A Generalized Reduction of Ordered Binary Decision Diagram

Internship at DFKI (Bremen, Germany) from 15/03/2017 to 15/06/2017

Joan Thibault
Supervisor: Rolf Drechsler

June 15, 2017

Abstract

The Reduced Ordered Binary Decision Diagram (ROBDD) [3, 10] is the state-of-the-art representation for Boolean functions. They are used in various fields such as logic synthesis, artificial intelligence or combinatorics. However, efficiently manipulating ROBDDs is memory expensive. Several variants exists which allows to capture small properties of Boolean functions in order to simplify their representation and reduce the computation time, such as "output negation". Some variants are specialized in the representation of some subsets of Boolean functions, for example Zero Suppressed Binary Decision Diagrams (ZBDDs) which are more suitable for sparse functions. Simply merging variants may break canonicity (for example ZBDD and "output negation"). Therefore, we designed GROBDD (Generalized Reduction of Ordered Binary Decision Diagram): a model that aims at providing sufficient conditions in order to ensure the semi-canonicity of the representation. Using GROBDD, we designed three new variants, namely: "extraction of useless variables", "extraction of input negation and output negation invariants" and "extraction of 1-predictions". These variants have been implemented using OCaml [7] and tested against several set of circuits [6, 11] and CNF formulas [5].

1 Introduction

Nowadays, most critical systems rely on digital circuits: in transportation (e.g. cars, train, plains) , communication (e.g. satellites), computation (e.g. data centers, super-computers), exploration (e.g. space rocket, rovers). One way to minimize risks in digital parts of these systems is to provide a formal proof that they respect their specification. On the other hand, we want to minimize costs and energy consumption while maximizing their performances. In order to efficiently optimize digital circuits we usually rely on complex programs. However, these programs are rarely proven themselves, thus, circuits optimized using them might not be equivalent to the initial design and therefore might not respect the specification. The obvious solution would be to prove optimizing programs, however two major issues arise : these programs are complex (thus, proving them would be expensive) and might be proprietary (thus, one cannot check that the proof is correct). A simpler alternative is to design a program which checks that two digital circuits are equivalent. With this alternative, the only piece of software which needs to be proven is the "equivalence checker".

In order to prove that two digital circuits are equivalent, there is two main algorithmic solutions: (1) the DPLL (Davis–Putnam–Logemann–Loveland) algorithm and (2) compilation to ROBDD (Reduced Ordered Binary Decision Diagram). On one hand, the DPLL algorithm is a procedure usually implemented with various heuristics such as *unit propagation*, early conflict detection or *conflict driven clause learning*. Secondly, the compilation of both circuits into Reduced Ordered Binary Decision Diagrams (ROBDDs). On the other hand, ROBDD is a canonical structure which represents a vector of functions. Therefore, by using hash-consing techniques, we can perform the equality test between two compiled functions in constant time. However, the compilation might take an exponential time in the number of variable. In this report we will focus on ROBDDs.

ROBDDs have various other applications such as: Bounded Model Checking, Planning, Software Verification, Automatic Test Pattern Generation, Combinational Equivalence Checking or Combinatorial Interaction Testing.

However, ROBDDs are memory expensive as their size tends to grow exponentially with the number of variables. Several variants have been invented in order to capture some semantic properties of the function and reduce the memory consumption. For example, Zero suppressed Binary Decision Diagram (ZBDD) are better suited for representing sparse functions. In this report we will use the "output negation" variant [2], which extends the reduction rules in order to guarantee canonicity under negation. Therefore, it allows (1) to reduce the size of the structure and (2) to negate a function in constant time. Other extensions of the reduction rules exist such as: "input negation" [9] (each edge can complement the locally first input), "shifting variables" [9] (each edge stores the number of useless variables before the next significant variables) or "dual edge" [8] (we define the dual of a function f by $\bar{f} = (x_1, \ldots, x_n) \longrightarrow \neg f(\neg x_1, \ldots, \neg x_n)$, therefore the reduction works similarly to the "output negation").

An other problem that is usually raised when dealing with ROBDD is their lack of conciseness, i.e. there are functions which have a polynomial representation using an And-Inverter-Graph (AIG, i.e. a logic circuit composed of `AND` and `NOT` gates), but only exponential representation when using ROBDD. For example, it has been proven [3] that the integer multiplication has a quadratic AIG

representation but no polynomial ROBDD representation (at least exponential).

