A probabilistic nonequivalence test for syntactic $(1, +k)$-branching programs

Petr Savický*

Abstract

We present a satisfiability test and a probabilistic nonequivalence test for syntactic $(1, +k)$-branching programs. The satisfiability test works in time at most $O(\left(\frac{4en}{k}\right)^k sd)$, where s and d are the size and depth of the input branching program. The probabilistic nonequivalence test works in time $O(\left(\frac{12en}{k}\right)^k sd\log^2 n)$. The result has consequences also for parity syntactic $(1, +k)$-branching programs.

1 Introduction

A nondeterministic branching program (or shortly bp) for representing a Boolean function $f(x_1, x_2, \ldots, x_n)$ is an acyclic directed graph with one source and two sinks labeled by 0 and 1. We distinguish two kinds of nonsink nodes. A nondeterministic node has an arbitrary number of outgoing edges without label. A testing node has two outgoing edges labeled by x_i and $\bar{x}_i$ for some $i = 1, 2, \ldots, n$. For an assignment $a_1, a_2, \ldots, a_n$ of the variables, we have $f(a_1, a_2, \ldots, a_n) = 1$ if and only if there is a path from the source to the 1-sink such that all the literals on the edges from the path are satisfied by the assignment.

If we do not explicitly call a bp nondeterministic, we assume that it is deterministic, i.e. we assume that it does not contain nondeterministic nodes.

By an occurrence of a variable, we mean an occurrence of any literal containing the variable. If every path in the bp from the source to a sink contains at most one occurrence of each variable, then the bp is called a read-once bp or 1-bp for short. The syntactic $(1, +k)$-bp are bp's satisfying the following restriction. For every path from the source to a sink there is a set of at most k variables such that only these at most k variables have more than one occurrence in the path.

The complexity of a function in a given type of bp's is the minimum number of nodes in a bp of the given type representing the function. It is known that even $(1, +1)$-bp's are strictly more powerfull than 1-bp's. Namely, there are functions with exponential complexity for 1-bp, but having polynomial syntactic $(1, +1)$-bp's. An example of such a function may be found in [3]. Another example, with complexity at least $2^{n-o(n)}$ for 1-bp's may be found in [6]. Moreover, for every $k \leq \frac{1}{2}n^{1/2}/\log n$ there are functions with a polynomial syntactic $(1, +k)$-bp, but having only exponential size syntactic $(1, +(k-1))$-bp's, see [7], [9]. Analogous result for nonsyntactic $(1, +k)$-bp's may be found in [6].

Let us point out that there are a few more results on lower bounds for nonsyntactic $(1, +k)$-bp's, see [4], [5], [11].

*Institute of Computer Science, Academy of Sciences of Czech Republic, Pod vodárenskou věží 2, 182 07 Praha 8, Czech Republic, e-mail: savicky@uivt.cas.cz . The research was supported by GA of the Czech Republic, grant No. 201/98/0717.

A 1-bp or even nondeterministic 1-bp is satisfiable, if and only if there is at least one path from the source to the 1-sink in it. There are simple efficient algorithms testing this. Moreover, by a result of Blum, Chandra and Wegman, see [1], it is possible to test nonequivalence of two (deterministic) 1-bp by a polynomial time probabilistic algorithm with one-sided error, namely, if the bp's are equivalent, then the algorithm gives always the correct answer.

In the present paper we demonstrate a satisfiability test and a probabilistic nonequivalence test for $(1,+k)$-bp's. If the size of the input bp is s and its depth is d, then the satisfiability test works in time $O(\left(\frac{4en}{k}\right)^k sd)$. If both input bp's have size at most s and depth at most d, then the probabilistic nonequivalence test works in time $O(\left(\frac{12en}{k}\right)^k sd\log^2 n)$ and has the same error probability as the algorithm from [1]. Moreover, we present a consistency test for $(1,+k)$-bp, i.e. an algorithm that tests if an input bp is or is not a syntactic $(1,+k)$-bp in time $O(\left(\frac{3en}{k+1}\right)^{k+1} s)$.

