (Russian). *in English: Sov.Phys.Dokl*, 6:545–547, see Math. Reviews MR0134413 (24 #B466).