The "output negation" variant, by adding expressiveness to edges, improves the conciseness. We want to go further in this approach by allowing more complex transformations on edges while maintaining canonicity. Therefore, we introduce `GROBDD`(Generalized Reduction of Ordered Binary Decision Diagram): a framework that aims at providing sufficient conditions Such transformations can disturb variables' order of evaluation, allowing to represent "simple" decision processes in a more concise way. Furthermore, some transformations (within the range of transformation allowed) could have their complexity drastically reduced. For example, using the "output negation" variant, the complementation's time and space complexity goes from linear in the number of nodes to constant.

In this report, we introduce three set of transformations: "`U`seless variables extraction" (or "`U` extract" for short), the "input `N`egation and output `N`egation `I`nvariant extraction" (or "`NNI` extract" for short) and "1-prediction extraction" (or "`X` extract" for short).

The "`U` extract" variant ensures that the function represented by any node does not have useless variable. Therefore, it sets an upper bound on the number of nodes of arity n to 2^{2^n} (this upper bound is not reached, because functions with useless variables cannot have a node to represent them). Furthermore, it allows to "copy" functions at (almost) no cost as the expression $f(x_1, x_3, x_5, x_7)$ and $f(x_0, x_1, x_3, x_5)$ are share the same structure.

The "`NNI` extract"variant would allow to complement inputs and output on any edge. The reduction rules would ensure that there is at most 2^{2^n-n} nodes of arity n (this upper bound is not reached). However, simply introducing such transformation breaks the canonicity, therefore, we introduce polarity-phase invariant detection: we compute for each encountered function the linear space of input/output negations which do not change the function. This variant is called "input `N`egation and output `N`egation `I`nvariant extraction" (or "`NNI` extract" for short) and is a generalization of the work of Burch et al. [4].

We define a 1-prediction[1] as follow: Let f be a Boolean function of arity $n+1$, $i \in [\![0, n]\!]$ be an integer, t and r be Booleans. We say that the function f admits a 1-prediction (i, t, r) iff $\forall (x_1, \ldots, x_n), f(x_1, \ldots, x_{i-1}, t, x_i, \ldots, x_n) = r$. The "`X` extract" variant, by allowing to extract 1-predictions, represents a generalization of ZBDD (Zero Suppressed Binary Decision Diagram), thus, allows to efficiently represent sparse functions. This variant is compatible with the "output negation" variant.

Due to the limited length of this report, we will not detail further the "`NNI` extract" and "`X` extract" variants. The Generalized Reduction of Ordered Binary Decision Diagram (`GROBDD`) is a framework that aims at providing examples of such reduction rules and boundaries on future reduction rules that would fit inside this framework.

The remainder of this report will be organized as follows. In Section 2, we formally introduce Boolean functions, useless variables and some notations. Then, we introduce ROBDD in Section 3 and GROBDD in Section 4. In Section 5, we introduce the "`U` extract" variant as part of the `GROBDD` framework, before exposing results against three different benchmarks [5, 6, 11] using our implementation in OCaml.

[1] As far as we know, their is no name in the literature for this property

2 Notations

Reduced Ordered Binary Decision Diagrams represent Boolean functions. In this section we introduce notations necessary to their manipulation.

We denote the set of Booleans $\mathbb{B} = \{0, 1\}$. The set of Boolean vector of size $n \in \mathbb{N}$ is denoted $\mathbb{B}^n$. The set of Boolean functions of arity (i.e. the number of variables) $n \in \mathbb{N}$ is denoted $\mathbb{F}_n = \mathbb{B}^n \longrightarrow \mathbb{B}$.

We denote the conjunction by $\wedge$ (AND), the disjunction by $\vee$ (OR), the negation by $\neg$ (NOT) and the symmetric difference by $\oplus$ (XOR). We denote the Shannon operator by $\longrightarrow_S$, defined by $\forall x, y, z \in \mathbb{B}, x \longrightarrow_S y, z = (\neg x \wedge y) \vee (x \wedge z))$.

Definition 1 *Restriction*
Let $f \in \mathbb{F}_{n+1}$ be a Boolean function of arity $n+1$ and $b \in \mathbb{B}$ be a Boolean. We denote $f_b = (x_1, \ldots, x_n) \longrightarrow f(b, x_1, \ldots, x_n)$. Remark: f_0 is called the negative restriction of f and f_1 is called the positive restriction of f

Definition 2 *Construction*
Let $f, g \in \mathbb{F}_n$ be a pair of Boolean function of arity n. We denote $f \star g$ the Boolean function of arity $n+1$ defined by $f \star g = (x_0, x_1, \ldots, x_n) = x_0 \longrightarrow_S f(x_1, \ldots, x_n), g(x_1, \ldots, x_n)$. N.B.:$(f \star g)_0 = f$ and $(f \star g)_1 = g)$.

