Theoretical Computer Science 38 (1985) 55-68
North-Holland

ON THE COMPLEXITY OF SLICE FUNCTIONS

Ingo WEGENER*
Fachbereich 20 (Informatik), Johann Wolfgang Goethe-Universität, Postfach 11 19 32, 6000 Frankfurt a.M., Fed. Rep. Germany

Communicated by M.S. Paterson
Received January 1984
Revised September 1984

Abstract. By a result of Berkowitz (1982), the monotone circuit complexity of slice functions cannot be much larger than the circuit (combinational) complexity of these functions for arbitrary complete bases. This result strengthens the importance of the theory of monotone circuits. We show in this paper that monotone circuits for slice functions can be understood as special circuits called set circuits. Here, disjunction and conjunction are replaced by set union and set intersection. All the main methods known for proving lower bounds on the monotone complexity of Boolean functions fail to work in their present form for slice functions. Furthermore, we show that the canonical slice functions of the Boolean convolution, the Nechiporuk Boolean sums, and the clique function can be computed with a linear number of gates.

1. Introduction

We investigate the complexity of slice functions. A function $f:\{0,1\}^n \to \{0,1\}^m$ is called a *k-slice* iff $f(x)$ equals the 0-vector if x has less than k ones and $f(x)$ equals the 1-vector if x has more than k ones. That means the interesting part of f happens when x has exactly k ones.

For the computation of Boolean functions we consider Boolean circuits (for the definition and elementary properties, see [6]) and the circuit complexity (combinational complexity) either over the complete basis of all binary Boolean functions or over the monotone basis consisting of binary conjunction and disjunction. These complexity measures are denoted by C and C_{m}. One knows that one may prove $\mathrm{NP} \neq \mathrm{P}$ by proving a nonpolynomial lower bound on the circuit complexity of a function in NP. In general, we know only little about the relation between C and C_{m}. Berkowitz [2] was able to show that these two complexity measures are closely connected for slice functions. In Section 2 we present these results in more detail.

In Section 3 we show by standard arguments the existence of many hard slice functions. In Section 4 we investigate the structure of monotone circuits for slice functions. If one ignores an additive $O(n \log n)$ term for the complexity, one may

* Supported in part by DFG Grant No. We 1066/1-1.

replace conjunctions by set intersections and disjunctions by set unions. Set circuits are the heart of circuits and monotone circuits for slice functions.

Many important functions like Boolean convolution, Boolean sums or the clique functions have the property that all prime implicants have the same length. If this length is k, the proper k-slice may be called the *canonical slice function.* It has among others the same prime implicants as the given function, while the $(k-1)$-slice equals the kth threshold function and all other slices have only other prime implicants. We show in Section 5 that the canonical slice of Boolean convolution has linear complexity while the monotone complexity of Boolean convolution is at least $n^{3/2}$ [11]. In Section 6 we show a similar result for Boolean sums. In this situation one gets also a one-output function f where the set complexity of the canonical slice is linear and where one believes that the monotone complexity of f is $\Theta(n^{3/2})$. Furthermore, we show in Section 7 that the canonical slice of each clique function can be computed with a linear number of gates.

By these results and the result of Section 2 that a hard function has hard slices one has to ask which slices of a given hard Boolean function are the hard ones. And one may ask which methods for proving large lower bounds work for slices. By the results of Sections 5 and 6 and some additional observations we show in Section 8 that the known methods in their present form do not work for slice functions.

2. Circuits for slice functions vs. monotone circuits

Definition 2.1. A Boolean function $f: \{0,1\}^n \rightarrow \{0,1\}^m$ is called a *k-slice* iff $f(x)$ equals the 0-vector if x contains less than k ones and equals the 1-vector if x contains more than k ones.

Definition 2.2. Let T_m^n denote the *threshold-m-function* on n variables, computing 1 iff the input has at least m ones. Let E_m^n denote the *exactly-m-function*, computing 1 iff the input has exactly m ones.

Definition 2.3. For an arbitrary Boolean function f its k-slice $f_{\text{sl-}k}$ is defined by $f_{\text{sl-}k} := (f \wedge E_k^n) \vee T_{k+1}^n$.

It is easy to see that $f_{\text{sl-}k} = (f \wedge T_k^n) \vee T_{k+1}^n$.

Definition 2.4. If all prime implicants of a monotone function have length k, we call $f_{\text{sl-}k}$ the canonical slice of f and denote it by f_{csl}.

We notice that in this situation all prime implicants of f are prime implicants of its canonical slice and no prime implicant of f is a prime implicant of any other slice with the sole exception of the $(k-1)$-slice which equals T_k^n. For monotone f with prime implicants of length k only we have $f_{\text{csl}} = f \vee T_{k+1}^n$.

It is well known that the set of all threshold functions $T^n = (T_1^n, \ldots, T_n^n)$ has linear complexity over a complete basis [6] and has complexity $O(n \log n)$ over the monotone basis [1]. Furthermore, $C_m(T_k^n)$ is $O(n)$ if k is fixed. Thus, the slices cannot be much harder than the given function f. It is also easy to see that some slices can be very easy (e.g., $k=1$ or $k=n$) even if f is hard. But we can prove that not all slices of a given hard function can be easy.