If the input bp is nondeterministic, then the satisfiability test may be done in time $O(\left(\frac{4en}{k}\right)^k s^3)$. Moreover, if the input bp is deterministic, then it is possible to compute the number of satisfying assignments of the function represented by the input bp in time $O(\left(\frac{12en}{k}\right)^k sd)$.

It appears that in the above results it is indeed necessary to work with syntactic $(1,+k)$-bp's. In the nonsyntactic case, where the restriction to the number of occurrences of variables is applied only to consistent computation paths, even for $(1,+1)$-bp the satisfiability and nonconsistency tests are NP-complete, see [8].

The results have consequences also for parity syntactic $(1,+k)$-bp's, i.e. for nondeterministic $(1,+k)$-bp's that accept an input if and only if there is an odd number of paths from the source to the 1-sink consistent with the input. In this case, testing nonequivalence of two bp's may be reduced to testing satisfiability of one bp, since we can easily combine two bp's by parity. We present a probabilistic algorithm for satisfiability, which works in time $O(\left(\frac{12en}{k}\right)^k s^2)$, where we omit logarithmic factors.

Let us point out that in the case of parity nondeterminism, computing the number of satisfying assignments is #P complete even for parity OBDDs and hence also for 1-bp. This follows easily from #P-completeness of the operation SAT-COUNT for Ordered Functional Decision Diagrams (OFDDs), see [10].

The presented algorithms are more efficient than the exhaustive search, whenever $k = o(n)$.

2 The satisfiability test

In this section, we consider a promise version of the satisfiability test, i.e. an algorithm that yields a correct answer if it is guaranteed that the input bp is indeed a syntactic $(1,+k)$-bp. Using some parts of the satisfiability test, it is also possible to obtain a guarantee that the result is indeed correct without any assumptions. However, this leads to a class of bp that is slightly larger than syntactic $(1,+k)$-bp. Hence, in the next section, we demonstrate an algorithm testing if an input bp is or is not a syntactic $(1,+k)$-bp.

Assume, we are given a nondeterministic syntactic $(1,+k)$-bp Q for a function of n variables. Let E denotes its set of edges. If not stated otherwise, a path in Q means a path from the source to the 1-sink. Let I be a set of indices of variables. For every path in Q, let its I-restriction be the set of literals appearing in the path that contain variables with indices from I. For every set I, there are $4^{|I|}$ possible I-restrictions, since for every variable x_i, $i \in I$, there are four possible contributions to the I-restriction, namely $\emptyset, \{x_i\}, \{\bar{x}_i\}, \{x_i, \bar{x}_i\}$.

Let us call the I-restriction containing both x_i and $\bar{x}_i$ for all $i \in I$ the full I-restriction. Let N_I be the number of path in Q such that their I-restriction is full. In particular, $N_\emptyset$ is the set of all paths in Q.

Consider a path in Q. Let J be the set of indices of all variables that occur in the path both positively and negatively. Then the path contributes to N_I if and only if $I \subseteq J$. Using this and the inclusion-exclusion principle, we obtain the following.

Lemma 2.1 *The number of all consistent paths in Q is*

$$\sum_{|I| \leq k} (-1)^{|I|} N_I.$$

Note that Q is satisfiable if and only if there is at least one consistent path in Q. Hence, in order to test satisfiability of Q, it is sufficient to calculate the numbers N_I for $|I| \leq k$.

In order to evaluate N_I, we consider for every node v and every possible I-restriction α the number $N_I^\alpha(v)$, which is the number of paths from v to the 1-sink that have the I-restriction α. The number $N_I^\alpha(v)$ with v equal to the source and α equal to the full I-restriction is equal to N_I. The numbers $N_I^\alpha(v)$ satisfy the following recurrence relations that allow to evaluate them by induction in a bottom up order.
If v is the 0-sink, then $N_I^\alpha(v) = 0$ for every I-restriction α.
If v is the 1-sink, then $N_I^\emptyset(v) = 1$ and $N_I^\alpha(v) = 0$ for every $\alpha \neq \emptyset$.
If v tests a variable x_i, $i \in I$ and v_0 resp. v_1 is the 0-successor resp. 1-successor of v, then

$$N_I^\alpha(v) = \sum_{\substack{\beta \\ \beta \cup \{\bar{x}_i\} = \alpha}} N_I^\beta(v_0) + \sum_{\substack{\beta \\ \beta \cup \{x_i\} = \alpha}} N_I^\beta(v_1).$$

If v tests a variable x_i and $i \notin I$ or v is a nondeterministic node, then

$$N_I^\alpha(v) = \sum_{\substack{u \\ (v,u) \in E}} N_I^\alpha(u).$$

Let us prove the first identity. Consider the set of paths from v to the 1-sink having the I-restriction α. The number of these paths is the left hand side of the identity. We describe, how to partition this set of paths in order to get subsets corresponding to the summands in the right hand side.

