Electronic Colloquium on Computational Complexity - Reports Series 1996 - available via:

ECCC
TR96-022

FTP: ftp.eccc.uni-trier.de:/pub/eccc/
WWW: http://www.eccc.uni-trier.de/eccc/
Email: ftpmail@ftp.eccc.uni-trier.de with subject 'help eccc'

Optimal Ordered Binary Decision Diagrams for Tree-like Circuits

Martin Sauerhoff Ingo Wegener Ralph Werchner

FB Informatik, LS II
Univ. Dortmund
44221 Dortmund
Germany

FB Mathematik
Univ. Frankfurt a. M.
60054 Frankfurt
Germany

Abstract

Many Boolean functions have short representations by OBDDs (ordered binary decision diagrams) if appropriate variable orderings are used. For tree-like circuits, which may contain EXOR-gates, it is proved that some depth first traversal leads to an optimal variable ordering. Moreover, an optimal variable ordering and the resulting OBDD can be computed in time linear in the number of variables and the size of the OBDD, respectively. Upper and lower bounds on the OBDD size of the functions representable by tree-like circuits are derived. For, e. g., 1024 inputs, we show that all tree-like circuits have OBDDs of size at 5349 and we give an example of a tree-like circuit requiring an OBDD of size 5152.

Email Addresses
sauerhof/wegener@ls2.informatik.uni-dortmund.de
werchner@mi.informatik.uni-frankfurt.de

Acknowledgements
The first and second author have been supported by DFG grant We 1066/7-3.

1. INTRODUCTION

Various types of *decision diagrams* (DDs) are often used in CAD systems for efficient representation and manipulation of Boolean functions. Areas of applications are among others verification of combinational and sequential circuits, automata, models, and protocols, timing analysis, and test pattern generation. The most popular data structure are *ordered binary decision diagrams* (OBDDs) introduced by Bryant [2] who has also written an excellent survey article [3].

In [2], Bryant has shown that the OBDD size is very sensitive to the variable ordering. Since it is NP-hard to compute an optimal variable ordering, many heuristics have been proposed. Early attempts (Butler, Ross, Kapur, and Mercer [4], Jeong, Plessier, Hachtel, and Somenzi [7], Malik, Wang, Brayton, and Sangiovanni-Vincentelli [8], and Ross, Butler, Kapur, and Mercer[12]) compute a variable ordering directly from the circuit representation. These variable orderings are often not good enough and only used as starting point for local search and simulated annealing algorithms (Bollig, Löbbing, and Wegener[1], Fujita, Matsunga, and Kakuda [5], Ishiura, Sawada, and Yajima [6], Mercer, Kapur, and Ross [9], Panda and Somenzi [11], and Rudell [13]). Very good results are obtained by the group sifting algorithm [11]. Nevertheless, it is important to choose a good starting ordering in order to reduce the number of sifting operations.

Another problem is to increase the class of circuits such that optimal variable orderings can be computed efficiently. This may also lead to new heuristics. For tree-like circuits over the basis of binary gates of type AND (AND, OR, NAND, NOR, $\overline{x}y$, $x\overline{y}$, $\overline{x}+y$, $x+\overline{y}$) it is easy to see that OBDDs according to an arbitrary DFS variable ordering have the optimal number of exactly n nonterminal nodes, where n is the number of variables. A *DFS variable ordering* is obtained by a depth first search traversal starting at the primary output. If we also allow EXOR- and NEXOR-gates, different DFS variable orderings may lead to OBDDs of different sizes. It is not at all obvious that *some* DFS variable ordering is optimal. This situation is the starting point of our paper. Tree-like circuits are basic types of circuits and interesting for their own sake. Furthermore, results on tree-like circuits may lead to better heuristics for almost tree-like circuits.

In Section 2, OBDDs, DFS variable orderings, and tree-like circuits are defined formally. In Section 3, it is proved that some optimal OBDD variable ordering for functions represented by a tree-like circuit over the full binary basis is a DFS variable ordering. This structural result and its proof lead directly to a theoretically and practically efficient algorithm for the computation of an optimal variable ordering for tree-like circuits and the corresponding OBDD. An optimal variable ordering can be computed in linear time with respect to the size of the tree-like circuit. The corresponding OBDD can be computed in linear time with respect to its size. With these results it is no problem to treat tree-like circuits of up to 10^5 inputs.

Moreover, it is desirable to have a bound on the size of optimal OBDDs for functions representable by tree-like circuits. In Section 4, we prove such a bound, which is less than $1.36n^\beta$ for $\beta = \log_4(3+\sqrt{5}) < 1.1943$. E.g., for $n = 1024$, we obtain the bound 5349. In Section 5, we present tree-like circuits such that the optimal OBDDs for the represented functions have size $\Theta(n^\beta)$. For e.g. $n = 1024$, this function needs 5152 nodes.

2. DEFINITIONS AND KNOWN RESULTS

OBDDs use the syntax of general ordered decision diagrams.

Definition 1: A *variable ordering* of the variables $x_1, \ldots, x_n$ is a permuted sequence $x_{\pi(1)}, \ldots, x_{\pi(n)}$ for some permutation π on $\{1, \ldots, n\}$. An *ordered decision diagram* respecting the variable ordering $x_{\pi(1)}, \ldots, x_{\pi(n)}$ is a rooted directed acyclic graph $G = (V, E)$. The node set V contains two terminal nodes labeled with 0 and 1 and non-terminal nodes, each labeled with one of the variables. Each non-terminal node v has exactly two successors denoted by $low(v)$ and $high(v)$. If a successor of a node with label x_i has the label x_j, then x_i has to precede x_j in $x_{\pi(1)}, \ldots, x_{\pi(n)}$.

The following definition explains the semantics of OBDDs.

Definition 2: The terminal nodes of OBDDs represent the constant functions given by their labels. A non-terminal node v with label x_i represents the function $f_v := \overline{x_i} f_{low(v)} + x_i f_{high(v)}$ (*Shannon decomposition*).

It is well-known that OBDDs are a canonical form of Boolean functions if the variable ordering is fixed. I.e., the OBDD of minimal size is unique and called reduced. For further purposes it is essential to describe which functions are represented at the nodes of reduced OBDDs. A function f is said to *depend essentially* on a variable x_i, if the *subfunctions* (*cofactors*) $f_{|x_i=0}$ and $f_{|x_i=1}$ are not equal.

Theorem 1: Let $x_1, \ldots, x_n$ be the given variable ordering and let f be defined on $x_1, \ldots, x_n$. The reduced OBDD for f contains as many x_i-nodes as there are different subfunctions $f_{|x_1=a_1,\ldots,x_{i-1}=a_{i-1}}$, for $a_1, \ldots, a_{i-1} \in \{0, 1\}$, which depend essentially on x_i.

The proof of Theorem 1 is due to Sieling and Wegener [14].

In this paper, we consider the full binary basis B_2^* of all ten binary operations depending essentially on both inputs, namely xy (AND), $x+y$ (OR), $\overline{xy}$ (NAND), $\overline{x+y}$ (NOR), $\overline{x}y$, $x\overline{y}$, $\overline{x}+y$, $x+\overline{y}$, $x \oplus y$ (EXOR) and $\overline{x \oplus y}$ (NEXOR).

Definition 3: A *tree-like circuit* is a binary tree with n leaves labeled by different variables $x_1, \ldots, x_n$. The inner nodes are gates labeled by functions from B_2^*.

Applying de Morgan's rules and the fact that $\overline{x \oplus y} = \overline{x} \oplus y$, we can push all negations towards the inputs of the tree-like circuit without altering the resulting function. Eliminating the negations of the inputs changes the function, however, the structure of the OBDD is preserved. Substituting $\overline{x_i}$ with x_i corresponds to switching the *high-* and *low-*successor of every x_i-node in the OBDD. This justifies that, subsequently, we will only consider circuits over the basis {AND, OR, EXOR }.

Definition 4: A *DFS variable ordering* for a tree-like circuit is a variable ordering obtained by some depth first traversal starting at the primary output of the circuit.

Obviously, there are 2^{n-1} depth first traversals leading to different DFS variable orderings. All these orderings are known to be optimal for OBDDs if the tree-like circuit does not contain EXOR- or NEXOR-gates. It is easy to see that in the more general case of the full binary basis different DFS orderings may lead to different OBDD sizes.