Proposition 2.5

(i) $C(f) \leq \sum_{1 \leq k \leq n} C(f_{\text{sl-}k}) + O(n)$,

(ii) $C(f_{\text{sl-}k}) \leq C(f) + O(n)$,

(iii) $C_m(f_{\text{sl-}k}) \leq C_m(f) + O(n \log n)$,

(iv) $C_m(f_{\text{sl-}k}) \leq C_m(f) + O(n)$ *if k is fixed.*

Proof. Obviously, the first assertion follows from the facts that

$$f = \bigvee_{1 \leq k \leq n} (f_{\text{sl-}k} \wedge E_k^n), \quad E_k^n = T_k^n \wedge \overline{T_{k+1}^n} \quad \text{and} \quad C(T^n) = O(n).$$

The second, third and fourth assertion follow from the definition of $f_{\text{sl-}k}$, $C(T^n) = O(n)$, $C_m(T^n) = O(n \log n)$, and $C_m(T_k^n) = O(n)$ for fixed k. □

Proposition 2.5 states that in order to prove large lower bounds on the (monotone) circuit complexity of f it is sufficient to consider all slices of f. For a complete basis we know that some slice of a hard function has to be hard. This cannot be proved for the monotone basis directly since we cannot compute f monotonely from its slices.

Until now we can prove nonlinear lower bounds for the monotone complexity of n-output functions only [10, 11] and not for complete bases. It seems to be much easier to prove lower bounds for the monotone basis than for complete bases. The following result of Berkowitz shows that for slice functions the monotone complexity and the circuit complexity are closely related. Thus, a hard Boolean function must have a slice whose monotone complexity is large. This proves the importance of the theory of monotone circuits.

Theorem 2.6. *Let f be a k-slice. Then*

$$C_m(f) \leq O(C(f)) + O(n \min\{k, n+1-k, \log^2 n\}).$$

Remark 2.7. Berkowitz [2] proved that $C_m(f) \leq O(C(f)) + O(n^2 \log n)$. Valiant [9] improved the additive term to $O(n \log^2 n)$. Here we give a shorter and, as we hope, easier proof of this bound. The result for small or large k uses ideas of Paterson also.

Proof of Theorem 2.6. At first we show how to change a circuit for f into a monotone circuit. An optimal circuit for f may be replaced by a circuit of conjunctions, disjunctions, and negations only. The complexity increases by a constant factor only. By the rules of De Morgan we can then get a circuit where only the variables are negated and the complexity is at most doubled.

The problem is how to compute $\bar{x}_i$ over the monotone basis. This is in general impossible, but we may use the following trick. Let $X := \{x_1, \ldots, x_n\}$ and $X_i := X - \{x_i\}$. For inputs with exactly k ones it holds that $x_i = 0$ iff $T_k^{n-1}(X_i) = 1$, that means $T_k^{n-1}(X_i) = \bar{x}_i$. We replace in our circuit for f, $\bar{x}_i$ by $T_k^{n-1}(X_i)$. The new circuit again computes f. This follows for inputs with k ones by the considerations above. For inputs with more than k ones, $T_k^{n-1}(X_i) = 1$ and by monotonicity the circuit again computes 1. For inputs with less than k ones, $T_k^{n-1}(X_i) = 0$ and by monotonicity the circuit again computes 0. Thus we have shown that

$$C_{\mathrm{m}}(f) \leq \mathrm{O}(C(f)) + C_{\mathrm{m}}(T_k^{n-1}(X_1), \ldots, T_k^{n-1}(X_n)).$$

Now we are going to estimate

$$C_{\mathrm{m}}(T_k^{n-1}(X_1), \ldots, T_k^{n-1}(X_n)).$$

The following algorithm is efficient for small k. Obviously,

$$T_k^{n-1}(X_i) = \bigvee_{p+q=k} T_p^{i-1}(x_1, \ldots, x_{i-1}) \wedge T_q^{n-i}(x_{i+1}, \ldots, x_n).$$

If we have computed all $T_p^i(x_1, \ldots, x_i)$ and all $T_p^{n-i}(x_{i+1}, \ldots, x_n)$ $(1 \leq p \leq k,\ 1 \leq i \leq n)$, $2nk$ gates are sufficient to compute all $T_k^{n-1}(X_i)$. Since

$$T_p^i(x_1, \ldots, x_i) = T_p^{i-1}(x_1, \ldots, x_{i-1}) \vee (T_{p-1}^{i-1}(x_1, \ldots, x_{i-1}) \wedge x_i),$$

we can compute all $T_p^i(x_1, \ldots, x_i)$ by at most $2nk$ gates and similarly also all $T_p^{n-i}(x_{i+1}, \ldots, x_n)$ by at most $2nk$ gates. Thus, $C_{\mathrm{m}}(T_k^{n-1}(X_1), \ldots, T_k^{n-1}(X_n)) \leq 6nk$.

By changing the roles of disjunction and conjunction, a monotone circuit for T_k^n becomes a monotone circuit for T_{n+1-k}^n. Thus, we obtain a similar algorithm with at most $6n(n+1-k)$ gates.