For each of the two successors of v consider the subset of paths going through the selected successor. In each of these two sets classify the path according to the I-restriction of the subpath starting in v_0 or v_1. Note that, if such a subpath has I-restriction β and if the skipped edge from v has label x_i resp. $\bar{x}_i$, then $\beta \cup \{x_i\} = \alpha$ resp. $\beta \cup \{\bar{x}_i\} = \alpha$. Hence, the summands in the right hand side are exactly the sizes of the blocks of the above described partition.

In order to prove the second identity, we partition the set of paths only according to the successor of v which the path goes through. In this case, the I-restrictions of the subpaths are the same as the I-restrictions of the whole paths.

Theorem 2.2 *Let a syntactic $(1,+k)$-bp Q be given. The number of consistent path in Q is computable in time $O(\left(\frac{4en}{k}\right)^k sd)$, if the diagram is deterministic and in time $O(\left(\frac{4en}{k}\right)^k s^3)$ if it is nondeterministic.*

Proof: The numbers $N_I^\alpha(v)$ have to be evaluated for all s nodes in the diagram and all I-restrictions for all subsets I of indices of variables of size at most k. The number of these subsets is at most $\sum_{i=0}^{k}\binom{n}{i} \leq \left(\frac{en}{k}\right)^k$. For each of them, we have at most 4^k I-restrictions. Hence, the algorithm performs at most $O(\left(\frac{4en}{k}\right)^k s)$ additions in the deterministic case and at most $O(\left(\frac{4en}{k}\right)^k s^2)$ in the nondeterministic case.

Each of the evaluated numbers is bounded from above by the total number of paths in Q. If Q is deterministic, then the total number of paths is at most 2^d. Hence, in each step, we perform addition of numbers with binary representation of length at most d. This implies the required bound. If the diagram is nondeterministic, the total number of paths is at most 2^s and the length of the numbers is at most s. This implies the bound in the nondeterministic case. $\square$

Clearly, in order to get a correct answer of the algorithm, it is sufficient if $N_I = 0$ for all sets I of size $k+1$, since then $N_I = 0$ also for all larger sets I. In order to test this, it is sufficient to run the above algorithm up to sets of size $k+1$ instead of k. Clearly, the corresponding time bound may be also obtained by replacing k by $k+1$.

By a simple analysis of this guaranteed version of the algorithm we obtain that it works correctly exactly for those bp's such that every inconsistent path from the source to the 1-sink contains at most k variables with both positive and negative occurrence. This class of bp's is slightly larger than syntactic $(1,+k)$-bp's.

3 Testing the $(1,+k)$-property

In order to test, if an input (nondeterministic) bp is a $(1,+k)$-bp, we will need a different classification of paths than in the previous section.

Let I be a set of indices of variables. Then, we assign to every path its I-count. We use Greek letters with a prime to denote I-counts. A I-count is a mapping $\alpha' : I \to \{0,1,2\}$ such that for all $i \in I$ $\alpha'(i) = \min(2, s_i)$ where s_i is the total number of both positive and negative occurrences of x_i in the path.

Let I be a set of indices of variables and let α' be an I-count. Then, by $T_I^{\alpha'}(v)$ we denote the truth value of the statement that there is at least one path from v to a sink, which has the I-count α'. For $T_I^{\alpha'}(v)$, we design similar recurrence relations as for $N_I^\alpha(v)$ from the previous section. The relations are as follows.

If v is a sink, then $T_I^\epsilon(v) = \text{true}$, where $\epsilon(i) = 0$ for all $i \in I$, and $T_I^{\alpha'}(v) = \text{false}$ for all nonzero I-counts α'.