3. AN EFFICIENT ALGORITHM FOR THE COMPUTATION OF AN OPTIMAL VARIABLE ORDERING FOR OBDDS AND TREE-LIKE CIRCUITS

All heuristics for the computation of good variable orderings suggest to use some DFS variable ordering for tree-like circuits. But we know for each known heuristic of a tree-like circuit where this heuristic does not construct an optimal variable ordering. Since the variable ordering problem has a lot of surprising features, we cannot be sure that we can restrict ourselves to DFS variable orderings. First we prove that, searching for an optimal variable ordering for a tree-like circuit, it is sufficient to consider DFS variable orderings only. This proof changes a well established hypothesis into a theorem. But the proof has even more consequences. It is the basis of an efficient algorithm for the computation of an optimal variable ordering.

Let f be a Boolean function computed by a tree-like circuit. Then we represent f as

$$f(x_1, \ldots, x_k, y_1, \ldots, y_m) = g(x_1, \ldots, x_k) \otimes h(y_1, \ldots, y_m)$$

for some functions g and h represented by tree-like circuits and a binary Boolean operator $\otimes$. A naive approach is the following. If $(x_1, \ldots, x_k)$ and $(y_1, \ldots, y_m)$ are optimal variable orderings for g and h resp., perhaps even DFS variable orderings, then $(x_1, \ldots, x_k, y_1, \ldots, y_m)$ or $(y_1, \ldots, y_m, x_1, \ldots, x_k)$ is an optimal variable ordering for f.

But we have to be careful. If $\otimes = \oplus$, an OBDD for f according to $(x_1, \ldots, x_k, y_1, \ldots, y_m)$ consists of an OBDD for g at the top and an SBDD (*shared* BDD, see Minato, Ishiura and Yajima [10]) for $(h, \overline{h})$ at the bottom. Thus in the next stage of our analysis, we need to consider SBDDs for $(h, \overline{h})$ too. Can this lead to more and more cases in subsequent stages? We prove that for each function f' represented in the circuit we only have to consider OBDDs for f', OBDDs for $\overline{f'}$ and SBDDs for $(f', \overline{f'})$. Hence, we only have to consider a limited number of cases. Moreover, size(f') = size$(\overline{f'})$, where size$(f_1, \ldots, f_r)$ is the minimal SBDD size (number of non-terminal nodes) for $(f_1, \ldots, f_r)$ according to an arbitrary variable ordering, and the same variable orderings which are optimal for f' are also optimal for $\overline{f'}$.

Lemma 1: Let $f(x_1, \ldots, x_k, y_1, \ldots, y_m) = g(x_1, \ldots, x_k) \otimes h(y_1, \ldots, y_m)$ for some $\otimes \in B_2^*$. Then there is an optimal OBDD variable ordering for f where all x-variables are tested before all y-variables or vice versa. The same holds for $(f, \overline{f})$.

Proof: Since negations have no effect on the OBDD size, it is sufficient to consider the two cases where $\otimes = \wedge$ and $\otimes = \oplus$. Let π be a variable ordering of the variables of f. W.l.o.g. the first variable in π is an x-variable if $\otimes = \oplus$, and the last variable in π is a y-variable if $\otimes = \wedge$. Then we claim that the following variable ordering π' is at least as good as π. With respect to π' we start with all x-variables in the same order as prescribed by π followed by all y-variables in the same order as prescribed by π. After renumbering, we can assume that π' is the variable ordering $(x_1, \ldots, x_k, y_1, \ldots, y_m)$. Let G be the reduced OBDD (SBDD) for f $((f, \overline{f}))$ according to π and G' the same for π'. We claim that G contains for each variable z at least as many z-nodes as G'. We prove the claim applying Theorem 1, which holds for SBDDs as well. We consider eight cases distinguishing whether we investigate f or $(f, \overline{f})$, $\wedge$ or $\oplus$, and x- or y-nodes.

Case 1 $(f, \wedge, x_i)$.
There is some $j \in \{0, \ldots, m\}$ such that, by Theorem 1, the number of x_i-nodes in G is equal to the number of different functions

$$f_{|x_1=a_1,\ldots,x_{i-1}=a_{i-1},y_1=b_1,\ldots,y_j=b_j} = g_{|x_1=a_1,\ldots,x_{i-1}=a_{i-1}} \wedge h_{|y_1=b_1,\ldots,y_j=b_j}$$

depending essentially on x_i. The number of x_i-nodes in G' is equal to the number of different functions

$$f_{|x_1=a_1,\ldots,x_{i-1}=a_{i-1}} = g_{|x_1=a_1,\ldots,x_{i-1}=a_{i-1}} \wedge h$$

depending essentially on x_i. Since h depends essentially on all its variables, we can choose $b_1, \ldots, b_j \in \{0, 1\}$ such that $h_{|y_1=b_1,\ldots,y_j=b_j}$ is not the constant 0. Already for this replacement of the y-variables by constants we obtain in G as many x_i-nodes as in G'.

Case 2 $((f, \overline{f}), \wedge, x_i)$.
Here we have to consider f and $\overline{f}$. For some j, the number of x_i-nodes in G is equal to the number of different subfunctions

$$g_{|x_1=a_1,\dots,x_{i-1}=a_{i-1}} \wedge h_{|y_1=b_1,\dots,y_j=b_j} \quad \text{and} \quad \overline{g_{|x_1=a'_1,\dots,x_{i-1}=a'_{i-1}} \wedge h_{|y_1=b'_1,\dots,y_j=b'_j}}$$

essentially depending on x_i. Note that $j < m$ by the above assumption and the fact that we are in the case $\otimes = \wedge$. Since $b_1, \dots, b_j$ can be chosen such that $h_{|y_1=b_1,\dots,y_j=b_j}$ is not the constant 0 or 1, the number of x_i-nodes in G is at least twice the number of different subfunctions $g_{|x_1=a_1,\dots,x_{i-1}=a_{i-1}}$ essentially depending on x_i. On the other hand, the number of x_i-nodes in G' is exactly twice this number.

Cases 3 $(f, \oplus, x_i)$.
For some j, the number of x_i-nodes in G is equal to the number of different subfunctions

$$g_{|x_1=a_1,\dots,x_{i-1}=a_{i-1}} \oplus h_{|y_1=b_1,\dots,y_j=b_j}$$

essentially depending on x_i. But this is at least the number of different subfunctions $g_{|x_1=a_1,\dots,x_{i-1}=a_{i-1}} \oplus h$ essentially depending on x_i, which is the number of x_i-nodes in G'.

Cases 4 $((f, \overline{f})), \oplus, x_i)$.
The number of x_i-nodes in G' is equal to the number of different subfunctions

$$g_{|x_1=a_1,\dots,x_{i-1}=a_{i-1}} \oplus h \quad \text{and} \quad \overline{g_{|x_1=a'_1,\dots,x_{i-1}=a'_{i-1}} \oplus h}$$

essentially depending on x_i. This is equal to the number of different subfunctions

$$g_{|x_1=a_1,\dots,x_{i-1}=a_{i-1}} \quad \text{and} \quad \overline{g_{|x_1=a'_1,\dots,x_{i-1}=a'_{i-1}}}$$

essentially depending on x_i. The number of x_i-nodes in G is at least that much, even considering just restrictions with $y_1 = \dots = y_j = 0$.

Case 5 $(f, \wedge, y_j)$.
There is some i such that, by Theorem 1, the number of y_i-nodes in G is equal to the number of different functions

$$f_{|x_1=a_1,\dots,x_i=a_i,y_1=b_1,\dots,y_{j-1}=b_{j-1}} = g_{|x_1=a_1,\dots,x_i=a_i} \wedge h_{|y_1=b_1,\dots,y_{j-1}=b_{j-1}}$$

depending essentially on y_j. The number of y_j-nodes in G' is equal to the number of different functions

$$f_{|x_1=a_1,\dots,x_k=a_k,y_1=b_1,\dots,y_{j-1}=b_{j-1}} = g_{|x_1=a_1,\dots,x_k=a_k} \wedge h_{|y_1=b_1,\dots,y_{j-1}=b_{j-1}}$$

depending essentially on y_j. Obviously, $g_{|x_1=a_1,\dots,x_k=a_k}$ is a constant. If the constant is 0, the corresponding subfunction of f cannot depend essentially on y_j. Otherwise we consider subfunctions of h. Similarly to Case 1, it is sufficient to choose $(a_1, \ldots, a_i)$ such that $g_{|x_1=a_1,\ldots,x_i=a_i}$ is not the constant 0.