For k not too small and not too large, another approach is more efficient. It is well known that each comparator network can directly be used as a monotone circuit for the threshold functions. We assume w.l.o.g. that $n = 2^m$; in general, we have to add at most n dummy variables which are constant 0. Let $i = p2^r + q$ where $1 \leq q \leq 2^r$ and let $X_{i,r} := \{x_{p2^r+1}, \ldots, x_{(p+1)2^r}\}$ be the 2^r-block of variables containing x_i. Let $Y_{i,r}$ be the complement of $X_{i,r}$, i.e., $Y_{i,r} := X - X_{i,r}$, and let $Z_{i,r}$ be the buddy of $X_{i,r}$, i.e., $Z_{i,r} := X_{i,r+1} - X_{i,r}$. We have to compute the kth element of $Y_{i,0} = X_i$.

At first we use the well-known Batcher sorting network of size $\mathrm{O}(n \log^2 n)$ to sort X. By this network we sort also all $X_{i,r}$ and $Z_{i,r}$ $(1 \leq i \leq n,\ 1 \leq r \leq m)$. Since $Y_{i,r} = Y_{i,r+1} \dot{\cup} Z_{i,r}$, we may sort $Y_{i,r}$ by merging the sorted lists of $Y_{i,r+1}$ and $Z_{i,r}$. We are only interested in the kth element of $Y_{i,0}$ and not in the whole sorted list of $Y_{i,0}$. We claim that we only need 2^{r+1} elements of $Y_{i,r+1}$, namely only the elements with rank $k - 2^{r+1} + 1, \ldots, k$. If we merge these 2^{r+1} elements of $Y_{i,r+1}$ and the 2^r elements of $Z_{i,r}$, we get a sorted list of $3 \cdot 2^r$ elements. It is easy to see that the elements with rank $2^r + 1, \ldots, 2^{r+1}$ in this list are the elements with rank $k - 2^r + 1, \ldots, k$ in $Y_{i,r}$.

We start with the sets $Y_{i,m-1}$ which are the two halves of X and therefore already sorted by the Batcher network. For $r=m-2$ to 0 we compute for each $Y_{i,r}$ the elements with rank $k-2^r+1,\dots,k$ by merging the elements of $Y_{i,r+1}$ with rank $k-2^{r+1}+1,\dots,k$ and all elements of $Z_{i,r}$. This is done by a Batcher merging network whose size is $O(r2^r)$ for $r>0$. Since $Y_{i,r}=X-X_{i,r}$, there exist exactly $n2^{-r}=2^{m-r}$ different sets $Y_{i,r}$ for fixed r. The cost of the merging networks for fixed $r>0$ is therefore $O(r2^m)=O(nr)$. For $r=0$ the cost is $O(n)$. Since we have to perform this procedure for $r\in\{0,\dots,m-2\}$, the cost altogether is $O(nm^2)$. Thus we have shown that

$$C_m(T_k^{n-1}(X_1),\dots,T_k^{n-1}(X_n))=O(n\log^2 n). \qquad \square$$

Altogether we have shown that a function f with a slice whose monotone complexity is large has a hard slice and is therefore hard itself.

3. The number of slice functions

By a well-known counting argument due to Shannon [7] we know that nearly all Boolean functions have complexity $\Theta(2^n/n)$. By the results of Section 2 there have to exist hard slices. Furthermore, we know that nearly all monotone Boolean functions have circuit complexity $\Theta(2^n/n^{3/2})$ [5, 7]. What about slices? We consider only $\lceil\frac{1}{2}n\rceil$-slices. There are $\binom{n}{\lceil n/2\rceil}$ possible prime implicants. Each subset S defines another slice, namely the canonical slice of the disjunction of all monomes in S. Therefore, there are at least $2^{\binom{n}{\lceil n/2\rceil}}$ slice functions which yields the following proposition.

Proposition 3.1. *Nearly all slice functions have circuit complexity* $\Theta(2^n/n^{3/2})$.

Combining Propositions 2.5(i) and 3.1 we get the following.

Proposition 3.2. *Each function with complexity* $\Theta(2^n/n)$ *has at least* $\Omega(n^{1/2})$ *hard slices, namely slices of circuit complexity* $\Omega(2^n/n^2)$.

4. Monotone circuits for slice functions vs. set circuits

We have shown that it is important to investigate the monotone complexity of slice functions. Here we will show that the main structure of a monotone circuit for a slice function is given by a set circuit which we define later.

Let us consider a k-slice f. We are interested in monotone circuits for functions f' which compute 0 if the input has less than k ones, which equal f for k ones, and which are arbitrary for inputs with more than k ones. Since $f=f'\vee T_{k+1}^n$ we get again a monotone circuit for f whose cost is larger than the cost of the circuit for

f' by an additive term of at most $O(n \log n)$. For constant k this additive term can be reduced to $O(n)$ since $C_m(T_k^n) = O(n)$ if k is fixed. At first we manipulate the inputs.

