Information Processing Letters 114 (2014) 124–129

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

A simpler counterexample to a long-standing conjecture on the complexity of Bryant's apply algorithm

Beate Bollig [1]

TU Dortmund, LS2 Informatik, Germany

ARTICLE INFO

Article history:
Received 18 April 2013
Received in revised form 4 October 2013
Accepted 4 November 2013
Available online 7 November 2013
Communicated by M. Yamashita

Keywords:
Analysis of algorithms
Computational complexity
Ordered binary decision diagrams

ABSTRACT

In 1986 in his seminal paper Bryant has introduced Ordered Binary Decision Diagrams (OBDDs) as a dynamic data structure for the representation and manipulation of Boolean functions which allow efficient algorithms for important operations like synthesis, the combination of two Boolean functions by a Boolean operator, and equivalence checking. Based on his empirical evaluation he has conjectured that his apply algorithm for the synthesis works in linear time with respect to the input and output size. Recently in 2012, Yoshinaka et al. have presented a counterexample which contradicts this conjecture but their example has the drawback that the chosen variable ordering for the OBDD representation of the input and output is bad. Therefore, they have raised the question whether Bryant's conjecture may still stand for reasonable variable orderings. Here, a negative answer is given by presenting a simple counterexample which works for such kind of variable orderings.

1. Introduction

In many areas in computer science, problems can be formulated in terms of Boolean functions. Ordered Binary Decision Diagrams, or OBDDs for short, introduced by Bryant in [4], are well suited for the representation and manipulation of Boolean functions. Therefore, they are a common dynamic data structure. Bryant has presented the most important algorithms for the manipulation of OBDDs, also including minimization. In order to understand the behavior of the algorithms their resource requirements have to be analyzed. The synthesis, the combination of two Boolean functions by a Boolean operator, is a key operation, e.g., the transformation of a logical description of a function like a circuit into an OBDD representation is done by a sequence of binary synthesis steps. In this paper we take a closer look at Bryant's apply algorithm for the synthesis operation. (For all formal definitions see Section 2.)

Because of the similarity between Deterministic Finite Automata, or DFAs for short, and OBDDs with respect to the identity permutation, a lot of knowledge on DFAs can be transferred to OBDDs. Let f and g be Boolean functions and let G_f and G_g be OBDDs with respect to the variable ordering π, or π-OBDDs for short, for f and g, respectively. Given a binary Boolean operation $\otimes$, like the $\wedge$- or $\vee$-operation (conjunction or disjunction), Bryant's apply algorithm computes a π-OBDD G_h for the function h defined as $h := f \otimes g$ in time and space $\mathcal{O}(|G_f| \cdot |G_g|)$, where $|G|$ denotes the size of the OBDD G. The size of the π-OBDD G_h for h is bounded above by $\mathcal{O}(|G_f| \cdot |G_g|)$. The idea of the procedure is to apply the synthesis algorithm for DFAs, to avoid the consideration of nodes not reachable from the source, and to take into account that some variables may be left out. Already Bryant has presented an example which shows that the result h may indeed need π-OBDD size $\Theta(|G_f| \cdot |G_g|)$. Therefore, from a worst case point of view this solution is optimal. Bryant's example has the drawback that the chosen variable ordering is bad for the functions f, g, and h and therefore, such a synthesis step is very unlikely to occur in applications. Wegener has improved Bryant's result by the construction of functions f and g for which the π-OBDD size for h is even $\Theta(|G_f| \cdot |G_g|)$ if π is a variable ordering which leads

[1] The author is supported by DFG project BO 2755/1.

0020-0190/$ – see front matter
http://dx.doi.org/10.1016/j.ipl.2013.11.003

to OBDDs of minimal size for f and g (see Theorem 3.3.7 in [7]). Nowadays there are several BDD packages available and almost all of them are influenced by the early implementation of Brace, Rudell, and Bryant [3]. Most of the OBDD packages start to look for a better variable ordering if a binary step leads to a π-OBDD much larger than the given π-OBDDs (see, e.g., [6]). As a result, the main step of an OBDD package is a binary synthesis step followed by a so-called reordering. The worst case behavior of the synthesis operation has been investigated more closely in [2] and a function h has been presented such that the size of any π'-OBDD for h is $\Omega(|G_f| \cdot |G_g|)$ for functions f and g essentially depending on all considered variables. Here, π' is an arbitrary variable ordering. In other words, the function h may need an OBDD representation of size $\Omega(|G_f| \cdot |G_g|)$ even if we choose an optimal variable ordering for h. Nevertheless, in many cases, in particular in real life applications, even the π-OBDD G_h will not be much larger than G_f and G_g. Hence, Bryant has raised the question whether his algorithm is output sensitive, i.e., efficient with respect to $|G_f|$, $|G_g|$, and $|G_h|$. Very recently, Yoshinaka et al. [8] have presented a counterexample which proves that the worst case complexity of the apply algorithm is $\Omega(|G_f| \cdot |G_g|)$ even if the π-OBDD size of h is linear in the π-OBDD size of f and g. Again this example has the drawback that the chosen variable ordering π is far from optimal and they have concluded that Bryant's conjecture might still stand if a reasonable variable ordering is used. Here, a negative answer is given by presenting a counterexample which works for variable orderings which are very good, i.e., variable orderings which lead to OBDDs whose size is asymptotically equal to the size of optimal OBDDs for the considered functions. Note that we do not use new methods but the main contribution is the construction of the very simple functions f and g.