Theorem 1 *Expansion Theorem*
Let $f \in \mathbb{F}_{n+1}$ be a Boolean function, then $f = f_0 \star f_1$.

Definition 3 *Useless Variables*
Let $f \in \mathbb{F}_{n+1}$ be a Boolean function and $i \in [\![0, n]\!]$ be an integer. We say that the i-th variable of f is useless, iff $\forall (x_1, \ldots, x_n) \in \mathbb{B}^n, f(x_1, \ldots, x_{i-1}, 0, x_i, \ldots, x_n) = f(x_1, \ldots, x_{i-1}, 1, x_i, \ldots, x_n)$

Definition 4 *1-Prediction*
Let f be a Boolean function of arity $n+1$, $i \in [\![0, n]\!]$ be an integer, t and r be Booleans. We say that the function f admits a 1-prediction (i, t, r) iff $\forall (x_1, \ldots, x_n), f(x_1, \ldots, x_{i-1}, t, x_i, \ldots, x_n) = r$.

Definition 5 *Polarity-Phase Invariant*
Let f be a Boolean function of arity n, and $X = (y, z_1, \ldots, z_n) \in \mathbb{B}^{n+1}$ be a vector of $n+1$ Booleans. We say that X is a polarity-phase invariant of f, iff $\forall (x_1, \ldots, x_n) \in \mathbb{B}^n, f(x_1, \ldots, x_n) = y \oplus f(x_1 \oplus z_1, \ldots, x_n \oplus z_n)$. Similarly to Burch and Long [4], we can prove that the set of polarity-phase invariant of a function is a linear space. Such a linear space can be represented as row-echelon matrix of Booleans.

3 Reduced Ordered Binary Decision Diagram (ROBDD) and Canonicity

Definition of Binary Decision Diagram (BDD)

A Reduced Ordered Binary Decision Diagram is a directed acyclic graph $(V \cup T, \Psi \cup E)$ representing a vector of Boolean functions $F = (f_1, ..., f_k)$ over an infinite set of variables (but with a finite support set). Nodes are partitioned

into two sets : the set of internal nodes V and the set of terminal nodes T. Every internal node $v \in V$ has one field *var*, which represents the index of a variable and two outgoing edges respectively denoted *if0* and *if1*. When using the "output negation" variant, there is only one terminal called 0, which represents the constant function returning 0. Edges are partitioned into two sets : the set of root edges Ψ and the set of internal edges E. There is exactly k root edges, a root edge is denoted Ψ_i (with $i \in [\![1, k]\!]$, informally, Ψ_i is the root of the ROBDD representing f_i. Every edge has an inversion field $neg \in \mathbb{B}$ and a destination node denoted *node*.

We denote $\phi(node)$ the semantics of the node *node* and $\psi(edge)$ the semantic of the edge *edge* as follow:

- $\forall i, f_i = \psi(\Psi_i)$
- $\forall edge \in \Psi \cup E, \psi(edge) = edge.neg \oplus \phi(edge.node)$
- $\phi(0 \in T) = 0$
- $\forall node \in V, \phi(node) = node.var \longrightarrow_S \psi(node.if0), \psi(node.if1)$

Definition 6 *Reduced Ordered Binary Decision Diagram (ROBDD)*
A ROBDD is said `Ordered` *if (1)* $\forall v \in V$, $v.if1.node \in V \Rightarrow v.var > v.if1.node.var$ *and* $v.if0.node \in V \Rightarrow v.var > v.if0.node.var$ *(i.e. the var field of any node is greater than the var field of its children). A ROBDD is said* `Reduced` *if (2)* $\forall v \in V, v.if0 \neq v.if1$ *and (3) every node has an in-degree strictly positive.*

Theorem 2 *ROBDDs are canonical*
Let us consider a ROBDD G representing the vector of Boolean functions $F = (f_1, ..., f_n)$, then for every nodes $v_1, v_2 \in G$, $\phi(v_1) = \phi(v_2) \Leftrightarrow v_1 = v_2$.

A proof of this theorem is available in the review of Somenzi et al. [10].

Effective construction

In practice, ROBDDs are created starting from the ROBDDs for constants and variables and by applying operators (e.g. NOT, AND, XOR, quantification operators, restriction, etc.) and are kept reduced at all times. A ROBDD is a multi-rooted diagram which represent a vector of Boolean function, the equality test between two of this function is a constant time operation. Canonicity is ensured by using a association map (usually a hash table), which maps all nodes to an identifier (usually a pointer to the node itself). Therefore, before to create a new node, all procedure must first check that it does not already exists. If it does, the procedure retrieve its identifier, otherwise, generate a new identifier and add the pair node-identifier into the association table. The association table is usually called *unique table*.

