

Finite Automata and Boolean Functions *

J.-M. Champarnaud
Université de Rouen, LIFAR
76821 Mont-Saint-Aignan Cedex, France
champarnaud@dir.univ-rouen.fr

Abstract

Our aim is to review some representations of Boolean functions of n variables based on models issued from automaton theory. Let us consider the family L_n of languages made of words of length n over a given alphabet of size two. Every language L of L_n can be interpreted as the truth table of a Boolean function f of n ordered variables. The minimal deterministic automaton of L yields a classical representation of the associated function f : the QROBDD (Quasi Reduced Ordered Binary Decision Diagram) of f . The minimal deterministic automata of maximal size of the languages of the family L_n have been characterized regarding their structure and their size. Here we describe a representation based on the construction and the reduction of a nondeterministic automaton, and we study its properties. The practical interest of this work is linked to the use of Boolean functions in the area of cryptology.

Keywords: Finite automata, Boolean functions, Non-determinism

1 Introduction

The Boolean functions of n variables ($f : \{0, 1\}^n \mapsto \{0, 1\}$) are extensively used in numerous applied fields such as fiability or cryptology. Practically, the size of the objects generated by the computation (truth tables, fault trees, ...) may become prohibitive; therefore many attempts have been done to define efficient data structures [8]. Here we are especially interested by representations based on models issued from automaton theory.

Let $A = \{a, b\}$ be an alphabet with two symbols, and let L_n be the family of languages made of words of A^n (i.e. words of length n). Every language L of L_n can be interpreted as the truth table of a Boolean function f of n ordered variables. According to Myhill-Nerode theorem [10], every regular language L is recognized by a minimal deterministic (complete and accessible) automaton \mathcal{A}_L which is unique up to an isomorphism and minimal regarding to the number of states. For a language L of L_n , the automaton \mathcal{A}_L is thus a canonical representation of the Boolean function f associated to L ; among BDDs community, this representation is called the Quasi Reduced Ordered Binary Decision Diagram or QROBDD of f [2]. The subfamily of languages of L_n whose minimal deterministic automaton has a maximal size has been studied in [5]; a characterization of the structure and of the size of the associated automata, called minmax automata, has been provided. The Boolean functions corresponding to minmax automata seem to have interesting properties for cryptology applications [9].

*Partially supported by the Scientific Research Program C2A2

In this paper, we present a nondeterministic approach for the representation of Boolean functions. It is well-known that there exists no canonical representative of the set of nondeterministic automata recognizing a given language, and that the computation of a nondeterministic automaton with a minimal number of states is PSPACE-hard in the general case [6] and NP-hard in the case of L_n languages [1].

However, designing a representation of Boolean functions based on the construction and the reduction of nondeterministic automata seems interesting since it is likely to reduce the size of the space necessary to store the function. Thus, one can expect, in some cases, to be able to compute a nondeterministic representation, when the construction of a deterministic solution would fail by lack of memory space.

The paper is organized as follows: we first recall basic notions of automaton theory, then we describe deterministic constructions (minimal deterministic automaton, QROBDD, OBDD, minmax automata) and finally we present a nondeterministic approach.

2 Preliminaries

We briefly recall the main notions of automaton theory used in this paper. For further details about these topics, we refer to classical books such as [11].

Let Σ be a finite alphabet. A *finite automaton* over Σ is a 5-tuple $\mathcal{M} = (\Sigma, Q, I, F, T)$ with:

- Q is a set of *states*,
- I is a subset of Q whose elements are the *initial states*,
- F is a subset of Q whose elements are the *final states*,
- T is a subset of the cartesian product $Q \times \Sigma \times Q$ whose elements are the *transitions*.

The graph $G = (Q, T, \Sigma)$ is the *state graph* of the automaton \mathcal{M} . The application δ from $Q \times \Sigma$ to 2^Q , such that: $\forall q \in Q, \forall a \in \Sigma, \delta(q, a) = \{q' \in Q \mid (q, a, q') \in T\}$ is the *transition function* of \mathcal{M} .

The automaton \mathcal{M} is *deterministic* if there is only one initial state and if for all $(q, a) \in Q \times \Sigma$ there is at most one state q' such that $(q, a, q') \in T$. We say that \mathcal{M} is a *DFA* if it is deterministic and a *NFA* otherwise. For a DFA, the transition function is an application from $Q \times \Sigma$ to Q , and the state $\delta(q, a)$ is denoted by $q \cdot a$.