Proposition 4.1. *If we replace in a monotone circuit for a k-slice f each variable input x_i by $x_i \wedge T_k^n$, the new circuit again computes f. The complexity of the new circuit is only by an additive term of $n + C_m(T_k^n)$ larger than the complexity of the given one.*

Proof. The second assertion is obvious. Since $x_i \wedge T_k^n \leq x_i$ and because of the monotonicity of the circuit the function f^* computed by the new circuit has the property $f^* \leq f$. Let us assume that, for some input $a, f^*(a) = 0$ and $f(a) = 1$. Then, again by monotonicity, $a_i = 1$ and $a_i \wedge T_k^n(a) = 0$ for some i. Thus, $T_k^n(a) = 0$ which implies, by the definition of a k-slice, $f(a) = 0$, a contradiction. □

Investigating the main structure of monotone circuits for slice functions we may assume w.l.o.g. that we have changed the circuit in the way described in Proposition 4.1. The effect of this transformation is that afterwards all functions computed in the circuit have prime implicants of length at least k only. Let $f' := \bigvee_{t \in \mathrm{PI}_k(f)} t$ where $\mathrm{PI}_k(f)$ is the set of all prime implicants of f of length k. Then, f' is one of the functions described at the beginning of this section where $f = f' \vee T_{k+1}^n$. In our monotone circuit for f we use now supergates instead of monotone gates. A supergate (super $\wedge$-gate or super $\vee$-gate) works at first like a normal $\wedge$- or $\vee$-gate and afterwards it destroys all prime implicants with more than k variables. By the first replacement rule of Mehlhorn and Galil [3] for monotone circuits, the new circuit computes f' instead of f. This can be easily shown by induction on the topological order of the gates. Instead of some function g we now compute everywhere $g' := \bigvee_{t \in \mathrm{PI}_k(g)} t$.

Altogether we now have a monotone (super-) circuit for f' where all prime implicants of all computed functions have length k. Let us consider the effect of supergates. Let g_1 and g_2 be two functions of the described class and let $g' :=$ super-$\vee$ (g_1, g_2) and $g'' :=$ super-$\wedge$ (g_1, g_2). We can conclude that $\mathrm{PI}(g') = \mathrm{PI}(g_1) \cup \mathrm{PI}(g_2)$. The property "$\subseteq$" always holds. Here, "$\supseteq$" holds too. The absorption rule cannot be applied since all prime implicants have the same length. Also, $\mathrm{PI}(g'') = \mathrm{PI}(g_1) \cap \mathrm{PI}(g_2)$. For $g := g_1 \wedge g_2$ we have

$$g = \left(\bigvee_{t \in \mathrm{PI}(g_1)} t \right) \wedge \left(\bigvee_{t' \in \mathrm{PI}(g_2)} t' \right).$$

All t and t' have length k. If $t \in \mathrm{PI}(g_1) \cap \mathrm{PI}(g_2)$, it is always a prime implicant of g too. Since it has length k, it is a prime implicant even of g''. All products tt' where $t \neq t'$ have length larger than k and become destroyed.

These observations motivate the following definition.

Definition 4.2. Let g be a monotone function whose prime implicants all have length k. A set circuit for g has inputs $x_i \wedge T_k^n$ for $1 \leq i \leq n$ and uses $\cap$- and $\cup$-gates. For two functions g_1 and g_2 whose prime implicants all have length k, $g' := g_1 \cup g_2$ and $g'' := g_1 \cap g_2$ are defined by

$$\mathrm{PI}(g') := \mathrm{PI}(g_1) \cup \mathrm{PI}(g_2) \quad \text{and} \quad \mathrm{PI}(g'') := \mathrm{PI}(g_1) \cap \mathrm{PI}(g_2).$$

The set complexity of g denoted by $\mathrm{SC_m}(g)$ is the minimal number of gates in a set circuit computing g.

Theorem 4.3. *Let f be a k-slice and $g := \bigvee_{t \in \mathrm{PI}_k(f)} t$.*

(i) $C_\mathrm{m}(f) \leq \mathrm{SC_m}(g) + \mathrm{O}(n \log n)$.

(ii) *If k is fixed,* $C_\mathrm{m}(f) \leq \mathrm{SC_m}(g) + \mathrm{O}(n)$.

(iii) $\mathrm{SC_m}(g) \leq C_\mathrm{m}(f)$.

Proof. (i) In order to compute f by a monotone circuit we compute all $x_i \wedge T_k^n$ (cost $\mathrm{O}(n \log n)$). We use afterwards an optimal set circuit for g, where the inputs are already computed, and replace all $\cap$ by $\wedge$ and all $\cup$ by $\vee$ (cost $\mathrm{SC_m}(g)$). We compute a function f' where all prime implicants have length at least k and where $\mathrm{PI}_k(f') = \mathrm{PI}_k(g) = \mathrm{PI}_k(f)$. Thus $f = f' \vee T_{k+1}^n$ and f can be computed from f' with cost $\mathrm{O}(n \log n)$.

(ii) The proof is similar to the proof of (i).

(iii) In order to compute g by a set circuit we use an optimal monotone circuit for f. We replace the inputs x_i by the inputs $x_i \wedge T_k^{n'}$ which are given for free in set circuits. Furthermore, we replace all $\vee$ by $\cup$ and all $\wedge$ by $\cap$. By our previous observations we obtain a set circuit for g. $\square$

Theorem 4.3 shows that the heart of a monotone circuit for a slice function is a set circuit. Thus, for investigating the monotone complexity of slice functions one should work in the model of set circuits.