4 Introduction of Generalized Reduction of Ordered Binary Decision Diagram (`GROBDD`)

The Generalized Reduction of Ordered Binary Decision Diagram (`GROBDD`) is a framework that aims at providing examples of such reduction rules and boundaries on future reduction rules that would fit inside this framework. In this

Section, we formally define GROBDDs, then we present a set of properties which are sufficient to ensure the semi-canonicity of GROBDDs.

4.1 Initial definition of GROBDD

A GROBDD is very similar to an actual ROBDD, the main difference is that on every edge there is a transformation descriptor (the "output negation" is an example of such descriptor).

A Reduced Ordered Binary Decision Diagram is a directed acyclic graph $(V \cup T, \Psi \cup E)$ representing a vector of Boolean functions $F = (f_1, ..., f_k)$. Nodes are partitioned into two sets : the set of internal nodes V and the set of terminal nodes T. Every internal node $v \in V$ has two outgoing edges respectively denoted *if0* and *if1*. Every internal node $v \in V$ has a field *index* which represents a unique identifier associated to each node. edges are partitioned into two sets : the set of root edges Ψ and the set of internal edges E. There is exactly k root edges, a root edge is denoted Ψ_i with $0 \leq i < k$, informally, Ψ_i is the root of the GROBDD representing f_i. Every edge has a transformation descriptor field γ and a destination node denoted *node*.

We denote $\rho(\gamma) : \mathbb{F}_n \longrightarrow \mathbb{F}_m$ the semantic interpretation of the transformation descriptor γ. We define $\phi(node)$ the semantic of the node *node* and $\psi(edge)$ the semantic of the edge *edge* as follow:

- $\forall i, f_i = \psi(\Psi_i)$
- $\forall edge \in \Psi \cup E, \psi(edge) = \rho(edge.\gamma)(\phi(edge.node))$
- $\forall node \in V, \phi(node) = \psi(node.if0) \star \psi(node.if1)$

We assume the function ϕ defined on all terminals T (we always assume, all terminals to have a different interpretation through ϕ).

4.1.1 Constraints on Transformations

We denote $\mathbb{Y}$ the set of all transformation descriptors and $\mathbb{T} = \{\rho(\gamma) \mid \gamma \in \mathbb{Y}\}$ the set of transformations (and $\forall n, m \in \mathbb{N}, \mathbb{T}_{n,m} = \mathbb{T} \cap (\mathbb{F}_n \longrightarrow \mathbb{F}_m)$).
Transformation descriptors are canonical: $\forall t \in \mathbb{T}, \exists! \gamma \in \mathbb{T}, \rho(\gamma) = t$.
Transformations are composable: $\forall t \in \mathbb{T}_{n,m}, t' \in \mathbb{T}_{m,l}, (t' \circ t) \in \mathbb{T}_{n,l})$. We define the set of asymmetric transformations by $A_n = \mathbb{T}_{n,n}$. We denote the set of symmetric transformations by $S_{n,m}$, such that $\forall t \in \mathbb{T}_{n,m}, \exists!(s, a) \in (S_{n,m} \times A_n), t = s \circ a$ and $S_{n,n} = \{Id_n\}$.
Symmetric transformations are composable: $\forall s \in S_{n,m}, s' \in S_{m,l}, (s' \circ s) \in S_{n,l}$.
We denote $\mathbb{F}_m^S = \{f \in \mathbb{F}_m \mid \forall n \leq m, \forall t \in \mathbb{T}_{n,m}, f' \in \mathbb{F}_n, t(f') = f\}$ the set of S-free Boolean functions or arity n. Boolean functions are uniquely S-extractable: $\forall f \in \mathbb{F}_m, \exists!(t, \tilde{f}) \in \left(S_{n,m} \times \mathbb{F}_n^S\right), f = t(\tilde{f})$. Two Boolean functions $f, f' \in \mathbb{F}_n$ are A-equivalent iff $\exists a \in A_n, f = a(f')$. Hence, if two functions are not A-equivalent, they are A-distinct. Furthermore, we say that a Boolean function $f \in \mathbb{F}_n$ is A-invariant free iff $\forall a \in A_n, f \neq a(f)$. We define $\mathbb{F}_n^A$ the subset of Boolean function $\mathbb{F}_n$ which are A-invariant free. Therefore, we define the constraint that $\forall n \in \mathbb{N}, \mathbb{F}_n^S \subset \mathbb{F}_n^A$ (i.e. a S-free Boolean function is also A-invariant free). N.B.: Several sets $S_{*,*}$ may fit this definition.