Case 6 $((f, \overline{f}), \wedge, y_j)$.
For some i, the number of y_i-nodes in G is equal to the number of different subfunctions

$$g_{|x_1=a_1,\ldots,x_i=a_i} \wedge h_{|y_1=b_1,\ldots,y_{j-1}=b_{j-1}} \text{ and } \overline{g_{|x_1=a'_1,\ldots,x_i=a'_i} \wedge h_{|y_1=b'_1,\ldots,y_{j-1}=b'_{j-1}}}$$

essentially depending on y_j. If $i < k$, we can use the arguments of Case 2. If $i = k$, in both G and G', the same set of variables is tested before y_i. Hence, the number of y_j-nodes in G is the same as in G'.

Case 7 $(f, \oplus, y_j)$.
Here we obtain as many y_i-nodes in G' as there are different functions $h_{|y_1=b_1,\ldots,y_{j-1}=b_{j-1}}$ and $h_{|y_1=b_1,\ldots,y_{j-1}=b_{j-1}} \oplus 1$ depending essentially on y_j. Now we apply the assumption (remembering that we are in the case $\otimes = \oplus$) that the first variable according to π is an x-variable, i.e. in G we consider the functions $g_{|x_1=a_1,\ldots,x_i=a_i} \oplus h_{|y_1=b_1,\ldots,y_{j-1}=b_{j-1}}$ for some $i \geq 1$. Since g depends essentially on all its variables, we obtain at least two different subfunctions $g_{|x_1=a_1,\ldots,x_i=a_i}$ leading to the same number of y_j-nodes in G as in G'.

Case 8 $((f, \overline{f}), \oplus, y_j)$.
We can apply the arguments of Case 7. In G', we get the same subfunctions for $\overline{f}$ as for f. □

The assumptions of Lemma 1 are fulfilled for tree-like circuits. Nevertheless, we cannot apply Lemma 1 directly to the subcircuits. We have to discuss which functions have to be represented in the two parts of the OBDD for f and the SBDD for $(f, \overline{f})$. W.l.o.g. we consider a variable ordering where the x-variables are tested before the y-variables. The results of the following case inspection are illustrated in Fig. 1.

Case 1 $(f, \wedge)$.
The OBDD G for f starts with an OBDD G_1 for g. The 0-sink of G_1 is identified with the 0-sink of G, while its 1-sink is identified with the source of an OBDD G_2 for h. Considering also the dual case (y-variables before x-variables) we obtain

$$\text{size}(f) = \text{size}(g) + \text{size}(h).$$

Case 2 $((f, \overline{f}), \wedge)$.
Here, $f = g \wedge h$ and $\overline{f} = \overline{g} + \overline{h}$. The SBDD G for $(f, \overline{f})$ starts with disjoint OBDDs G_1 and $\overline{G_1}$ for g and $\overline{g}$. The 0-sink of G_1 is identified with the 0-sink of G, and the 1-sink of $\overline{G_1}$ is identified with the 1-sink of G. The 1-sink of G_1 is identified with the source for h of an SBDD G_2 for $(h, \overline{h})$, and the 0-sink of $\overline{G_1}$ is identified with the source for $\overline{h}$ of

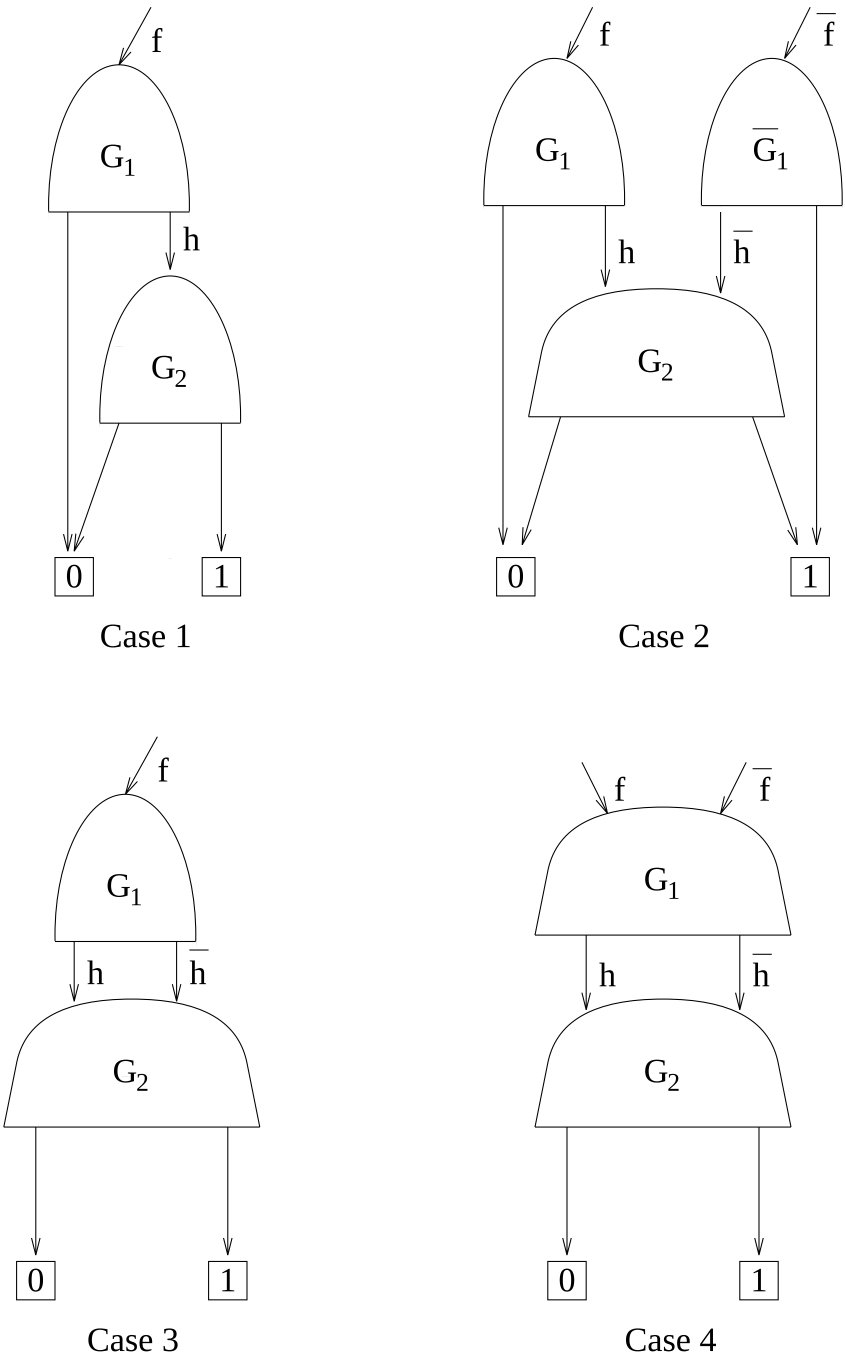

Figure 1: OBDDs and SBDDs for f and $(f, \overline{f})$ resp. for the different types of gates.

G_2. The resulting SBDD is reduced as we will show now. It is sufficient to prove that x-nodes cannot be merged. If not, $f_{|x_1=a_1,\ldots,x_{i-1}=a_{i-1}} = \overline{f_{|x_1=a'_1,\ldots,x_{i-1}=a'_{i-1}}}$ for some constants $a_1, a'_1, \ldots, a_{i-1}, a'_{i-1} \in \{0,1\}$. This is impossible since there exists some b with $h(b) = 0$. Considering the dual case and applying the fact that $\text{size}(g) = \text{size}(\overline{g})$, we obtain

$$\text{size}(f,\overline{f}) = \min\{2 \cdot \text{size}(g) + \text{size}(h,\overline{h}), 2 \cdot \text{size}(h) + \text{size}(g,\overline{g})\}.$$

Note that the results of the first two cases also hold for $f = g \vee h$ since $\text{size}(\overline{f}) = \text{size}(f)$.