To gain even more structure we make the following observations. Let us at first assume that we may partition the set of variables to k subsets and that each prime implicant of g contains exactly one variable of each subset. This property is fulfilled for the Boolean convolution or the Boolean matrix product for $k = 2$ and for the generalized Boolean matrix product [10] for arbitrary k. Let x_j^i denote a variable of the ith subclass. We can compute with $n - k$ gates all h_i, the disjunction of all variables in the ith subclass. Let $g_i := \bigwedge_{j \neq i} h_j$ for $1 \leq i \leq k$. These 'punctured conjunctions' can be computed with $p(k) = 3k - 6$ gates from the functions h_j. If k is even we compute with $\frac{1}{2}k$ gates the pairs $h_1 \wedge h_2, \ldots, h_{k-1} \wedge h_k$. With $p(\frac{1}{2}k)$ gates we compute the punctured conjunctions of these $\frac{1}{2}k$ terms. Afterwards, we need only one gate for the computation of each g_i. Thus $p(k) = p(\frac{1}{2}k) + \frac{3}{2}k$ if k is even. If k is odd, $\frac{1}{2}(k-1)$ gates are sufficient for the computation of the pairs $h_1 \wedge h_2, \ldots, h_{k-2} \wedge h_{k-1}$. For the recursion we have $\frac{1}{2}(k+1)$ terms, namely the $\frac{1}{2}(k-1)$ pairs and h_k. Afterwards, $k-1$ gates are sufficient since g_k is already computed. Thus $p(k) = p(\frac{1}{2}(k+1)) + \frac{3}{2}k - \frac{3}{2}$ if k is odd. Since $p(2) = 0$, we can conclude that $p(k) = 3k - 6$. Altogether, all g_i can be computed with $n+2k-6=O(n)$ gates. Proposition 4.1 and the appropriate version of Theorem 4.3 remain correct if we replace each x_j^i by $x_j^i \wedge g_i$. The advantage of this procedure is the following. The set of all possible prime implicants is the set of all combinations of one variable of each class, that means it is a k-dimensional discrete block or even a k-dimensional discrete cube if all classes have the same size. Each input is a $(k-1)$-dimensional subblock or subcube. The prime implicants form an arbitrary subset (pattern) of the points of the block or cube.

The problem of constructing optimal set circuits for g (or optimal monotone circuits for slice functions f) is therefore equivalent to the geometric problem of constructing the subset of points of a k-dimensional block formed by the prime implicants of g by intersections and unions of its $(k-1)$-dimensional subblocks.

Arbitrary functions with prime implicants of length k only can be changed in the following way in order to apply the geometric approach. The set of variables $\{x_1, \ldots, x_n\}$ is replaced by the kn variables x_j^i $(1 \leqslant i \leqslant k,\ 1 \leqslant j \leqslant n)$. The prime implicants $x_{i_1} \ldots x_{i_k}$ where $i_1 < \cdots < i_k$ are replaced by $x_{i_1}^1 \ldots x_{i_k}^k$.

We combine our results. We are interested in the circuit complexity of f. Instead of that we may investigate the circuit complexity of the slices of f. For slice functions it is nearly equivalent to consider monotone circuits. Finally, we have shown that this problem can be replaced by the investigation of the set complexity of $g := \bigvee_{t \in \mathrm{PI}_k(f_{\mathrm{sl}\text{-}k})} t$. On one hand, this last problem turned out to be the key problem but on the other hand we will show for some slices the existence of set circuits whose efficiency is remarkable.

5. The Boolean convolution

The canonical slice is that slice which at first sight is most similar to the given function. Here and in the following sections we show that the canonical slice may be much easier than the given function.

Definition 5.1. The Boolean convolution $f: \{0,1\}^{2n} \to \{0,1\}^{2n-1}$ on the set of variables $\{x_1, \ldots, x_n, y_1, \ldots, y_n\}$ is given by $f := \bigvee_{i+j=k} x_i y_j$ for $2 \leqslant k \leqslant 2n$.

Weiss [11] has shown that $C_\mathrm{m}(f) = \Omega(n^{3/2})$ and one conjectures that $C_\mathrm{m}(f) = \Theta(n^2)$.

Theorem 5.2. *The monotone complexity of the canonical slice of the Boolean convolution is linear.*

Proof. By Theorem 4.3 it is sufficient to prove $\mathrm{SC}_\mathrm{m}(f) = O(n)$. This follows from the facts that here the canonical slice is the 2-slice and that f equals the function g of Theorem 4.3.

We use the geometric approach and consider the square $\{1,\dots,n\}^2$ where the input $A_i = x_i \wedge (y_1 \vee \cdots \vee y_n)$ of the set circuit corresponds to the ith row of the square and $B_j = y_j \wedge (x_1 \vee \cdots \vee x_n)$ corresponds to the jth column. The output f_k corresponds to the k-diagonal of all (i,j) where $i+j=k$. We assume that $n=m^2$ for some natural number m. Otherwise, we could add some variables which we fix afterwards to 0. We use the following algorithm.

Step 1. $D_l := \bigcup_{1\le i\le m} A_{(l-1)m+i}$ $(1\le l\le m)$ (cost m^2-m).