Let α' be an I-count of a path. Consider a prolongation of the path by one edge labeled by x_i resp. $\bar{x}_i$. If $i \notin I$, then the prolonged path has the same I-count as the original path. If $i \in I$, then the I-count of the prolonged path is uniquely determined by α' and the label of the new edge. For simplicity of notation, we denote the I-count of the prolongation as $\alpha' + i$.

Let v_0 resp. v_1 be the 0-successor of v resp. 1-successor of v. Let the variable tested in v be x_i. Then, we have the following relations for $T_I^{\alpha'}(v)$.

If $i \in I$, then

$$T_I^{\alpha'}(v) = \bigvee_{\substack{\beta' \\ \beta' + i = \alpha'}} \left(T_I^{\beta'}(v_0) \vee T_I^{\beta'}(v_1)\right).$$

If $i \notin I$ or if v is a nondeterministic node, then

$$T_I^{\alpha'}(v) = \bigvee_{\substack{u \\ (v,u)\in E}} T_I^{\alpha'}(u).$$

The proof of these relations can be done by appropriate splitting of the set of paths corresponding to the left hand side of each of the identities, similar to that used in the previous section.

Theorem 3.1 *The test if an input branching program is or is not a $(1,+k)$-bp can be done in time $O(\left(\frac{3en}{k+1}\right)^{k+1} s)$.*

Proof: In order to test the required property, it is sufficient to verify that for every I, $|I| = k+1$ and every α' such that $\alpha'(i) = 2$ for all $i \in I$, $T_I^{\alpha'}(v_s)$ is false, where v_s is the source. In order to do this, we have to evaluate $T_I^{\beta'}(v_s)$ for all possible subsets I of size $k+1$ and for all possible I-counts β'. Clearly, this requires at most the number of steps claimed in the theorem. $\Box$

4 The probabilistic nonequivalence test

Every Boolean function is expressible by a polynomial over the real numbers. Using identities $x_i^2 = x_i$, the polynomial can be made multilinear. Moreover, if we have n variables then we have exactly 2^n multilinear monomials and also the dimension of the linear span over the real numbers of all Boolean functions is 2^n. It follows that every function in the linear span of the Boolean functions is expressible by a unique multilinear polynomial. The same holds also for any other field instead of the real numbers.

The probabilistic nonequivalence test for read-once decision diagrams from [1] is based on the following lemma.

Lemma 4.1 *Let $g(x_1, x_2, \ldots, x_n)$ be a nonzero multilinear polynomial over a field F and let $M \subseteq F$ be a finite set. Let $\tilde{a}_i$ be independent random variables with the uniform distribution on M. Then, the probability that $g(\tilde{a}_1, \tilde{a}_2, \ldots, \tilde{a}_n) \neq 0$ is at least $\left(1 - \frac{1}{|M|}\right)^n$.*

In view of this lemma, the only thing which is needed to perform the probabilistic nonequivalence test is to have an efficient way to evaluate for general inputs the unique multilinear polynomials corresponding to the two Boolean functions, which we like to compare. Then, we apply the above lemma to the difference of the two polynomials with, say, $|M| = 2n$.

If we have a read-once branching program representing a Boolean function, then, as shown in [1], we can efficiently evaluate the unique multilinear polynomial corresponding to the function for general inputs. The main step of our generalized equivalence test is a procedure that evaluates the unique multilinear polynomial for a function represented by a $(1,+k)$-bp. Using the procedure described in the next section to evaluate the polynomials for random assignments from the set $M = \{\frac{1}{2n}, \frac{2}{2n}, \ldots, 1\}$, we obtain

Theorem 4.2 *There is a probabilistic nonequivalence test with one-sided error for* $(1,+k)$*-bp, which works in time* $O(\left(\frac{12en}{k}\right)^k sd\log^2 n)$*. If the input diagrams are inequivalent, then the algorithm makes an error with probability at most* $\frac{1}{2}$*.*

If $g(x_1, x_2, \ldots, x_n)$ is a multilinear polynomial corresponding to a Boolean function f, then $g(\frac{1}{2}, \frac{1}{2}, \ldots, \frac{1}{2})$ is the probability that a random assignment yields the value 1 in f. Hence, the procedure from the next section may also be used to compute the number of satisfying assignmets for a $(1,+k)$-bp.