Case 3 $(f, \oplus)$.
The OBDD G for f starts with an OBDD G_1 for g. The 0-sink of G_1 is identified with the source for h of an SBDD G_2 for $(h,\overline{h})$, and the 1-sink of G_1 is identified with the source for $\overline{h}$ of G_2. Hence, considering also the dual case we obtain

$$\text{size}(f) = \min\{\text{size}(g) + \text{size}(h,\overline{h}), \text{size}(h) + \text{size}(g,\overline{g})\}.$$

Case 4 $((f,\overline{f}), \oplus)$.
The SBDD G for $(f,\overline{f})$ starts with an SBDD G_1 for $(g,\overline{g})$. The source for g in G_1 becomes the source for f in G and the source for $\overline{g}$ the source for $\overline{f}$. The 0-sink of G_1 is identified with the source for h of an SBDD G_2 for $(g,\overline{g})$, and the 1-sink is identified with the source for $\overline{h}$. Hence, considering also the dual case we obtain

$$\text{size}(f,\overline{f}) = \text{size}(g,\overline{g}) + \text{size}(h,\overline{h}).$$

Now we know a lot more. The upper and lower part of optimal OBDDs and SBDDs for tree-like circuits lead to the same type of problem as the given one. In particular, we can apply Lemma 1 to these subproblems.

Theorem 2: If f is represented by a tree-like circuit, there exists a DFS ordering which is an optimal OBDD variable ordering for both f and $(f,\overline{f})$.

Proof: We only have to prove that the same DFS variable ordering is optimal for f and $(f,\overline{f})$. This follows from our case inspection. For each gate type there is only one case where we have to decide whether it is better to test the x-variables before the y-variables or vice versa. For the other case both decisions are optimal. □

Our considerations directly lead to an efficient algorithm for the computation of an optimal variable ordering for a tree-like circuit. For each gate `g`, we compute the size `g.size1` of an optimal OBDD for the function computed at `g` (which is also denoted by g) and the size `g.size2` of an optimal SBDD for $(g,\overline{g})$. Furthermore, we compute an optimal variable ordering described as list of variables. This can be done easily in constant time, if the corresponding results are known for both inputs of g. We only have to apply the results of the above case inspection. Finally, for an input x_i the optimal variable ordering is x_i, its OBDD size is 1, and the SBDD size of $(x_i, \overline{x_i})$ is 2. We can treat the gates of a tree-like circuit in the usual order (given, e. g., as SLIF file) or by a recursive approach starting at the primary output. In Algorithm 1 we have followed the second approach. Here we also have to consider negations in order to realize the full binary basis.

Algorithm 1: `VarOrder(g: gate)`, **returns** `list of variables`

```
list, list1, list2:  list of variables;
case
  g = xi:
    g.size1:=1; g.size2:=2;
    list:={xi};
  g = ¬g1:
    list:=VarOrder(g1);
    g.size1:=g1.size1; g.size2:=g1.size2;
  g = g1 ∨ g2, g = g1 ∧ g2:
    list1:=VarOrder(g1);
    list2:=VarOrder(g2);
    g.size1:=g1.size1 + g2.size1;
    s1:=2·g1.size1 + g2.size2;
    s2:=2·g2.size1 + g1.size2;
    if s1 < s2 then g.size2:=s1; list:=list1 + list2
             else g.size2:=s2; list:=list2 + list1
    fi;
  g = g1 ⊕ g2:
    list1:=VarOrder(g1);
    list2:=VarOrder(g2);
    g.size2:=g1.size2 + g2.size2;
    s1:=g1.size1 + g2.size2;
    s2:=g2.size1 + g1.size2;
    if s1 < s2 then g.size1:=s1; list:=list1 + list2
             else g.size1:=s2; list:=list2 + list1
    fi;
esac;
return list;

end VarOrder
```

Theorem 3: Algorithm 1 computes an optimal OBDD variable ordering for a tree-like circuit on n variables in time and space $O(n)$.

Proof: We only need time $O(1)$ for each gate if we ensure by a pointer to the end of each list that we can append lists in constant time, which gives the claimed time bound. For the space bound, we remark that we create lists only for the variables. Later we append lists, i. e. we automatically destroy the lists for the predecessors. Hence, the total length of all lists is always bounded by n. □

Knowing an optimal variable ordering, we can construct the corresponding OBDD with the well-known synthesis algorithm of Bryant [2]. All gates of a tree-like circuit represent subfunctions of the primary output. Hence, the optimal OBDD for the function computed at some gate is always smaller than the optimal OBDD for the primary output. The synthesis algorithm is very fast in practice but it does not guarantee a run time linear in the size of the resulting OBDD. Our inspection of the four different cases illustrated in Fig. 1 is also a basis of a direct construction of the OBDD. Then we can guarantee to construct the OBDD in linear time with respect to its size. However, we do not recommend to implement this approach, since the synthesis algorithm is fast enough for all practical purposes. Nevertheless, it is a theoretically interesting result that for tree-like circuits and a DFS ordering the OBDD directly can be constructed without synthesis algorithm in linear time.

Usually, one does not work with OBDDs in its pure form presented here. Minato, Ishiura, and Yajima [10] have introduced OBDDs with complemented edges. We show that, for tree-like circuits, there is a DFS variable ordering which is an optimal variable ordering for OBDDs with and without complemented edges. For this purpose, we use the notation size_{ce} for OBDDs with complemented edges and $\text{size}(\cdot, \pi)$ for the size of reduced π-OBDDs.

Theorem 4: For each Boolean function f and each variable ordering π it holds that $\text{size}(f, \overline{f}, \pi) = 2 \cdot \text{size}_{ce}(f, \pi)$. If f is represented by a tree-like circuit, the variable ordering computed by Algorithm 1 is optimal for OBDDs with and without complemented edges.

Proof: The second assertion follows from the first one. The first one implies that the same variable orderings are optimal for SBDDs without complemented edges for $(f, \overline{f})$ and for OBDDs with complemented edges for f. The variable ordering computed by Algorithm 1 is optimal for OBDDs for f and for SBDDs for $(f, \overline{f})$.

We prove the first assertion w. l. o. g. for the variable ordering $x_1, \ldots, x_n$. Let us denote by $f_1, \ldots, f_r$ the different subfunctions $f_{|x_1=a_1,\ldots,x_{k-1}=a_{k-1}}$ depending essentially on x_k. If we construct the x_k-level of the reduced π-OBDD with complemented edges, we can save the node for f_j, if for some $i < j$, we have $f_i = \overline{f_j}$. Hence, we get $r - s$ nodes if s is the number of indices j such that $f_i = \overline{f}_j$, for some $i < j$. If we construct the x_k-level of the reduced π-SBDD for $(f, \overline{f})$, we can start with $2r$ nodes for $f_1, \ldots, f_r, \overline{f}_1, \ldots, \overline{f}_r$. We can save a node if $f_i = \overline{f}_j$. But then also $f_j = \overline{f}_i$. Hence, we can save $2s$ nodes, and $\text{size}(f, \overline{f}, \pi) = 2r - 2s = 2 \cdot \text{size}_{ce}(f, \pi)$. □

4. AN UPPER BOUND ON THE OBDD SIZE FOR FUNCTIONS WITH TREE-LIKE CIRCUITS

By the previous results we are able to efficiently compute optimal variable orderings for OBDDs and functions represented by tree-like circuits. Now we present upper bounds on the OBDD size for all functions on n variables representable by tree-like circuits. The best known bound $n^{\log 3} \leq n^{1.585}$ due to Wegener [15] is improved significantly.

As before, let f be computed by a tree-like circuit where the output gate computes $f = g \otimes h$. It has been shown in Section 3 that both $\text{size}(f)$ and $\text{size}(f, \bar{f})$ depend on all of $\text{size}(g)$, $\text{size}(g, \bar{g})$, $\text{size}(h)$, and $\text{size}(h, \bar{h})$. To prove an asymptotically sharp upper bound on $\text{size}(f)$, we have to consider both measures $\text{size}(f)$ and $\text{size}(f, \bar{f})$ simultaneously. We will inductively prove a bound on $\varphi(\text{size}(f), \text{size}(f, \bar{f}))$, for a suitable function $\varphi : \mathbb{R}_+^2 \to \mathbb{R}_+$. Let

$$\varphi(s, t) = \max(a_1 s + a_2 t, a_1' s + a_2' t, a_1'' s + a_2'' t),$$

where $a_1, a_2, a_1', a_2', a_1'', a_2'' \in \mathbb{R}_+$ are defined in the Appendix. In the sequel, we will show that $\text{size}(f) = O(n^\beta)$ with $\beta = \log_4(3 + \sqrt{5}) < 1.1943$.