Step 2. $E_l := \bigcup_{1\le i\le m} B_{(l-1)m+i}$ $(1\le l\le m)$ (cost m^2-m).

Step 3. $F_{ij} := D_i \cap E_j$ $(1\le i,j\le m)$ (cost m^2).

(F_{ij} is the subsquare (i,j) of side length m.)

Step 4. $G_l := \bigcup_{i+j=l} F_{ij}$ $(2\le l\le m)$ (cost m^2-2m+1).

(G_l is the l-diagonal of subsquares, see Fig. 1.)

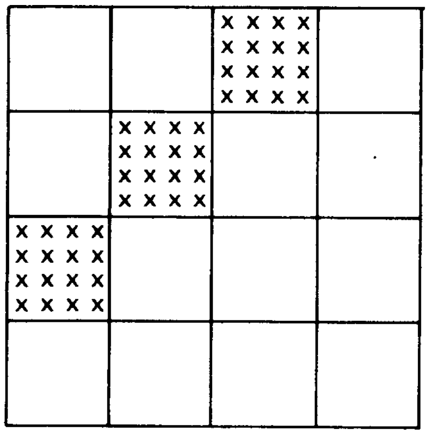

Fig. 1. $n=16$, G_4.

Step 5. $H_l := \bigcup_{1\le i\le m} A_{(i-1)m+l}$ $(1\le l\le m)$ (cost m^2-m).

Step 6. $I_l := \bigcup_{1\le i\le m} B_{(i-1)m+l}$ $(1\le l\le m)$ (cost m^2-m).

Step 7. $J_{ij} := H_i \cap I_j$ $(1\le i,j\le m)$ (cost m^2).

(J_{ij} is the pattern consisting of the (i,j)-elements of each subsquare.)

Step 8. $K_l := \bigcup_{i+j=l} J_{ij}$ $(2\le l\le m)$ (cost m^2-2m+1).

(K_l is the union of all l-diagonals of all subsquares, see Fig. 2.)

The set corresponding to f_k is $T_k := \bigcup_{i+j=k}(A_i \cap B_j)$, the k-diagonal of the whole square. A typical situation is given in Fig. 3. T_k touches one or two diagonals of

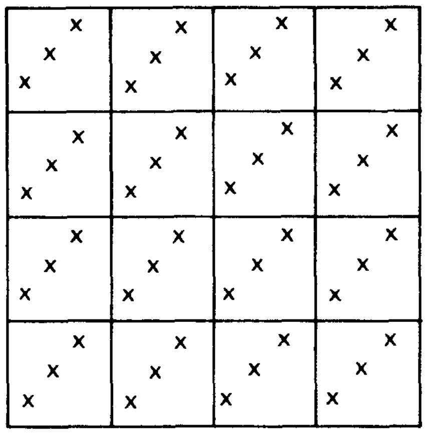

Fig. 2. $n=16$, K_4.

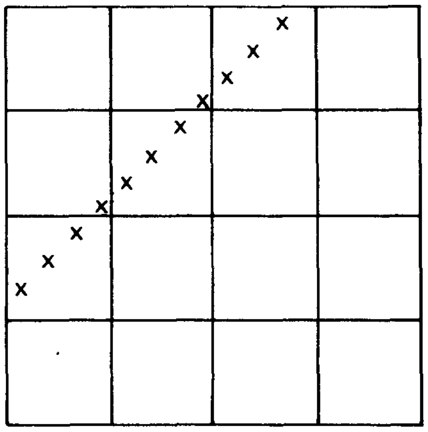

Fig. 3. $T_{12}=(G_3\cap K_8)\cup(G_4\cap K_4)$.

subsquares, say $G_{h(k)}$ and $G_{h(k)+1}$, in our figure the 3- and 4-diagonal. The intersection of T_k and $G_{h(k)}$ (respectively T_k and $G_{h(k)+1}$) is some diagonal of the corresponding subsquares, say the diagonal $d_1(k)$ of the subsquares of $G_{h(k)}$ and the diagonal $d_2(k)$ of the subsquares of $G_{h(k)+1}$ (in our figure, $d_1(12)=8$ and $d_2(12)=4$). Thus we may finish our algorithm by Step 9.

Step 9. $T_k:=(G_{h(k)}\cap K_{d_1(k)})\cup(G_{h(k)+1}\cap K_{d_2(k)})$, or

$T_k:=G_{h(k)}\cap K_{d(k)}$ $(2\leq k\leq 2n)$ (cost less than $3(2n-1)$).

The cost altogether is less than $14n-8m-1$. □

6. Boolean sums

Our example of Section 5 was a function with nearly the same number of outputs as variables. The function $f':=f_2\vee\cdots\vee f_{2n}$ equals $(x_1\vee\cdots\vee x_n)\wedge(y_1\vee\cdots\vee y_n)$ and is therefore an easy function. Here, we consider a similar function where we may prove a similar result for the n-output function but where one conjectures that also the disjunction of all outputs has monotone complexity $\Omega(n^{3/2})$.

Let us assume $n = p^2$ where p is a prime. Nechiporuk [4] constructed the following function $f: \{0,1\}^n \to \{0,1\}^n$. We consider an $n \times n$-matrix M of zeroes and ones. We partition this matrix to p^2 submatrices of size $p \times p$ each, which we denote by $M_{a,b}$. $M_{a,b}$ has exactly p ones as shown in Fig. 4 where $h := ab \bmod p$.