Theorem 4.3 *The number of satisfying assignments of a syntactic* $(1,+k)$*-bp of size* s *and depth* d *can be computed deterministically in time* $O(\left(\frac{12en}{k}\right)^k sd)$*.*

5 Evaluating the multilinear polynomial

Assume, we are given a deterministic syntactic $(1,+k)$-decision diagram Q computing a Boolean function f. By a monomial corresponding to a path, we mean the product of all literals contained in the path. Repetitions are denoted using exponents, i.e. the monomial $x_i^2\bar{x}_i$ means that the path contains x_i two times and $\bar{x}_i$ once. The symbol $\bar{x}_i$ is considered as a formal symbol denoting $1 - x_i$.

For every node v in the diagram, consider the polynomial consisting of monomials corresponding to all paths from v to the 1-sink and call it $g(v)$. It is easy to evaluate this polynomial for a general input by a simple induction in e.g. bottom up order, since $g(v) = x_i g(v_1) + \bar{x}_i g(v_0)$, if x_i is the variable tested in v and v_1 resp. v_0 is the 1-successor resp. 0-successor of v. If v_s is the source, then $g(v_s)$ coincides with f on the Boolean inputs. However, since the diagram may contain paths with several occurrences of some variable, the polynomial obtained in this way neednot be multilinear.

In order to evaluate the unique multilinear polynomial corresponding to the function, we associate several polynomials to Q in such a way that
(i) the polynomials may be efficiently evaluated for general inputs;
(ii) an appropriate combination of these polynomials is equal to the unique multilinear polynomial corresponding to the function computed by Q.

The polynomials will be sums of monomials corresponding to specific subsets of paths. These subsets are defined in terms of the number of occurrences of variables. In order to describe the subsets, we introduce the following notion.

Let I be a set of indices of variables. Then, we assign to every path its I-type. It is a monomial over the variables with index from I defined as follows. It is a product of contributions of the variables with index in I. The contribution of x_i is one of the monomials $\{1, x_i, x_i^2, \bar{x}_i, \bar{x}_i^2, x_i\bar{x}_i\}$ chosen as follows. If the path contains both x_i and $\bar{x}_i$, then the contribution of x_i to the I-type is $x_i\bar{x}_i$. Otherwise, the contribution is the largest of the listed monomials which is contained in the path.

By an I-type, we mean any monomial that can be an I-type of a path. For every set I of indices of the variables, there are $6^{|I|}$ possible I-types. We use Greek letters with double prime to denote I-types.

Let K be a set of indices of variables. A K-type γ'' will be called full, if the contribution to γ'' of every variable with index from K belongs to $\{x_i^2, \bar{x}_i^2, x_i\bar{x}_i\}$. For a full K-type γ'', let $\mathcal{W}_K^{\gamma''}$ denote the set of paths in Q such that their K-type is γ'' and all variables with index not in K have at most one occurrence in the path. In particular, if ϵ denotes the empty type (formally equal to 1), then $\mathcal{W}_\emptyset^\epsilon$ denotes the set of path in Q that are read-once. Note that the collection of sets $\mathcal{W}_K^{\gamma''}$ forms a partition of the set of all paths in Q.

If K is a set of indices of variables, then a K-eliminated monomial corresponding to a path is the product of all literals in the path except those, which contain variables with index from K.

Let $W_K^{\gamma''}$ denote the sum of K-eliminated monomials corresponding to the paths in $\mathcal{W}_K^{\gamma''}$. Moreover, let Z denote the multilinear polynomial corresponding to the function computed by Q. A K-type γ'' will be called consistent, if it does not contain $x_i\bar{x}_i$ for any variable.

Lemma 5.1 *We have*

$$Z = \sum_{|K|\le k} \sum_{\substack{\gamma'' \text{ is full, cons.} \\ K\text{-type}}} W_K^{\gamma''} \left(\prod_{\substack{i\in K \\ x_i^2\in\gamma''}} x_i \right) \left(\prod_{\substack{i\in K \\ \bar{x}_i^2\in\gamma''}} \bar{x}_i \right),$$

where $x_i^2 \in \gamma''$ resp. $\bar{x}_i^2 \in \gamma''$ means that the contribution of x_i to γ'' is x_i^2 resp. $\bar{x}_i^2$.