A *path* of \mathcal{M} is a sequence $(q_i, a_i, q_{i+1}), i = 1, \dots, n$, of consecutive edges. Its *label* is the word $w = a_1 a_2 \dots a_n$. A word $w = a_1 a_2 \dots a_n$ is *recognized* by the automaton \mathcal{M} if there is a path with label w such that $q_1 \in I$ and $q_{n+1} \in F$. The language *recognized* by the automaton \mathcal{M} is the set of words which it recognizes; it is denoted by $L(\mathcal{M})$. Two automata are *equivalent* if they recognize the same language. A language L is *recognizable* if and only if there exists an automaton \mathcal{A} such that $L(\mathcal{A}) = L$. Kleene's theorem [7] states that a language is recognizable if and only if it is *regular*.

Let q be a state of the deterministic automaton $\mathcal{M} = (\Sigma, Q, i, F, T)$. The (right) language of q is the language recognized by the automaton $\mathcal{M}_q = (\Sigma, Q, q, F, T)$ deduced from \mathcal{M} by taking q as the initial state.

An automaton is *complete* if its transition function is defined for every element of $Q \times \Sigma$. A state q is *accessible* (resp. *coaccessible*) if there exists a path from an initial state to q (resp. from q to a final state). An automaton is *accessible* (resp. *coaccessible*) if all its states are accessible (resp. coaccessible). An automaton is *trim* if all its states are accessible and coaccessible.

Let L be a regular language and u be a word of Σ^* . The (left) quotient of L w.r.t. u is the language $u^{-1}L = \{v \in \Sigma^* \mid uv \in L\}$. The following property holds: a language is

regular if and only if the set of its quotients is finite. It leads to the definition of the automaton $\mathcal{A}_L = (\Sigma, Q, i, F, \delta)$ which is such that:

- Q is the set of the quotients of L ,
- i is the language L itself,
- F is the set of quotients which contain the empty word,
- δ is defined by: $\delta(u^{-1}L, x) = (ux)^{-1}L$.

The automaton \mathcal{A}_L is deterministic, complete, accessible. It recognizes L and it is uniquely defined, up to an isomorphism. Moreover it can be shown that it has a minimal number of states. Therefore it is called the minimal automaton of L . It can be deduced from any deterministic automaton recognizing L by computing its Nerode equivalence [10].

3 Deterministic representation of a Boolean function

Let n be a positive integer and let x_1, \dots, x_n be n Boolean variables. Let B_n be the set of Boolean functions of the n variables x_i .

3.1 Minimal deterministic automaton

Let us consider a two-symbol alphabet A , say $A = \{a, b\}$, and let L_n be the family of languages made of words of A^n (i.e. words of length n). There is a one-to-one mapping from B_n to L_n which associates the language L^f to the function f . As mentioned in [4], L^f can easily be deduced from the truth table of f as follows: if $m = m_1 \wedge m_2 \dots \wedge m_n$ is a minterm such that $f(m) = 1$ then the word $u = u_1 \cdot u_2 \dots \cdot u_n$ is in L^f , with $u_i = a$ if $m_i = \overline{x_i}$ and $u_i = b$ if $m_i = x_i$.