The rest of the paper is organized as follows. In Section 2 we define some notation and basics concerning OBDDs and Bryant's apply algorithm. Section 3 contains our counterexamples for the $\wedge$- and $\vee$-operations which prove that the apply algorithm does not work in linear time with respect to input and output size even if the functions f and g are represented by OBDDs of asymptotically optimal size. We finish the paper with two open questions.

2. Preliminaries

In this section we briefly recall basics concerning OBDDs and the apply procedure for the combination of two Boolean functions by a Boolean operator already presented in Bryant's seminal paper [4].

OBDDs. OBDDs are a very popular dynamic data structure in areas working with Boolean functions, like circuit verification or model checking. (For a history of results on binary decision diagrams see, e.g., [7].)

Definition 1. Let $X_n = \{x_1, \ldots, x_n\}$ be a set of Boolean variables. A variable ordering π on X_n is a permutation on $\{1, \ldots, n\}$ leading to the ordered list $x_{\pi(1)}, \ldots, x_{\pi(n)}$ of the variables. A π-OBDD on X_n is a directed acyclic graph $G = (V, E)$ whose sinks are labeled by the Boolean constants 0 and 1 and whose non-sink (or decision) nodes are labeled by Boolean variables from X_n. Each decision node has two outgoing edges, one labeled by 0 and the other by 1. The edges between decision nodes have to respect the variable ordering π, i.e., if an edge leads from an x_i-node to an x_j-node, then $\pi^{-1}(i) < \pi^{-1}(j)$. Let B_n be the class of all Boolean functions from $\{0,1\}^n$ to $\{0,1\}$. Each node v represents a Boolean function $f_v \in B_n$ defined in the following way. A c-sink, $c \in \{0,1\}$, represents the constant function c. If f_{v_0} and f_{v_1} are the functions represented at the 0- or 1-successor of v, resp., and v is labeled by x_i, then $f_v = \overline{x_i} f_{v_0} \vee x_i f_{v_1}$ (Shannon's decomposition rule). The size of a π-OBDD G, denoted by $|G|$, is equal to the number of its nodes. A π-OBDD of minimal size for a given function f and a fixed variable ordering π is unique up to isomorphism. A π-OBDD for a function f is called reduced if it is the minimal π-OBDD for f. The π-OBDD size of a function f, denoted by π-OBDD(f), is the size of the reduced π-OBDD representing f. An OBDD is a π-OBDD for an arbitrary variable ordering π. The OBDD size of f, denoted by OBDD(f), is the minimum of all π-OBDD(f).

The number of nodes in the reduced π-OBDD representing f is described by the following structure theorem [5].

Theorem 2. *The number of $x_{\pi(i)}$-nodes of the reduced π-OBDD for f is the number s_i of different subfunctions*

$$f_{|x_{\pi(1)}=a_1,\ldots,x_{\pi(i-1)}=a_{i-1}}, \quad a_1, \ldots, a_{i-1} \in \{0,1\},$$

that essentially depend on $x_{\pi(i)}$ (a function g depends essentially on a Boolean variable z if $g_{|z=0} \neq g_{|z=1}$).

The most important issue of OBDDs is the possibility to choose the variable ordering. Depending of this choice the size of an OBDD representing a function f, defined on n Boolean variables and essentially dependent on all of them, may vary between linear and exponential size with respect to n. The most significant bit of binary addition or the direct storage access or multiplexer function are examples for such functions. Therefore, it is an important problem in applications to choose a suitable variable ordering.

Definition 3. Let $f = (f_n)$ be a sequence of Boolean functions, where $f_n : \{0,1\}^n \to \{0,1\}$, and $\pi = (\pi_n)$ be a sequence of variable orderings π_n, where π_n is a permutation on $\{1, \ldots, n\}$ and $n \in \mathbb{N}$. A sequence π is called optimal for f if π_n-OBDD(f_n) = OBDD(f_n) for all $n \in \mathbb{N}$. A sequence of variable orderings π is called bad for f if the quotient of π_n-OBDD(f_n) and OBDD(f_n) is exponential with respect to n for all $n \in \mathbb{N}$. It is called very good for f if π_n-OBDD(f_n) is at most a constant factor larger than OBDD(f_n) for all $n \in \mathbb{N}$.

By abuse of notation we often speak about functions and variable orderings instead of sequences of functions and variable orderings and identify a variable ordering on