Proof: Every consistent path belongs to exactly one of the sets $\mathcal{W}_K^{\gamma''}$ for a full consistent K-type γ''. Since we work with a $(1,+k)$-bp, it suffices to consider $|K| \le k$. Moreover, by multiplying a monomial from $W_K^{\gamma''}$ by x_i or $\bar{x}_i$ for all $i \in K$ according to the type γ'', we obtain the multilinear monomial corresponding to the same path. For every Boolean input there is at most one path in Q consistent with the input. Recall that we consider only paths from the source to the 1-sink. Clearly, such a consistent path exists if and only if the value of the function for the given input is 1. This implies that the polynomial determined by the formula in the lemma coincides with the Boolean function computed by Q on all Boolean inputs. Since the formula in the lemma determines a multilinear polynomial, it is equal to Z. $\Box$

The criterion of including a monomial into $W_K^{\gamma''}$ depends on the number of occurrences of the variables in the monomial, including variables with index outside K. In order to compute $W_K^{\gamma''}$, we define some other polynomials, where the criterion of including a monomial will depend only on occurrences of variables from some set of size at most k.

Let $K \subseteq I$ and let γ'' be a full K-type. Then, let $\mathcal{V}_{I,K}^{\text{full},\gamma''}$ be the set of all path in Q such that their I-type is a full I-type and their K-type is γ''. Moreover, let $V_{I,K}^{\text{full},\gamma''}$ denote the sum of K-eliminated monomials corresponding to the paths in $\mathcal{V}_{I,K}^{\text{full},\gamma''}$.

Lemma 5.2 *We have*

$$W_K^{\gamma''} = \sum_{\substack{I\supseteq K \\ |I|\le k}} (-1)^{|I\setminus K|} V_{I,K}^{\text{full},\gamma''}.$$

Proof: Let γ'' be a full K-type. Consider a path in Q having this K-type γ''. Let J be the set of variables having repeated occurrences in the path. Clearly, $J \supseteq K$ and $|J| \le k$. Note that the considered path contributes to $V_{I,K}^{\text{full},\gamma''}$ if and only if $K \subseteq I \subseteq J$. Hence, if $J \ne K$, then the contributions of the path to the right hand side of the identity in the lemma cancel. $\Box$

Note that the criterion for including a monomial into $V_{I,K}^{\mathrm{full},\gamma''}$ depends only on the I-type of the monomial. In order to describe an efficient way of evaluating these polynomials, we express them using one more set of polynomials.

Let α'' be any I-type and let v be a node of Q. Then, let $\mathcal{V}_I^{\alpha''}(v)$ denote the set of those paths in Q from v to the 1-sink that have the I-type equal to α''. Let, moreover, $K \subseteq I$. Then, let $V_{I,K}^{\alpha''}(v)$ be the sum of the K-eliminated monomials corresponding to all paths in $\mathcal{V}_I^{\alpha''}(v)$.

Let $K \subseteq I$ and let γ'' be a full K-type. Clearly, if v_s is the source, then

$$V_{I,K}^{\mathrm{full},\gamma''} = \sum_{\substack{\alpha'' \text{ is full} \\ \alpha'' \supseteq \gamma''}} V_{I,K}^{\alpha''}(v_s), \tag{1}$$

where $\alpha'' \supseteq \gamma''$ means that the contribution of variables with index in K to α'' is the same as their contribution to γ''.

In order to finish the proof, we demonstrate the recurrence relations which allow to evaluate the last set of polynomials. Recall that the diagram is deterministic.

If v is the 0-sink, then $V_{I,K}^{\alpha''}(v) = 0$ for all α''. If v is the 1-sink, then $V_{I,K}^{\epsilon}(v) = 1$, where ϵ is the empty type, and $V_{I,K}^{\alpha''}(v) = 0$ for all nonempty I-types α''.