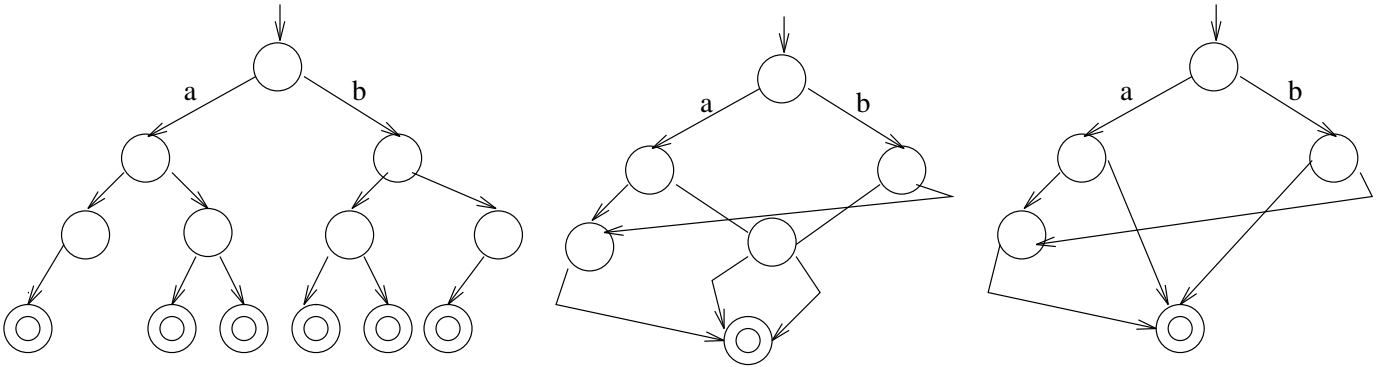


Figure 1: Truth table, minimal automaton and OBDD.

For example, let $g = (\overline{x_1} \wedge ((\overline{x_2} \wedge \overline{x_3}) \vee x_2)) \vee (x_1 \wedge (\overline{x_2} \vee (x_2 \wedge \overline{x_3})))$. The first graph of Figure 1 represents both the truth table of g and a deterministic automaton recognizing the language $L^g = \{aaa, aba, abb, baa, bab, bba\}$.

Thus, according to Myhill-Nerode theorem [10], the minimal deterministic automaton \mathcal{A}_{L^f} of the language L^f associated to a given Boolean function f is a canonical representation of f . This representation is known as the Quasi Reduced Ordered Binary Decision Diagram or QROBDD of f [2]. For example, the second graph of Figure 1 is the minimal automaton \mathcal{A}_{L^g} .

3.2 Ordered Binary Decison Diagram

As a language representation, the state graph of \mathcal{A}_{L^f} cannot be reduced. Conversely, as far as it is interpreted as the representation of a Boolean function, it can be reduced, according to the Boolean properties $x_i \vee \overline{x_i} = 1$ and $x \wedge 1 = x$, where x is any x_i or any $\overline{x_i}$. This leads to the second rule of the procedure REDUCE [2] and to the representation called OBDD (Ordered Binary Decision Diagram) which is canonical too and minimal w.r.t. the number of nodes. Notice that this rule is equivalent to the computation of the ε -closure of the graph deduced from \mathcal{A}_{L^f} by replacing pairs of transitions $((i, a, j), (i, b, j))$ by a unique ε -transition. Of course the resulting automaton does not recognize L^f any more. For example, the third graph of Figure 1 is the OBDD of the function g .

3.3 Minmax automata

According to [5], given a positive integer n , the minimal automaton of a language L of L_n is said to be a minmax automaton if and only if the number of its states is maximal. It is shown that for all i , $0 \leq i \leq n$, the number of states of the level i of the minimal automaton of a language L of L_n is not greater than $s_i = \min(2^i, 2^{2^{n-i}} - 1)$. Furthermore, if x is the real number such that $n = 2^x + x$ and $k = \lceil x \rceil$, we have: $i < k \Rightarrow 2^i < 2^{2^{n-i}} - 1$ and $i \geq k \Rightarrow 2^i > 2^{2^{n-i}} - 1$. Hence, given a positive integer n , it is possible to construct a minimal automaton with exactly s_i states at level i , where $s_i = 2^i$ for all i , $0 \leq i \leq k$ and $s_i = 2^{2^{n-i}} - 1$ for all i , $k < i \leq n$. This construction is illustrated by Figure 2, with $n = 5$.

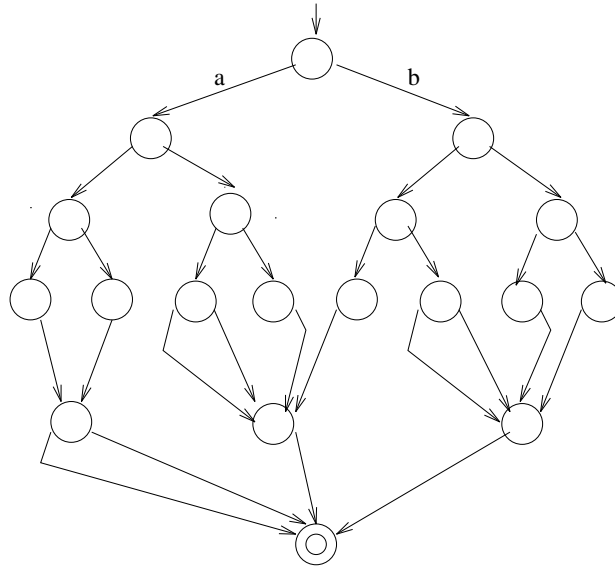


Figure 2: A minmax automaton.