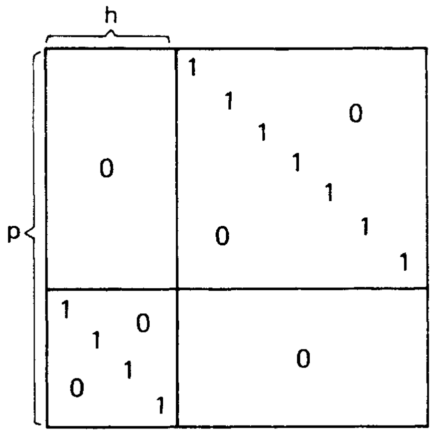

Fig. 4. $p = 11$, $a = 8$, $b = 6$, $h = ab \bmod p = 4$.

f_i corresponds to the ith row of M. f_i is the disjunction of all y_j such that the ith row of M has a one at position j. Nechiporuk [4] proved $C_m(f_1, \ldots, f_n) = \Theta(n^{3/2})$. If we let $g_i := x_i \wedge f_i$ for some new variables $x_1, \ldots, x_n$, obviously also $C_m(g_1, \ldots, g_n) = \Theta(n^{3/2})$. One conjectures that also for $g := g_1 \vee \cdots \vee g_n$ we have $C_m(g) = \Theta(n^{3/2})$.

Theorem 6.1. *The monotone circuit complexity of the canonical slice of* $(g_1, \ldots, g_n)$ *and of* g *is linear.*

Proof. Let $U, U_1, \ldots, U_n$ be the patterns corresponding to $g, g_1, \ldots, g_n$. We use the algorithm of Theorem 5.2 and compute with less than $14n$ gates in a set circuit all diagonals but here we take the diagonals from left to right. We need to compute the sets $N_{a,b}$ corresponding to the ones of the submatrices $M_{a,b}$ of Fig. 4. This can be done by taking the union of two diagonals and afterwards the intersection with $M_{a,b}$. $M_{a,b}$ corresponds to some F_{ij} of the algorithm of Theorem 5.2. That means that two gates are sufficient for each submatrix, altogether $2n$ gates. Since U is the union of all $N_{a,b}$, we may construct U with another $n-1$ gates. Finally, U_i is the intersection of U and the ith row which is given as input. Altogether, $SC_m(g) \leq 17n$ and $SC_m(g_1, \ldots, g_n) \leq 18n$. $\square$

7. Clique functions

Definition 7.1. The k-clique function $f_k: \{0,1\}^N \to \{0,1\}$ where $N = \binom{n}{2}$ is defined in the following way. The variables are denoted by x_{ij} where $1 \leq i < j \leq n$. x_{ij} corresponds to the possible edge between i and j in an n-vertex graph G. f_k computes 1 iff the graph specified by the variables x_{ij} contains a clique of size k.

It is well known that the clique problem is NP-complete. Therefore, one believes that there does not exist any polynomial size circuit for the clique function if k is chosen in an appropriate way. Valiant [8] has shown an exponential lower bound for monotone circuits of three logical levels. Yao [12] improved this result to four levels. If the conjecture above is correct, the clique function must have hard slices. We prove that the canonical slice is easy to compute. Therefore, one has to raise the question which slices may be hard. The canonical slice is the K-slice where $K = \binom{k}{2}$.

Theorem 7.2. *The circuit complexity of the canonical slice of any clique-function is* $O(N)$ *and its monotone complexity is* $O(N \log N)$. *For fixed k even the monotone complexity is* $O(N)$.

Proof. The results on the monotone complexity follow as always in the same way.

Let f^* be the canonical slice of the k-clique function on graphs with n vertices. By Definition 2.3,

$$f^* = (f \wedge E_K^N) \vee T_{K+1}^N = (f \wedge T_K^N) \vee T_{K+1}^N.$$

T_K^N and T_{K+1}^N can be computed with $O(N)$ gates. In the above formula for f^* we may replace f by any function g which coincides with f on the set of inputs where $E_K^N(x) = 1$. Let $X_i := \{x_{1,i}, \ldots, x_{i-1,\,i}, x_{i,i+1}, \ldots, x_{i,n}\}$. We shall replace f by $T_k^n(T_{k-1}^{n-1}(X_1), \ldots, T_{k-1}^{n-1}(X_n))$. Each $T_{k-1}^{n-1}(X_i)$ and afterwards the function T_k^n may be computed with $O(n)$ gates, therefore we only need $O(n^2) = O(N)$ gates for g. Altogether, we have proved the theorem if the replacement is correct. The justification of the replacement is given by the proof of the following graph-theoretical claim.

Graphs with exactly K edges contain a k-clique if and only if at least k vertices have outdegree at least $k-1$.

"*Only if*": The k-clique has K edges. That means a graph with K edges and a k-clique contains exactly the edges of the k-clique. The k vertices of the clique have outdegree $k-1$.