Let α'' be an I-type of a path. Consider a prolongation of the path by one edge labeled by x_i resp. $\bar{x}_i$. If $i \notin I$, then the prolonged path has the same I-type as the original path. If $i \in I$, then the I-type of the prolonged path is uniquely determined by α'' and the label of the new edge. For simplicity, we denote the type of the prolongation as $\alpha'' x_i$ resp. $\alpha'' \bar{x}_i$, although the exponents in the product may be larger than in the resulting I-type.

Let v_0 resp. v_1 be the 0-successor of v resp. 1-successor of v. Let the variable tested in v be x_i. Then, we have the following relations for $V_{I,K}^{\alpha''}(v)$.
If $i \in K$, then

$$V_{I,K}^{\alpha''}(v) = \sum_{\substack{\beta'' \\ \beta''\bar{x}_i = \alpha''}} V_{I,K}^{\beta''}(v_0) + \sum_{\substack{\beta'' \\ \beta'' x_i = \alpha''}} V_{I,K}^{\beta''}(v_1).$$

If $i \in I \setminus K$, then

$$V_{I,K}^{\alpha''}(v) = \sum_{\substack{\beta'' \\ \beta''\bar{x}_i = \alpha''}} V_{I,K}^{\beta''}(v_0)\bar{x}_i + \sum_{\substack{\beta'' \\ \beta'' x_i = \alpha''}} V_{I,K}^{\beta''}(v_1)x_i.$$

If $i \notin I$, then

$$V_{I,K}^{\alpha''}(v) = V_{I,K}^{\alpha''}(v_0)\bar{x}_i + V_{I,K}^{\alpha''}(v_1)x_i.$$

The proof of these relations can be done again by appropriate splitting of the set of paths corresponding to the left hand side of each of the identities.

If we are given a general (possibly nonboolean) input, then the same relations hold for the values of the polynomials in the given input. Using this, we can efficiently evaluate all the needed polynomials.

Theorem 5.3 *Let $g(x_1, x_2, \ldots, x_n)$ be the multilinear polynomial corresponding to the Boolean function computed by a syntactic $(1,+k)$-bp Q. Assume that $a_1, a_2, \ldots, a_n$ is an assignment of the variables, where a_i for all $i = 1, 2, \ldots, n$ are rational numbers with the* *same denominator and with the bit-length of both the numerator and the denominator at most t. Then, the value $g(a_1, a_2, \ldots, a_n)$ is computable in time $O(\left(\frac{12en}{k}\right)^k sdt^2)$.*

Proof: In the recurrence relations, we have to work with all possible I-types α'', although, the resulting value is needed only for full types α''. Hence, for every set I, we have to consider all $6^{|I|}$ types and there are $2^{|I|}$ possible subsets K. Since we consider only sets I with $|I| \leq k$, we have at most $\left(\frac{12en}{k}\right)^k$ possible tripples I, K, α'' to be considered. For each of these tripples and for all nodes v in the diagram, we have to calculate $V_{I,K}^{\alpha''}(v)$. Note that in the recurrence relations determining this number, we have that in each sum there are at most 3 β'''s satisfying the requirement.

In order to finish the proof, we have to estimate the number of bit operations in the calculation. Since for every path we multiply at most d numbers and there are at most 2^d paths, all the numbers we work with have both the numerator and denominator of size at most $O(td)$. In each step, we multiply a number satisfying this by a_i or $1 - a_i$ for some $i = 1, 2, \ldots, n$. Using the assumption on the numbers a_i, we obtain the required time bound. □

6 Parity nondeterminism

The probabilistic nonequivalence test of [1] was modified to work for parity OBDDs in [2]. In fact, it works even for parity 1-bp's. The test is performed by selecting a random assignment of the variables by arbitrary values from $GF(2^m)$, where $m = \lceil \log 2n \rceil$. Then, we evaluate the two unique multilinear polynomials corresponding to the two diagrams for the selected assignment. Using Lemma 4.1 one can prove that if the diagrams are inequivalent, then the values of the two polynomials differ with probability at least $\frac{1}{2}$.

Assume, we are given a nondeterministic syntactic $(1, +k)$-bp with the parity acceptance mode. For every path in the diagram, consider the monomial over $GF(2)$ defined as follows. If the path is inconsistent, then the monomial is zero. If the path is consistent, then the monomial contains exactly one occurrence of every literal occurring in the path. It is easy to see that the obtained polynomial is multilinear and that it coincides with the computed Boolean function on the Boolean inputs. Hence, it is exactly the unique multilinear polynomial corresponding to the function. The method of classification of path used in the previous section allows us to evaluate this polynomial. This yields the following theorem.