Finally the following property holds for the number $s(n)$ of states in a minmax automaton of L_n :

$$1 = \liminf_{n \rightarrow \infty} ns(n)/2^n \leq \limsup_{n \rightarrow \infty} ns(n)/2^n = 2$$

4 Nondeterministic representation of a Boolean function

Let f be a Boolean function of n variables and L be the associated language in L_n . Let \mathcal{A}_L be the minimal deterministic automaton of L . Our aim is to deduce from \mathcal{A}_L a nondeterministic automaton \mathcal{B} which recognizes L and which has fewer states than \mathcal{A}_L .

We describe a reduction procedure based on occurrences of specific subgraphs in the state graph of \mathcal{A}_L . This procedure involves k successive levels denoted by $i, i + 1, \dots$ and $i + k - 1$. It is likely to reduce the number of states of levels $i + 1$ to $i + k - 2$, as well as the number of transitions going out from these levels, while it keeps constant the number of states of the levels i and $i + k - 1$. We first present the version involving three levels, illustrated by the Figure 3.

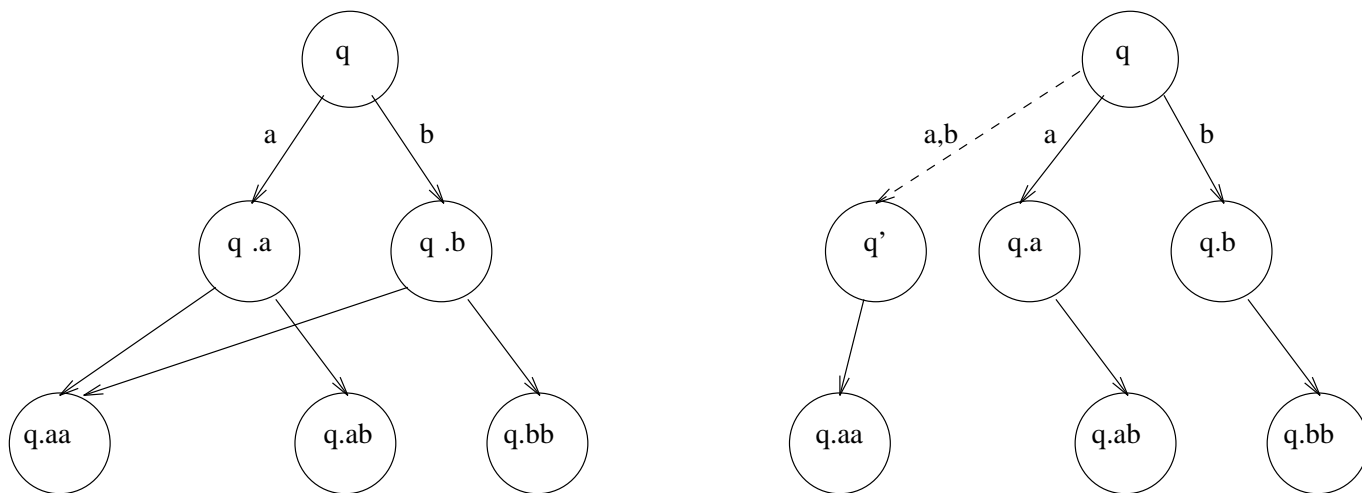


Figure 3: Transformation of type A.

Let q be a state of the automaton \mathcal{A}_L , belonging to the level i and such that $q \cdot a \neq q \cdot b$. An elementary transformation of type A is applied to the subtree rooted in q , if $q \cdot aa = q \cdot ba$. In this case, a new state q' is created at the level $i + 1$, as well as a transition by a from q' to $q.aa$, and a transition by a and one by b from q to q' (these two transitions are represented by a dash line from q to q'). Transitions by a from $q.a$ and $q.b$ to $q.aa$ are removed. The resulting automaton \mathcal{B} is nondeterministic since there exist two transitions by a (and two transitions by b) exiting from the state q . Similarly, an elementary transformation of type B is applied to the subtree rooted in q , if $q \cdot ab = q \cdot bb$.