We obtain the main technical tool with the following lemma which is proved in the Appendix.

Lemma 2: Let $s, t, \tilde{s}, \tilde{t}$ be nonnegative real numbers with

$$\begin{aligned} s \leq t &\leq 2s \\ \tilde{s} \leq \tilde{t} &\leq 2\tilde{s} \\ \varphi(s, t) &\leq m^\beta \\ \varphi(\tilde{s}, \tilde{t}) &\leq \tilde{m}^\beta. \end{aligned}$$

Then we have

$$\varphi(s + \tilde{s}, 2s + \tilde{t}) \leq (m + \tilde{m})^\beta \text{ if } \quad s < \tilde{s} \quad \text{or} \quad s = \tilde{s} \wedge \tilde{t} \leq t$$

and

$$\varphi(s + \tilde{t}, t + \tilde{t}) \leq (m + \tilde{m})^\beta \text{ if } \quad \tilde{t} < t \quad \text{or} \quad \tilde{t} = t \wedge s \leq \tilde{s}.$$

Lemma 3: Let f be a function on n inputs representable by a tree-like circuit. Then we have

$$\varphi(\text{size}(f), \text{size}(f, \bar{f})) \leq \varphi(1,2)\, n^{\beta}.$$

Proof: We prove the lemma by induction on n. For $n = 1$ the claim holds since, in this case, $\text{size}(f) \leq 1$ and $\text{size}(f, \bar{f}) \leq 2$.

Let $n > 1$ and let f be computed by $f = g \otimes h$, where g and h are functions on k and m variables respectively. Set $\bar{\alpha} = \varphi(1,2)$. By the induction hypothesis we have

$$\begin{aligned}
\varphi(\text{size}(g), \text{size}(g, \bar{g})) &\leq (\bar{\alpha}^{1/\beta}\, k)^{\beta} \\
\varphi(\text{size}(h), \text{size}(h, \bar{h})) &\leq (\bar{\alpha}^{1/\beta}\, m)^{\beta}.
\end{aligned}$$

We distinguish two cases according to $\otimes$.

Case 1 $\otimes = \wedge$.

We assume w.l.o.g. that

$$\begin{aligned}
& \text{size}(g) < \text{size}(h) \\
\text{or}\quad & \text{size}(g) = \text{size}(h) \wedge \text{size}(h, \bar{h}) \leq \text{size}(g, \bar{g}).
\end{aligned}$$

Then the Cases 1 and 2 in Section 3 together with the first inequality of Lemma 2 yield

$$\begin{aligned}
\varphi(\text{size}(f), \text{size}(f, \bar{f})) &\leq \varphi(\text{size}(g) + \text{size}(h), 2\ \text{size}(g) + \text{size}(h, \bar{h})) \\
&\leq (\bar{\alpha}^{1/\beta} k + \bar{\alpha}^{1/\beta} m)^{\beta} \\
&= \bar{\alpha}\, n^{\beta}.
\end{aligned}$$

Case 2 $\otimes = \oplus$.

We assume w.l.o.g. that

$$\begin{aligned}
& \text{size}(h, \bar{h}) < \text{size}(g, \bar{g}) \\
\text{or}\quad & \text{size}(h, \bar{h}) = \text{size}(g, \bar{g}) \wedge \text{size}(g) \leq \text{size}(h).
\end{aligned}$$

Then the Cases 3 and 4 in Section 3 together with the second part of Lemma 2 yield

$$\begin{aligned}
\varphi(\text{size}(f), \text{size}(f, \bar{f})) &\leq \varphi(\text{size}(g) + \text{size}(h, \bar{h}), \text{size}(g, \bar{g}) + \text{size}(h, \bar{h})) \\
&\leq (\bar{\alpha}^{1/\beta} k + \bar{\alpha}^{1/\beta} m)^{\beta} \\
&= \bar{\alpha}\, n^{\beta}.
\end{aligned}$$

□

Theorem 5: Let f be a function on n inputs representable by a tree-like circuit. Setting $\alpha = \varphi(1,2)/\varphi(1,1)$, we have $\text{size}(f) \le \alpha\, n^\beta$.

Proof: By the preceding lemma, $\varphi(\text{size}(f), \text{size}(f,\bar{f})) \le \varphi(1,2)\, n^\beta$. Using the fact that $\varphi(1,y)$ is monotonically increasing in y and that, for $s,t,c \in \mathbb{R}_+$, $\varphi(c\,s, c\,t) = c\,\varphi(s,t)$, we have

$$\begin{aligned} \text{size}(f) &= \varphi(\text{size}(f), \text{size}(f,\bar{f}))\, \varphi(1, \text{size}(f,\bar{f})/\,\text{size}(f))^{-1} \\ &\le \varphi(1,2)\, n^\beta\, \varphi(1,1)^{-1} \\ &= \alpha\, n^\beta. \end{aligned}$$

□

In the Appendix it is shown how we can fix the parameters in such a way that $\alpha < 1.35916$.

Theorem 6: Let f be a function on n inputs representable by a tree-like circuit. Then f can be represented by an OBDD with complemented edges of size at most n^β.

Proof: We have

$$\begin{aligned} \text{size}(f,\bar{f}) &= \varphi(\text{size}(f), \text{size}(f,\bar{f}))\, \varphi(\text{size}(f)/\,\text{size}(f,\bar{f}), 1)^{-1} \\ &\le \varphi(1,2)\, n^\beta \varphi(0.5, 1)^{-1} \\ &= 2\, n^\beta. \end{aligned}$$

The claim follows with the result of Theorem 4. □

In Table 1 we present concrete values of our upper bounds.

number of variables n	old upper bound for OBDDs $n^{\log 3}$	new upper bound for OBDDs αn^β	upper bound for OBDDs with compl. edges n^β
32	243	85	62
64	729	195	143
128	2187	446	328
256	6561	1021	751
512	19683	2337	1719
1024	59049	5349	3935

Table 1: Upper bounds for OBDDs for functions representable by tree-like circuits.

Our improvement of the upper bound for OBDDs is significant. Moreover, we have obtained an even better bound for OBDDs with complemented edges. The use of complemented edges decreases the upper bound by approximately 26.4%. We conclude that functions with tree-like circuits have small OBDDs which can be computed efficiently.

5. TREE-LIKE CIRCUITS WITH "LARGE" OBDDS

We have seen that no tree-like circuit requires "really large" OBDDs. Now we are interested in relatively hard tree-like circuits for a given number of inputs. The results will show how tight the upper bounds of Section 4 are.

We restrict ourselves to circuits consisting of $\wedge$- and $\oplus$-gates only. In general, $\oplus$-gates are more difficult than $\wedge$-gates. Moreover, the proofs of the upper bounds suggest to use tree-like circuits based on balanced trees. We investigate complete binary trees with alternating levels of $\wedge$- and $\oplus$-gates, where the last gate is an $\oplus$-gate. Since $x_1 \oplus x_2 \oplus x_3 \oplus x_4$ is harder than $x_1x_2 \oplus x_3x_4$ for OBDDs without complemented edges, we have decided to sometimes start with two levels of $\oplus$-gates. This leads to the following function, called Reed-Muller-tree RMT, since the Reed-Muller decomposition rule is based on $\wedge$- and $\oplus$-gates.

Definition 5: RMT_n is defined on $n = 2^k$ variables. If $n = 1$, $RMT_1(x_1) = x_1$. Otherwise, the tree-like circuit is a complete binary tree with k gate levels. The root is an $\oplus$-level. If k is odd, the levels alternate between $\oplus$ and $\wedge$. If k is even, the levels also are alternating with the exception that the first level below the leaves consists also of $\oplus$-gates.