Theorem 6.1 *The satisfiability and nonequivalence test for parity syntactic $(1, +k)$-bp of size s may be done probabilistically with one-sided error in time $O(\left(\frac{12en}{k}\right)^k s^2)$, where we omit logarithmic factors.*

Proof: Since two parity $(1, +k)$-bp are equivalent if and only if their parity is unsatisfiable, it is sufficient to describe a satisfiability test. Using Lemma 4.1, this can done by selecting the values of $x_1, x_2, \ldots, x_n$ at random from $GF(2^m)$, where $m = \lceil \log 2n \rceil$ and by evaluating the unique multilinear polynomial corresponding to the $(1, +k)$-bp. For evaluating the polynomial we use the algorithm from Section 5 with a minor modification described in the following paragraphs.

Assume, we are given a nondeterministic syntactic $(1, +k)$-bp Q. Let E denotes its set of edges. In order to analyze Q, we use the same definition of I-type, $W_K^{\gamma''}$, $\mathcal{V}_{I,K}^{\mathrm{full},\gamma''}$, $V_{I,K}^{\alpha''}(v)$ as in

Section 5. The only difference is that, now, the polynomials are over a field of characteristic 2. It is easy to see that Lemmas 5.1, 5.2 and equation (1) remain true. Moreover, the recurrence relations for $V^{\alpha''}_{I,K}(v)$ remain true for all nodes v testing a variable. We only have to include also a relation for the case that v is a nondeterministic node. The required relation is

$$V^{\alpha''}_{I,K}(v) = \sum_{\substack{u \\ (v,u)\in E}} V^{\alpha''}_{I,K}(u).$$

In order to compute the numbers $V^{\alpha''}_{I,K}(v)$, we have to perform at most $O(\left(\frac{12en}{k}\right)^k s^2)$ additions and at most $O(\left(\frac{12en}{k}\right)^k s)$ multiplications of numbers in $GF(2^m)$, where $m = \lceil \log 2n \rceil$. $\square$

Acknowledgement. The author would like to thank Martin Sauerhoff and Detlef Sieling for pointing out a question stimulating the inclusion of Section 3.

References

[1] M. Blum, A. Chandra, M. Wegman, Equivalence of free Boolean graphs can be decided probabilistically in polynomial time, *Information Processing Letters*, Vol. 10, No. 2 (1980), pp. 80–82.

[2] J. Gergov, Ch. Meinel, Mod-2-OBDDs - a data structure that generalizes EXOR-sum-of-products and ordered binary decision diagrams. *Formal Methods in System Design*, 8 (1996), pp. 273–282.

[3] S. Jukna, Entropy of contact circuits and lower bounds on their complexity, TCS 57 (1988), pp. 113-129.

[4] S. Jukna, A. A. Razborov, Neither reading few bits twice nor reading illegaly helps much, *Disc. Appl. Math.* 85:3 (1998), pp. 223–238.

[5] P. Savický, S. Žák, A lower bound on branching programs reading some bits twice, *TCS* 172 (1997), pp. 293–301.

[6] P. Savický, S. Žák, A read-once lower bound and a (1,+k)-hierarchy, to appear in *TCS*.

[7] D. Sieling, New lower lower bounds and hirerarchy results for restricted branching programs, JCSS 53:1 (1996), pp. 79–87.

[8] D. Sieling, A Separation of Syntactic and Nonsyntactic (1,+k)-Branching Programs, preprint.

[9] D. Sieling, I. Wegener, New lower bounds and hierarchy results for restricted branching programs, WG'94, LNCS 903, pp. 359–370.

[10] R. Werchner, T. Harich, R. Drechsler, B. Becker, Satisfiability problems for OFDDs, In *Representations of Discrete Functions*, T. Sasao, M. Fujita (eds.) Kluwer Academic Publishers, pp. 233–248.

[11] S. Žák, A superpolynomial lower bound for (1,+k(n))-branching programs, MFCS'95, LNCS 969, pp. 319–325.