Proposition 1 *Applying transformations of type A or B to the states of a given level of the automaton \mathcal{A}_L preserves the recognized language.*

Proof. The (right) language of each state at level $i + 2$ is unchanged. Therefore, the language of the state q in the automaton \mathcal{B} is equal to its language in the automaton \mathcal{A}_L . Notice that there is no condition concerning the connection of the subtree rooted in q to the rest of the state graph. ■

The immediate benefit of these transformations is not obvious: an individual transformation generally makes the number of states and the number of transitions be increased by 1, even

though they are decreased by 1 in the case when $q \cdot aa = q \cdot ba$ and $q.ab$ (or $q.bb$) is not defined, or symmetrically when $q \cdot ab = q \cdot bb$ and $q.aa$ (or $q.ba$) is not defined.

In fact, transformations of type A and B are interesting because they are likely to modify the right languages of some states at level $i + 1$. Thus, performing these transformations globally at level i may lead to merge some states at level $i + 1$.

Let us give a generic example of such a reduction of the number of states. Let us suppose that there exist p states q_1, \dots, q_p at level i , for which the transformation A is suitable and such that $q_j \cdot ab = r$ and $q_j \cdot bb = s$, for all j from 1 to p . The Figure 4 illustrates this situation with $p = 2$.

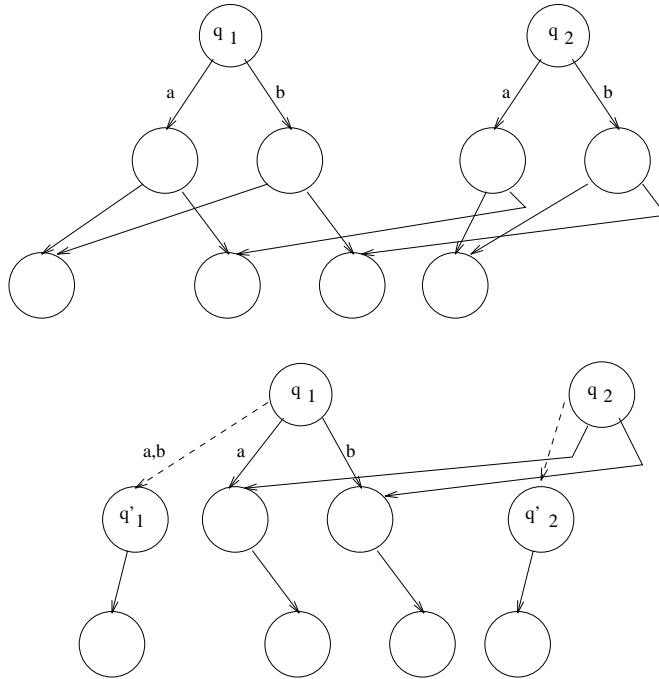


Figure 4: Example of reduction on three levels.

After transformation A is performed, the states $q_j \cdot a$, for all j from 1 to p , appear to be all equivalent. Similarly the states $q_j \cdot b$, for all j from 1 to p , are all equivalent. In this generic example, the number of states is decreased by $p - 2$, as well as the number of transitions.

The transformations A and B can be generalized to the case when k successive levels are involved ($k > 3$).

This will be illustrated by extending the previous example to the case of 4 levels. We suppose that there exist p states q_1, \dots, q_p at level i such that the transformation A is suitable for the states $q_j \cdot a$ and $q_j \cdot b$, for all j from 1 to p . Moreover we assume that: $q_j \cdot aab = r$, $q_j \cdot abb = r$, $q_j \cdot aba = t$ and $q_j \cdot abb = u$, for all j from 1 to p . The Figure 5 illustrates this situation with $p = 2$.