Remark: A GROBDD should ensure by construction that (1) all nodes are S-free and (2) all nodes are A-distinct.

Constraints on Terminals Terminals are S-free ($\forall t \in T, \phi(t) \in \mathbb{F}_0^S$) and A-distinct ($\forall t, t' \in T, \phi(t) = \phi(t') \Rightarrow t = t'$). We assume defined the function $E_0 : \mathbb{F}_0 \longrightarrow \texttt{edge}$, such that $\forall b \in \mathbb{F}_0, \psi(E_0(b)) = b$.

4.1.2 Buildable (definition of B)

Let X be a function, we denote I_X the identifier of an hypothetical node whose semantic interpretation is X. For all $n \in \mathbb{N}$, we denote $\mathbb{I}_n$ the set of identifiers corresponding to node representing functions of arity n.

We define B an algorithm over $\mathbb{Y}$ which respect the signature:

$$\begin{aligned}
&\texttt{B} : \mathbb{Y}_{n_0,m} \times \mathbb{I}_{n_0} \longrightarrow \mathbb{Y}_{n_1,m} \times \mathbb{I}_{n_1} \longrightarrow \\
&\quad | \ \text{ConsNode} \ \ \mathbb{Y}_{n',m} \times (\mathbb{Y}_{n_x,n'} \times \mathbb{I}_{n_x}) \times (\mathbb{Y}_{n_y,n'} \times \mathbb{I}_{n_y}) \\
&\quad | \ \text{Merge} \ \ \mathbb{Y}_{n_z,m} \times \mathbb{I}_{n_z}
\end{aligned}$$

(with $x, y, z \in \{0, 1\}$)
Furthermore, for all $(t_g, I_g, t_h, I_h) \in \mathbb{T}_{n_0,m} \times \mathbb{I}_{n_0} \times \mathbb{T}_{n_1,m} \times \mathbb{I}_{n_1}$: $\texttt{B}(t_g, I_g, t_h, I_h) = \texttt{ConsNode}(t, (t'_X, I_X), (t'_Y, I_Y)) \Rightarrow f = t\,(t'_X(X) \star t'_Y(Y)))$ and, $\texttt{B}(t_g, I_g, t_h, I_h) = \texttt{Merge}(t'_Z, I_Z) \Rightarrow f = t'_Z(Z)$ (with $X, Y, Z \in \{g, h\}$ and $f = t_g(g) \star t_h(h)$).

Definition 7 *a node is B-stable*
Let G be a GROBDD, we denote v an internal node of G. We denote $t_0 = v.if0.\gamma, I_0 = v.if0.node, t_1 = v.if1.\gamma$ and $I_1 = v.if1.node$. The node v is said B-stable iff $\texttt{B}(t_0, I_0, t_1, I_1) = \texttt{ConsNode}(Id, (t_0, I_0), (t_1, I_1))$.

The exhaustive list of sufficient constraints are list in Annexes 7.1 (namely: B is B-stable, B is S-free preserving and B is A-distinct preserving).

4.2 Reduction Rules

In addition to the previous constraints, we define two reduction rules:

1. The syntactical reduction : all sub-graphs are different up to graph-isomorphism (i.e. all identical sub-graphs are merged)
2. The local semantic reduction : all internal node $v \in V$ is B-stable.
3. All node have at least one incoming edge. A GROBDD is said reduced if it satisfies the reduction rules.

In this section we prove:

1. For all vector of Boolean function F it exists a reduced GROBDD G representing it.
2. A reduced GROBDD G is semi-canonical, defined as :
   (a) For all node $v \in V$, $\phi(v)$ is S-free.
   (b) For all pair of nodes v, v', $\phi(v)$ and $\phi(v')$ are A-distinct.
   (c) If two edges a and a' represent the same function, then $a.\gamma = a'.\gamma \wedge a.node = a'.node$.
3. Between two reduced GROBDD G and G' representing the same vector of Boolean functions F, it exists a one-to-one mapping $\sigma : V \longrightarrow V'$ such that $\forall v, v' \in V \times V', \sigma(v) = v' \Rightarrow (\exists a \in A_*, \phi(v) = \rho(a)(\phi(v'))$.

We refer the reader to Annexes (cf. Section 7.2) for a proof that under the constraint of the previous subsection (formally defined in Annexes, Section **??**), these statements are correct.