Theorem 7: Let $n = 2^k$, $r = 3 + \sqrt{5}$, $s = 3 - \sqrt{5}$, $A_1 = \frac{1}{2}\left(3\sqrt{5} + 7\right)$, $B_1 = \frac{1}{2}\left(7 - 3\sqrt{5}\right)$, $A_2 = \frac{1}{10}\left(7\sqrt{5} + 15\right)$, and $B_2 = \frac{1}{10}\left(15 - 7\sqrt{5}\right)$. Then the minimal OBDD size of RMT_n fulfills

$$\text{size}(RMT_n) = \begin{cases} A_1 r^{k/2-1} + B_1 s^{k/2-1} & \text{if } k \text{ is even} \\ A_2 r^{(k-1)/2} + B_2 s^{(k-1)/2} & \text{if } k \text{ is odd.} \end{cases}$$

Moreover, $\text{size}(RMT_n) = \Theta(n^{\beta})$ for $\beta = \log_4(3 + \sqrt{5})$.

Proof: Because of the symmetry of the circuit the DFS variable ordering $x_1, \ldots, x_n$ enumerating the leaves from left to right is optimal. It is sufficient to compute the size of the resulting OBDD. Let S_k be this OBDD size for $n = 2^k$ and T_k the corresponding SBDD size for $(RMT_n, \overline{RMT_n})$. We know by case inspection that $S_1 = 3$, $T_1 = 4$, $S_2 = 7$, and $T_2 = 8$.

Now we apply the results of Section 3 and consider the last two levels of the circuit for RMT_n. Hence,

$$RMT_n(u, v, x, y) = \left(RMT_{n/4}(u) \wedge RMT_{n/4}(v)\right) \oplus \left(RMT_{n/4}(x) \wedge RMT_{n/4}(y)\right).$$

We abbreviate this as $f = (f_1 \wedge f_2) \oplus (f_3 \wedge f_4)$. Then, by the results of Section 3,

$$\begin{aligned} \text{size}(f) &= \text{size}(f_1 \wedge f_2) + \text{size}\left((f_3 \wedge f_4), (\overline{f_3 \wedge f_4})\right) \\ &= \text{size}(f_1) + \text{size}(f_2) + 2 \cdot \text{size}(f_3) + \text{size}(f_4, \overline{f_4}). \end{aligned}$$

Hence,

$$S_k = 4S_{k-2} + T_{k-2}.$$

Also by the results of Section 3

$$\begin{aligned} \text{size}(f, \overline{f}) &= \text{size}\left((f_1 \wedge f_2), (\overline{f_1 \wedge f_2})\right) + \text{size}\left((f_3 \wedge f_4), (\overline{f_3 \wedge f_4})\right) \\ &= 2 \cdot \text{size}(f_1) + \text{size}(f_2, \overline{f_2}) + 2 \cdot \text{size}(f_3) + \text{size}(f_4, \overline{f_4}). \end{aligned}$$

Hence,

$$T_k = 4S_{k-2} + 2T_{k-2}.$$

The exact solution for S_k follows with standard techniques. □

We only remark that we also are able to compute exact formulas for T_k. This is interesting since $T_k/2$ is the size of the optimal OBDD with complemented edges for RMT_n and $n = 2^k$. We present the exact results for $k \in \{1, \ldots, 10\}$.

k	1	2	3	4	5	6	7	8	9	10
S_k	3	7	16	36	84	188	440	984	2304	5152
$T_k/2$	2	4	10	22	52	116	272	608	1424	3184

Table 2: S_k is the minimal OBDD size for RMT_n and $n = 2^k$, $T_k/2$ the corresponding value for OBDDs with complemented edges.

For $n = 512$ we know that RMT_n needs OBDDs with 2304 nodes and no function representable by a tree-like circuit needs more than 2337 nodes. Hence, our results are very precise. For $n = 1024$ the bounds are not so close, 5152 vs. 5349. This can be explained by the formula for size(RMT_n) in Theorem 7. Since $s < 1$, the second term is small. The first term equals

$$(A_1/r)n^\beta \approx 1.309n^\beta, \text{ if } k \text{ is even,}$$

and

$$(A_2/r^{\frac{1}{2}})n^\beta \approx 1.340n^\beta, \text{ if } k \text{ is odd.}$$

CONCLUSION

The variable ordering problem is crucial for successful applications of OBDDs. Variable orderings are computed with simple heuristic algorithms on the circuits and then improved with sifting, group sifting, and simulated annealing. Better initial variable orderings reduce the cost for the improvement steps. In order to better understand the variable ordering problem, circuits of the simplest structure namely tree-like circuits are investigated. The computation of an optimal variable ordering is easy if we consider the restricted basis of gates of type AND. Otherwise, it has been a lot of work to prove the hypothesis that some DFS variable ordering is optimal. All known heuristics compute on some tree-like circuit a nonoptimal variable ordering. A careful case inspection leads to an algorithm which computes in linear time and space an optimal variable ordering. General tree-like circuits are the first nontrivial case where such an algorithm is presented. The size of optimal OBDDs for functions with tree-like circuits cannot become large even if the full binary basis is available. But there are functions not representable by OBDDs of linear size.

ACKNOWLEDGEMENTS

We thank Bernd Becker and Rolf Drechsler for many stimulating discussions on the subject of this paper.

REFERENCES

[1] B. Bollig, M. Löbbing, and I. Wegener. Simulated annealing to improve variable orderings for OBDDs. In Int'l Workshop on Logic Synthesis, pp. 5.1-5.10, 1995.

[2] R. E. Bryant. Graph-based algorithms for Boolean function manipulation. IEEE Trans. on Comp., vol. C-35(8), pp. 677–691, 1986.

[3] R. E. Bryant. Symbolic boolean manipulation with ordered binary decision diagrams. ACM Comp. Surveys, vol. 24, pp. 293–318, 1992.

[4] K. M. Butler, D. E. Ross, R. Kapur, and M. R. Mercer. Heuristics to compute variable orderings for efficient manipulation of ordered binary decision diagrams. In Design Automation Conf., pp. 417–420, 1991.

[5] M. Fujita, Y. Matsunga, and T. Kakuda. On variable ordering of binary decision diagrams for the application of multi-level synthesis. In European Design Automation Conf., pp. 50–54, 1991.

[6] N. Ishiura, H. Sawada, and S. Yajima. Minimization of binary decision diagrams based on exchanges of variables. In Int'l Conf. on CAD, pp. 472-475, 1991.

[7] S.-W. Jeong, B. Plessier, G. D. Hachtel, and F. Somenzi. Variable ordering and selection for FSM traversal. In Int'l Conf. on CAD, pp. 476–479, 1991.

[8] S. Malik, A. R. Wang, R. K. Brayton, and A. L. Sangiovanni-Vincentelli. Logic verification using binary decision diagrams in a logic synthesis environment. In Int'l Conf. on CAD, pp. 6–9, 1988.

[9] M. R. Mercer, R. Kapur, and D. E. Ross. Functional approaches to generating orderings for efficient symbolic representation. In Design Automation Conf., pp. 614–619, 1992.

[10] S. Minato, N. Ishiura, and S. Yajima. Shared binary decision diagrams with attributed edges for efficient boolean function manipulation. In Design Automation Conf., pp. 52–57, 1990.

[11] S. Panda and F. Somenzi. Who are the variables in your neighborhood. In Int'l Conf. on CAD, pp. 74–77, 1995.

[12] D. E. Ross, K. M. Butler, R. Kapur, and M. R. Mercer. Fast functional evaluation of candidate OBDD variable orderings. In European Design Automation Conf., pp. 4–10, 1991.

[13] R. Rudell. Dynamic variable ordering for ordered binary decision diagrams. In Int'l Conf. on CAD, pp. 42–47, 1993.

[14] D. Sieling and I. Wegener. NC-algorithms for operations on binary decision diagrams. Parallel Processing Letters, vol. 3, pp. 3–12, 1993.

[15] I. Wegener. Comments on "A characterization of binary decision diagrams." IEEE Trans. on Comp., vol. C-43(4), pp. 383–384, 1994.

APPENDIX

We still have to prove the rather technical Lemma 2. We have some freedom to choose the parameters in the definition of φ. Remember that

$$\varphi(s,t) = \max(a_1 s + a_2 t, a_1' s + a_2' t, a_1'' s + a_2'' t)$$

and $\beta = \log_4(3+\sqrt{5}) = \log_2(\sqrt{3+\sqrt{5}})$. The upper bound in Theorem 5 is $\frac{\varphi(1,2)}{\varphi(1,1)} n^\beta$. Hence, we do not have to care about constant factors of φ.