"*If*": Since the graph has K edges and since each edge joins two vertices, the sum over all outdegrees is $2K = k(k-1)$. If at least k vertices have outdegree at least $k-1$, there are exactly k vertices with outdegree $k-1$ and all other $n-k$ vertices are isolated. A vertex with outdegree $k-1$ is connected to $k-1$ other vertices. The only possible vertices are the $k-1$ other nonisolated vertices. Thus the k nonisolated vertices form a k-clique. □

8. Methods for proving lower bounds

Until now we have not been able to prove nonlinear lower bounds for the circuit complexity, monotone complexity or set complexity of one-output functions in NP. Even for n-output-functions we do not know any nonlinear bound on the circuit complexity. The results of this paper show the importance of proving lower bounds on the monotone complexity or set complexity of slice functions. We discuss here which of the known methods for nonlinear lower bounds on the monotone complexity of n-output-functions may work also for slice functions.

All those methods used for lower bounds for the Boolean convolution or Boolean sums are no longer available in their pure form as has been shown in Theorems 5.2 and 6.1.

Let us consider the important replacement rules due to Mehlhorn and Galil [3]. The first replacement rule, the elimination of prime implicants which have no lengthening which is a prime implicant of some output, becomes unimportant. For k-slices this rule cannot be applied for monomes of length at most k and it is not interesting to destroy monomes of length larger than k since they are implicants. In set circuits we have this replacement rule implicitly.

The second replacement rule cannot be applied in the same successful way as before. We will show this for the Boolean matrix product

$$c_{ij} := \bigvee_{1 \leq k \leq n} a_{ik} b_{kj} \quad (1 \leq i, j \leq n)$$

and its canonical slice. We cite this replacement rule.

If a monotone circuit computes f and if it computes at some gate s

(i) $tt_1, tt_2 \in \mathrm{PI}(s)$, and

(ii) $\forall$monome $\bar{t}$ $\forall$output f_k: $\bar{t}tt_1, \bar{t}tt_2 \in I(f_k) \Rightarrow \bar{t}t \in I(f_k)$, where $I(f_k)$ is the set of implicants of f_k,

then we may replace s by $s \vee t$ and the circuit still computes f.

For the Boolean matrix product the replacement rule can be applied. For example, if $t = 1$, $t_1 = a_{i1}$, $t_2 = a_{i'1}$ where $i \neq i'$, s can be replaced by the constant 1 and the given circuit was not optimal. For the canonical slice of the Boolean matrix product the second assumption of the replacement rule is no longer correct. (One may easily generalize this counter-example to other slices.) Let $\bar{t}$ be a monome consisting of two variables which do not form a prime implicant of the matrix product and which are unequal to a_{i1} and $a_{i'1}$. $\bar{t}tt_1$ and $\bar{t}tt_2$ are implicants of each output since they contain three variables. But $\bar{t}t = \bar{t}$ is not an implicant of any output.

The other important methods are the elimination method and the method of using value functions (for a discussion, see [10]). These methods may be valuable for slice functions too. But until now one has used these methods in particular after having restructured the given circuit by the replacement rules discussed above.

Thus we still do not know any nonlinear lower bound on the monotone complexity even of n-output slice functions.

9. Conclusion

We have seen that monotone circuits and in particular set circuits are important tools for the proof of lower bounds on the circuit complexity of hard Boolean functions. The investigation of monotone circuits is sometimes easier than the investigation of circuits over a complete basis. But we have indicated that for all models it seems to be difficult to prove lower bounds for slice functions. Nevertheless, the concept of considering the slice functions of a given function is an important new concept.

Acknowledgment

I like to thank Les Valiant, Mike Paterson, Rüdiger Reischuk, and Friedhelm Meyer auf der Heide for valuable discussions about this paper.

References

[1] M. Ajtai, J. Komlós and E. Szemerédi, An O(n log n) sorting network, *Proc. 15th STOC* (1983) 1-9.
[2] S. Berkowitz, On some relationships between monotone and non-monotone circuit complexity, Thesis, Univ. of Toronto, 1982.
[3] K. Mehlhorn and Z. Galil, Monotone switching circuits and Boolean matrix product, *Computing* **16** (1976) 99-111.
[4] E. I. Nechiporuk, On a Boolean matrix, *Systems Res. Theory* **21** (1971) 236-239.
[5] N. Pippenger, The complexity of monotone Boolean functions, *Mathematical Systems Theory* **11** (1978) 289-316.
[6] J. E. Savage, *The Complexity of Computing* (Wiley, New York, 1976).
[7] C. E. Shannon, The synthesis of two-terminal switching circuits, *Bell Systems Techn. J.* **28** (1949) 59-98.
[8] L. G. Valiant, Exponential lower bounds for restricted monotone circuits, *Proc. 15th STOC* (1983) 110-117.
[9] L. G. Valiant, Negation is powerless for Boolean slice functions, Techn. Rept., Harvard Univ., 1984.
[10] I. Wegener, Boolean functions whose monotone complexity is of size $n^2/\log n$, *Theoret. Comput. Sci.* **21** (1982) 213-224.
[11] J. Weiss, An $\Omega(n^{3/2})$ lower bound on the monotone complexity of Boolean convolution, *Inform. Control* **59** (1983) 184-188.
[12] A. C. Yao, Lower bounds by probabilistic arguments, *Proc. 24th FOCS* (1983) 420-428.