4.3 Conclusion

In this section, we introduced the concept of Generalized Reduction of Ordered Binary Decision Diagrams (`GROBDD`) and provided a set of conditions and reduction rules, which are sufficient to guarantee the reduced structure to be semi-canonical. This approach still allows to perform the equality test in a reasonable amount of time: proportional to the size of the transformation descriptor, which (for practical reasons) should be polynomial in the arity of the post-transformation function. In the remaining of this report, a `GROBDD` is to be assumed reduced. In the next section, we will introduce the "`U`seless variables extraction" variant (or "`U` extract" for short) as being a simple model (called "U") in the `GROBDD` formalism that we just introduced.

5 Useless Variable Extraction : `GROBDD` model NU

5.1 Motivation

There are several interests in extracting useless variables, the more obvious one is that it tends to reduce the number of nodes (relatively to a regular ROBDD representing the same function with the same order). Here is a list of other interesting properties:

- When formalizing a problem it may appear that it exists a sub-problem that is present several times but on different variables with the same relative order, for example: in the n-queens problem : "there is exactly one queen per line" is a constraint that either you replicate or solve once and replicate the result. With the "`U` extract" variant, you can solve the problem once and then, by creating the appropriate edge, replicating the solution without creating new nodes.
- Given a function, you can know which variables are useless, just by looking at the transformation descriptor, thus without going through the all structure.

5.2 Definition

Definition of the Transformation Descriptor Set (TDS) We define the Transformation Descriptor Set (TDS) of the "output Negation" + "Useless variable extraction" model (or "NU" model for short) by $\mathbb{Y} = \mathbb{B} \times \bigcup_{n\in\mathbb{N}} \mathbb{B}^n$. Let γ be a transformation, we denote $\gamma.neg \in \mathbb{B}$ the first component of γ and $\gamma.sub \in \bigcup_{n\in\mathbb{N}} \mathbb{B}^n$ the second component of γ. Furthermore, we denote $\gamma.sub_k \in \mathbb{B}$ the k-th component of the Boolean vector $\gamma.sub$. For all transformation descriptor $\gamma \in \mathbb{B} \times \mathbb{B}^m$, we define $m(\gamma) = m$ and $n(\gamma) = \sum_{0\leq k<m} \gamma.sub_k$. We define $\mathbb{Y}_{n,m} = \{\gamma \in \mathbb{Y} \mid n(\gamma) = n \wedge m(\gamma) = m\}$. For all transformation descriptor $\gamma \in \mathbb{Y}_{n,m}$, we denote $\rho(\gamma) \in \mathbb{F}_n \to \mathbb{F}_m$ its semantic interpretation.

Informally, let $\gamma \in \mathbb{Y}_{n,m}$ be a transformation descriptor and $f \in \mathbb{F}_n$ be a function without useless variables, then the k-variable of the Boolean function $\rho(\gamma)(f)$ is useless iff $\gamma.sub_k$ is false.

Additionally, we define the set of terminals $T = \{0\}$, we define $\phi(0) = () \longrightarrow 0 \in \mathbb{F}_0$, we define $\forall b \in \mathbb{B}, E_0(() \longrightarrow b \in \mathbb{F}_0) = \{\gamma = \{neg = b, sub = ()\}, node = I_0\}$.

Definition of the semantic interpretation For all transformation descriptor $\gamma \in \mathbb{Y}_{n,m}$, we denote $S_\gamma = (i_1 < i_2 < \cdots < i_n)$ the exhaustive list for which $\gamma.sub_k$ is true. Therefore, for all function $f \in \mathbb{F}_n$ we define $\rho(\gamma)(f) = (x_1, \ldots, x_m) \longrightarrow \gamma.neg \oplus f(x_{i_1}, x_{i_2}, \ldots, x_{i_n})$. For convenience reasons, we allow ourselves to define the $\gamma.sub$ component using the S_γ notation instead of the Boolean vector one.

The `GROBDD` model NU is semi-canonical We can prove (cf. Annexes, Section 7.3) that the model "NU" satisfies all constraints formulated in Section 4 (Introduction of `GROBDD`). Therefore, we proved that a `GROBDD` which uses the "NU" model is semi-canonical. Actually, as the building algorithm `B` only uses the equality test between node's identifier, we can prove that the structure is canonical (up to graph isomorphism). In the next section, we compare representations generated by this new variant and regular ROBDD (with just the "output negation" variant) on three different benchmarks in terms of number of nodes and estimated memory cost.

5.3 Results

In this section, we start by quickly presenting our implementation of previous concepts, then, average results of our approach on three different benchmarks (involving circuits and CNF formulas), finally, we expose the impact on using different `GROBDD`models to represent solutions of the N-Queens problem.