Minimizing $\frac{\varphi(1,2)}{\varphi(1,1)} n^\beta$ subject to the validity of Lemma 2, we obtain the following choice of parameters.

Let

$$\begin{aligned} p_1 &= \sqrt{3+\sqrt{5}} \Big/ 2 = 2^{\beta-1} \\ p_2 &= (2+\sqrt{5}) \Big/ \sqrt{3+\sqrt{5}} = (2+\sqrt{5}) 2^{-\beta} \\ q_1 &= (3+\sqrt{5}) \Big/ 4 = 4^{\beta-1} \\ q_2 &= (1+\sqrt{5}) \Big/ 2. \end{aligned}$$

Then we define $a_1, a_2, a_1', a_2', a_1'', a_2''$ as the unique solution of the the following system of linear equations:

$$\begin{aligned} a_1 p_1 + a_2 p_2 &= 1 \\ a_1' p_1 + a_2' p_2 &= 1 \\ a_1' q_1 + a_2' q_2 &= 1 \\ a_1'' q_1 + a_2'' q_2 &= 1 \\ a_1'' &= 0 \\ a_1 + 2\,a_2 &= 2^{-\beta}\,(3\,a_1' + 4\,a_2') \end{aligned}$$

One can verify that $a_1, a_2, a_1', a_2', a_1'', a_2''$ are nonnegative.

We only consider φ on the region of all (s,t) with $s \le t \le 2s$. This region is shown in Fig. 2. One can verify that the set of all (s,t) with $\varphi(s,t) = 1$ consists of three segments which meet in (p_1, p_2) and (q_1, q_2), as shown in Fig. 2. Therefore, we divide the considered region into three sectors by the lines $t = (p_2/p_1)s$ and $t = (q_2/q_1)s$. We number the sectors from bottom to top by I, II, and III. Note that φ is linear within each sector.

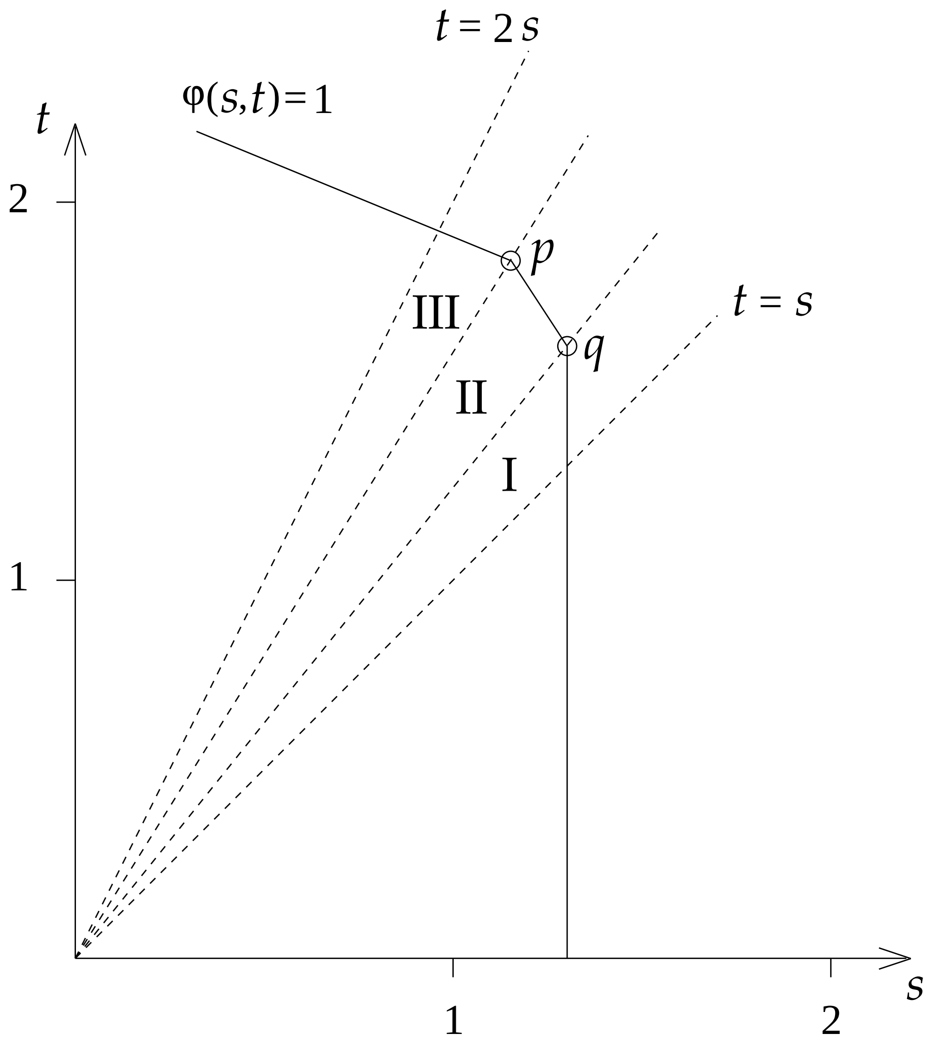

Figure 2: The positive quadrant with $p = (p_1, p_2)$ and $q = (q_1, q_2)$ and the set of (s, t) with $\varphi(s, t) = 1$.

Lemma 2: Let $s, t, \tilde{s}, \tilde{t}$ be nonnegative real numbers with

$$\begin{aligned} s \le t &\le 2s \\ \tilde{s} \le \tilde{t} &\le 2\tilde{s} \\ \varphi(s,t) &\le m^\beta \\ \varphi(\tilde{s},\tilde{t}) &\le \tilde{m}^\beta. \end{aligned}$$

Then we have

$$\varphi(s+\tilde{s}, 2s+\tilde{t}) \le (m+\tilde{m})^\beta \text{ if } \quad \begin{array}{l} s < \tilde{s} \\ \text{or } \; s = \tilde{s} \wedge \tilde{t} \le t \end{array}$$

and

$$\varphi(s+\tilde{t}, t+\tilde{t}) \le (m+\tilde{m})^\beta \text{ if } \quad \begin{array}{l} \tilde{t} < t \\ \text{or } \; \tilde{t} = t \wedge s \le \tilde{s}. \end{array}$$

Proof: We start with the first inequality. From now on assume that $s, \tilde{s}, t, \tilde{t}$ fulfill the condition of the first claim of the lemma. We have to show that $\mu(s, \tilde{s}, t, \tilde{t})$ is nonnegative, where $\mu : \mathbb{R}_+^4 \to \mathbb{R}$ is defined by

$$\mu(s,\tilde{s},t,\tilde{t}) = \left(\varphi(s,t)^{1/\beta} + \varphi(\tilde{s},\tilde{t})^{1/\beta}\right)^\beta - \varphi(s+\tilde{s}, 2s+\tilde{t}).$$

Denote by $\varphi_1(s,t)$ the partial derivative of $\varphi(s,t)$ with respect to s and, accordingly, denote by $\varphi_2(s,t)$ the partial derivative with respect to t. μ is monotonically increasing in $\tilde{s}$ since it is continuous and, on every line in $\tilde{s}$-direction, $\frac{\partial\mu}{\partial\tilde{s}}$ is defined on all but at most 4 points with

$$\begin{aligned} \frac{\partial\mu}{\partial\tilde{s}} &= \beta\left(\varphi(s,t)^{1/\beta} + \varphi(\tilde{s},\tilde{t})^{1/\beta}\right)^{\beta-1} \frac{1}{\beta}\varphi(\tilde{s},\tilde{t})^{1/\beta-1}\varphi_1(\tilde{s},\tilde{t}) \\ &\quad -\varphi_1(s+\tilde{s}, 2s+\tilde{t}) \\ &= \left(1 + \left(\frac{\varphi(s,t)}{\varphi(\tilde{s},\tilde{t})}\right)^{1/\beta}\right)^{\beta-1} \varphi_1(\tilde{s},\tilde{t}) - \varphi_1(s+\tilde{s}, 2s+\tilde{t}) \\ &\ge \varphi_1(\tilde{s},\tilde{t}) - \varphi_1(s+\tilde{s}, 2s+\tilde{t}) \\ &\ge 0. \end{aligned}$$

The last inequality follows from the fact that, in the positive quadrant, moving in the direction of the vector $(1,2)$ never increases φ_1. The function φ_1 is constant within each of the sectors I, II, and III, with the smallest value in I and the largest value in III. Since the lines bounding the sectors have a slope of at most 2, the last inequality is correct. Thus,