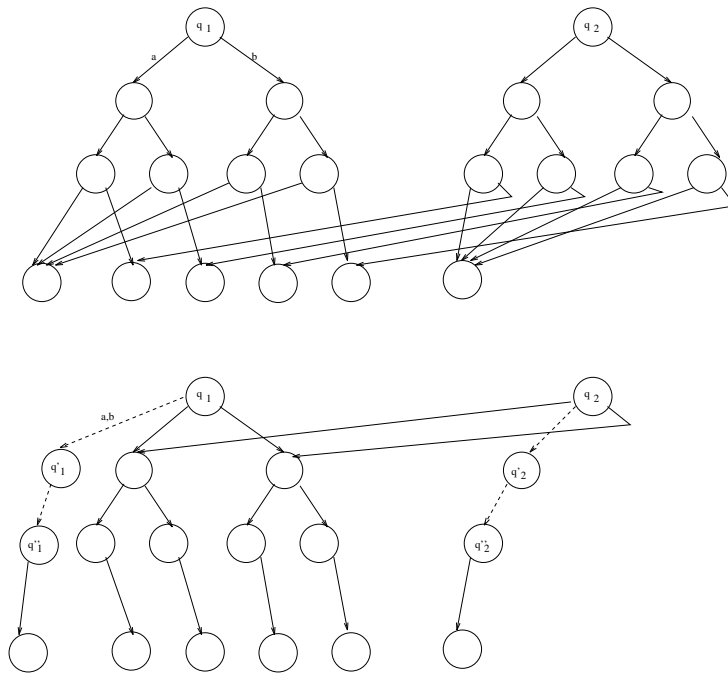


Figure 5: Example of reduction on four levels.

After transformation A is performed, the states $q_j \cdot a$, for all j from 1 to p , appear to be all equivalent. Similarly the states $q_j \cdot b$, for all j from 1 to p , are all equivalent. In this example, the number of states is decreased by $4p - 2$, and the number of transitions by $7p - 8$.

Notice that the second rule of the procedure REDUCE can be applied to the state q'_j , for all j from 1 to p , since q'_j is connected to the same state q''_j by one transition by a and one by b . The result is that q'_j is removed and q_j is directly connected to q''_j by one transition by a and one by b . Hence an additional reduction of p states and $2p$ transitions.

5 Conclusion

The inductive computation of the minimal deterministic automaton of the language denoted by an arbitrary regular expression [3] associates a nondeterministic representative to each subexpression, and the determinization and minimization operations are only performed once, on the result of the evaluation. On the opposite, the step by step computation of the QROBDD of a Boolean expression [2], associates its minimal deterministic representative to each subexpression. The main objective in designing an efficient nondeterministic representation is to reduce the memory space. We expect that in some cases this approach may yield a solution when the deterministic one would fail.

References

- [1] J. Amilhastre, Heuristiques de minimisation des automates non-déterministes, *Proceedings of WIA '99*, Potsdam, 1999.
- [2] R.E. Bryant, Graph Based Algorithms for Boolean Functions Manipulation, *IEEE Transactions on Computers* C-35(8)(1986), 677-691.

- [3] J.-M. Champarnaud and G. Hansel. Automate, a computing package for automata and finite semigroups. *J. Symbolic Comput.*, 12:197–220, 1991.
- [4] J.-M. Champarnaud and J.-F. Michon, Automata and Binary Decision Diagrams, in WIA'98, *Lecture Notes in Computer Science*, **1660**, J.-M. Champarnaud, D. Maurel and D. Ziadi eds., Springer-Verlag, 1999, 178–182.
- [5] J.-M. Champarnaud and J.-E. Pin, A Maxmin Problem on Finite Automata, *Discrete Applied Mathematics* 23(1989), 91–96.
- [6] T. Jiang and B. Ravikumar, Minimal NFA Problems are hard, *SIAM Journal of Computation*, 22(1993), 1117–1141.
- [7] S. Kleene. Representation of events in nerve nets and finite automata. *Automata Studies*, Ann. Math. Studies 34:3–41, 1956. Princeton U. Press.
- [8] J. Lafferty and A. Vardy, Ordered Binary Decision Diagrams and Minimal Trellises, *IEEE Transactions on Computers*, 48-9(1999), 971–986.
- [9] J.-F. Michon, Complexité des fonctions Booléennes, *Manuscript*, 2001.
- [10] A. Nerode, Linear Automata Transformation, *Proc. AMS* 9 (1958), 541–544.
- [11] S. Yu. Regular languages. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume I, Words, Languages, Grammars, pages 41–110. Springer-Verlag, Berlin, 1997.