5.3.1 Implementation details

Our implementation in OCaml of the previous concepts is available on GitHub as part of the DAGaml framework within the `grobdd` branch [7]. The core program, i.e. the `GROBDD` abstraction, is about 1 900 lines of code. Implementation details for models is about 6 100 lines of code, with 728 lines for the model "NU", the remaining lines implementing models "NNI" (1 939 lines, mentioned in Section 4, is a generalization of "inputs negation"), "NU-X" (2 587 lines, mentioned in Section 4, allows to extract "1-predictions"), but also regular ROBDD (318 lines) and ZBDD (201 lines). Various useful tools are implemented (about 4 300 lines). All in all, this project is about 12 400 lines long. The implementation is designed in order to maximize code reusing through several layers of abstraction. This choice was made in order to reduce the size of the source code and maximizing the chance of finding evidence of bugs if one was to be introduced. Additionally, a significant part of the source code and computation time is dedicated to self-checking the consistency of the operation, in order to further reduce the risk of undetected issue. During this project, we focused our efforts into finding and validating new generalizations ("NU", "NNI" and "NU-X" so far) which would reduce the number of nodes. However, we did

	lgsynth91		iscas99		satlib/uf20-91	
model	#node	**memo**	#node	**memo**	#node	**memo**
Z	+233%	+203%	+162%	+135%	-41%	-42%
NU	-25%	-42%	-25%	-41%	-3%	-3%
NNI	-60%	-64%	-57%	-61%	-29%	-14%
NU-X	-64%	-68%	-55%	-58%	-93%	-95%

Table 1: Average improvement in term of number of nodes and estimated memory cost. Results are given relatively to regular ROBDD(with the "output negation" variant"), Z represents regular Zero Suppressed Binary Decision Diagram (ZBDD), and NU, NNI and NU-X represent GROBDD model briefly introduced in Section 4.1

not spent much time on minimizing the memory cost of our implementation, therefore, the estimated memory cost introduces in the next section should be considered as an indicator. Especially, because the nodes in a regular ROBDD are of constant size (e.g. 22 bytes in Minoto et al. implementation [9]), however the model "NU" ("NNI" and "NU-X" as well) use a variable size encoding of the transformation descriptor, which should complexifies the memory management of an industrial-proof software using these concepts.

5.3.2 Results

We compare the number of nodes and the estimated memory cost for each one of the following model : N (regular ROBDD + "output negation" variant), Z (ZBDD), NU (GROBDD with "output negation" and "U extract"), NNI (GROBDD with "NNI extract") and and NU-X ("output negation", "U extract" and "X extract"), this variant allows to capture 'logical deductions' similar to what is done in SAT-solver with the unit propagation heuristic).

We tested our implementation on three different benchmarks : lgsynth91 [11], iscas99 [5] and satlib/uf20-91 [6]. Benchmarks lgsynth91 and iscas99, which represent various circuits, were pre-processed using the framework of logic synthesis ABC [1], in order to turn an arbitrary circuits into a simpler to parse And-Inverter-Graph (AIG). The benchmark satlib/uf20-91 represents a set of 1 000 satisfiable CNF formulas with 20 variables and 91 clauses. Results are given relatively to the model N. In columns "#node" we have the relative number of nodes and in columns "**memo**" the relative estimated memory cost.

5.3.3 The N-Queens problem

The N-Queens problem, is the problem of positioning N queens on an $N \times N$ chessboard, such that no queen threaten an other. This problem can easily be reformulated as a Boolean function (either using an AIG or a CNF formula) by representing each cell as Boolean representing the statement: "there is a queen in this cell" and then be compiled into a GROBDD. We call this representation the "quadratic version" with each cell is represented (starting in the top left corner of chessboard, and then, line by line, down to the bottom right corner) A slightly more complex formulation, use the fact that there is exactly one queen per colon of the chessboard, thus, we can used an integer ($\log n$ Booleans) to represent its position in the colon. We call this representation the "pseudo-linear

quadratic version					pseudo-linear version				
		#node					#node		
N	#SAT	NU	NNI	NU-X	N	#SAT	NU	NNI	NU-X
1	1	1	0	0	1	1	1	0	0
2	0	0	0	0	2	0	0	0	0
3	0	0	0	0	3	0	0	0	0
4	2	29	14	1	4	2	14	6	1
5	10	166	26	6	5	10	73	15	6
6	4	129	36	3	6	4	61	16	3
7	40	1098	106	30	7	40	348	39	30
8	92	2450	262	70	8	92	645	77	65

Table 2: Comparing models "NU", "NNI" and "NU-X" on the representation of the solutions of the N-Queens problems using a quadratic encoding (one variable per cell) or a pseudo-linear encoding (one integer or logarithmic size per queen).

version", with integer being represented as interleaved Boolean vectors starting from the most significant bits. Numbers of node for both version on each model are summarized in Table 2, we display a representation of the solutions the 5-queen problem (quadratic version) in Figure 1.