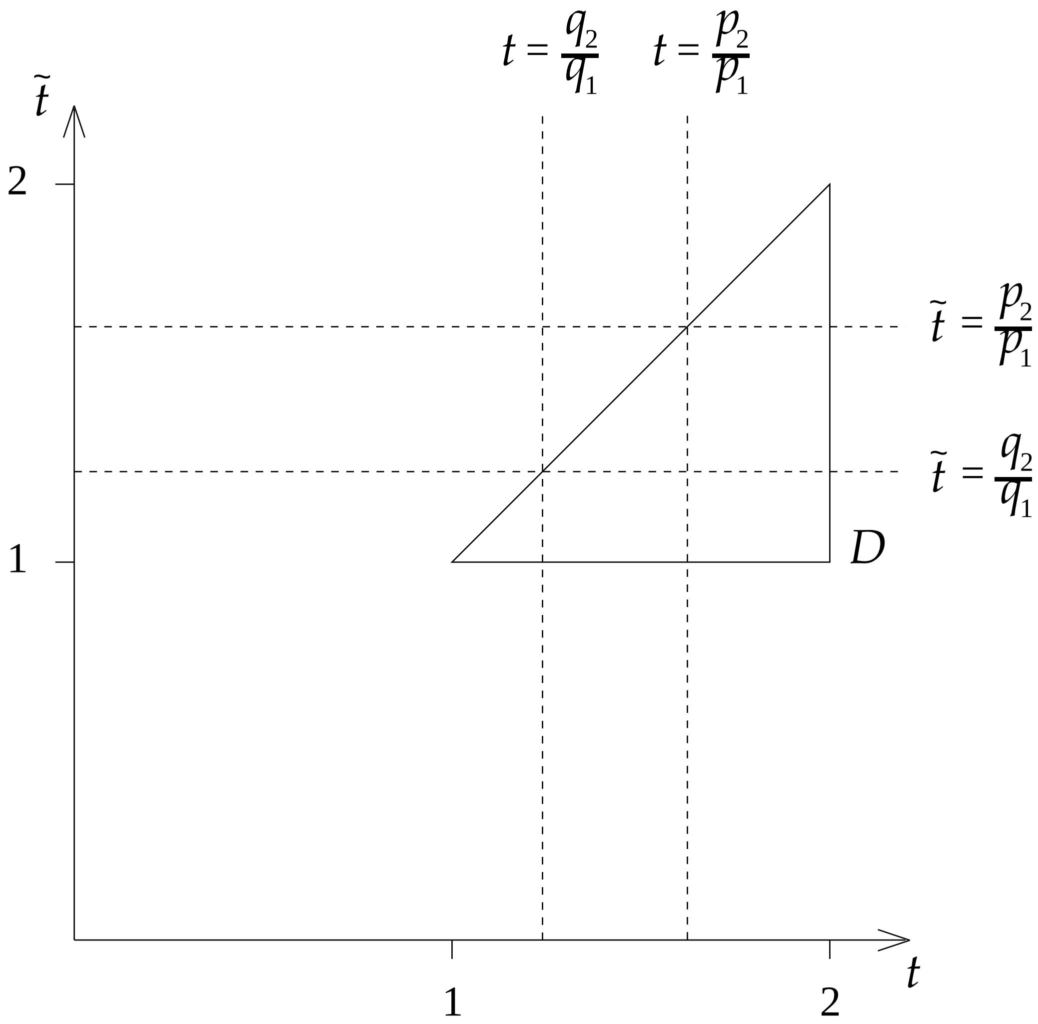

Figure 3: The triangle D divided into the 6 regions.

we can further restrict ourselves to the case of $s = \tilde{s}$, $\tilde{t} \le t$. Since μ is multiplicative, we can even assume $s = \tilde{s} = 1$.

Now define $\hat{\mu}$ on the triangle $D \subset \mathbb{R}^2_+$ with the corners $(1,1),(2,1),(2,2)$ by

$$\begin{aligned}\hat{\mu}(t,\tilde{t}) &= \mu(1,1,t,\tilde{t}) \\ &= \left(\varphi(1,t)^{1/\beta} + \varphi(1,\tilde{t})^{1/\beta}\right)^{\beta} - \varphi(2,2+\tilde{t}).\end{aligned}$$

It remains to show that $\hat{\mu}$ is nonnegative on D. As shown in Fig. 3, the four lines $t = q_2/q_1$, $t = p_2/p_1$, $\tilde{t} = q_2/q_1$, and $\tilde{t} = p_2/p_1$ partition D into 6 regions, which are triangles and rectangles. Restricted to each of those regions, $\hat{\mu}$ is a "linear transformation" of the function $\psi : \mathbb{R}^2_+ \to \mathbb{R}_+$ defined by

$$\psi(z,\tilde{z}) = \left(z^{1/\beta} + \tilde{z}^{1/\beta}\right)^{\beta}.$$

This means, there are linear functions l_1, l_2, l_3 such that

$$\hat{\mu}(t,\tilde{t}) = l_1(t,\tilde{t}) + \psi(l_2(t), l_3(\tilde{t}))$$

(note that also $\varphi(2, 2+\tilde{t})$ is linear on each of the 6 regions since $2 + q_2/q_1 = 2\,p_2/p_1$). Since ψ is concave on $\mathbb{R}^2$ (the matrix of second order partial derivatives is negative semi-definite), $\hat{\mu}$ is concave on each of the 6 regions of D. Thus the minimum value of $\hat{\mu}$ on D appears on a corner of one of the 6 regions.

It is $\hat{\mu}(q_2/q_1, q_2/q_1) = 0$. With the help of a computer one easily verifies that $\hat{\mu}$ is positive on the other 9 corners of the regions. Thus, μ is nonnegative on D and we have proved the first inequality of the lemma.

The second claim of the lemma is proved analogously. Assume that $s, \tilde{s}, t, \tilde{t}$ fulfill the condition of the second claim. Setting

$$\nu(s,\tilde{s},t,\tilde{t}) = \left(\varphi(s,t)^{1/\beta} + \varphi(\tilde{s},\tilde{t})^{1/\beta}\right)^{\beta} - \varphi(s+\tilde{t}, t+\tilde{t})$$

we need to show that $\nu(s,\tilde{s},t,\tilde{t}) \ge 0$. Since

$$\begin{aligned}\frac{\partial \nu}{\partial t} &= \left(1 + \left(\frac{\varphi(\tilde{s},\tilde{t})}{\varphi(s,t)}\right)^{1/\beta}\right)^{\beta-1} \varphi_2(s,t) - \varphi_2(s+\tilde{t}, t+\tilde{t}) \\ &\ge \varphi_2(s,t) - \varphi_2(s+\tilde{t}, t+\tilde{t}) \\ &\ge 0,\end{aligned}$$

ν is monotonically increasing in t. Again, the last inequality follows from the fact that moving in direction of $(1,1)$ never increases φ_2. Thus we can restrict ourselves to $t = \tilde{t} = 1$ and $s \leq \tilde{s}$. Define $\hat{\nu}$ on the triangle $D' \subset \mathbb{R}^2_+$ with the corners $(0.5, 0.5), (0.5, 1), (1, 1)$ by

$$\begin{aligned} \hat{\nu}(s, \tilde{s}) &= \nu(s, \tilde{s}, 1, 1) \\ &= \left(\varphi(s, 1)^{1/\beta} + \varphi(\tilde{s}, 1)^{1/\beta}\right)^\beta - \varphi(s+1, 2). \end{aligned}$$

The four lines $s = p_1/p_2$, $s = q_1/q_2$, $\tilde{s} = p_1/p_2$, and $\tilde{s} = q_1/q_2$ divide D' into 6 regions. On each of those regions $\hat{\nu}$ is a linear transformation of ψ, and thus concave (note that even $\varphi(s+1, 2)$ is linear on every region since $p_1/p_2 + 1 = 2\, q_1/q_2$). Evaluating $\hat{\nu}$ on the 10 corners of the regions shows that $\hat{\nu}(p_1/p_2, p_1/p_2) = 0$, $\hat{\nu}(0.5, 0.5) = 0$, and $\hat{\nu} > 0$ on the other corners. This completes the proof of the second claim. □