6 Conclusion

ROBDD allows to efficiently manipulate functions appearing in various fields of computer science such as: Bounded Model Checking, Planning, Software Verification, Automatic Test Pattern Generation, Combinational Equivalence Checking or Combinatorial Interaction Testing.

However, ROBDD manipulation is memory intensive and several variants exist (such as "output negation") in order to reduce the memory cost.

In this report, we introduced a new class of variant called `GROBDD` (Generalized Reduction of Ordered Binary Decision Diagram). We presented a set of constraints which ensure a `GROBDD` to be semi-canonical (i.e. canonical up to graph-isomorphism and A-equivalence, introduced in Section 4). Moreover, we defined five `GROBDD` models called "N" (model equivalent to ROBDDs with the "output negation" variant), "Z" (equivalent to ZBDD), "NU", "NNI" and "NU-X". Models "NU", "NNI" and "NU-X" allow to significantly reduce the number of nodes (relatively to the model "N").

Future work will be focus on merging the ""`NNI` extract"" variant and ""`X` extract"" variant into the "NNI-X extract" variant. Moreover, we will prove the correctness of "NNI extract" and "X extract" variants, improve the binary representation of current transformation descriptors, implement quantification operators, implement heuristics to improve the compilation of CNF formulas.

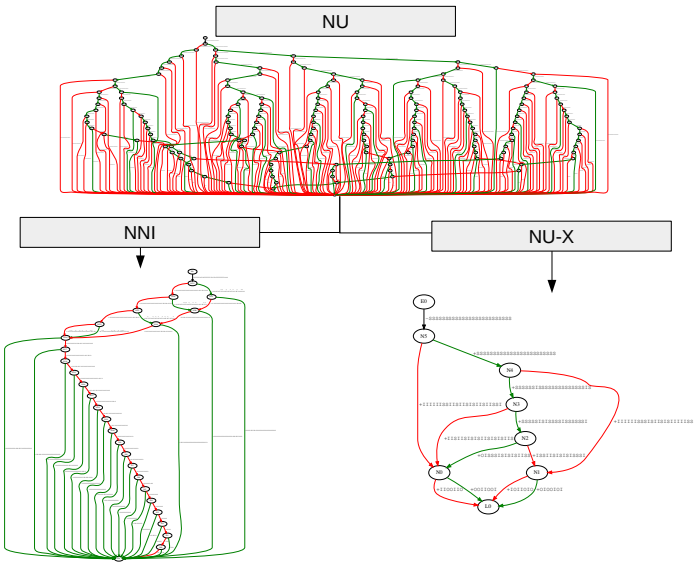

Figure 1: Representation of the 5-queens, using the "NU" model ("output negation" and "Useless variables extraction"), the "NNI" model ("input Negation and output Negation Invariant extraction"), finally the "NU-X" model ("NU" model and "1-prediction extraction").

References

[1] Berkeley Logic Synthesis and Verification Group, Berkeley, Calif. *ABC: A System for Sequential Synthesis and Verification*. `http://www.eecs.berkeley.edu/~alanmi/abc/`.

[2] Karl S. Brace, Richard L. Rudell, and Randal E. Bryant. Efficient implementation of a BDD package. In *Proceedings of the 27th ACM/IEEE Design Automation Conference*, DAC '90, pages 40–45, New York, NY, USA, 1990. ACM.

[3] Randal E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Trans. Comput.*, 35(8):677–691, August 1986.

[4] Jerry R. Burch and David E. Long. Efficient boolean function matching. In *Proceedings of the 1992 IEEE/ACM International Conference on Computer-aided Design*, ICCAD '92, pages 408–411, Los Alamitos, CA, USA, 1992. IEEE Computer Society Press.

[5] F. Corno, M.S. Reorda, and G. Squillero. Rt-level itc'99 benchmarks and first atpg results. *Design Test of Computers, IEEE*, 17(3):44–53, Jul 2000.

[6] Holger H. Hoos and Thomas Stützle. Satlib: An online resource for research on sat. pages 283–292. IOS Press, 2000.

[7] Thibault J. Abstract manipulations of directed acyclic graph using ocaml. `https://github.com/JoanThibault/DAGaml/tree/grobdd-dev`, 2016.

[8] D. M. Miller and R. Drechsler. Dual edge operations in reduced ordered binary decision diagrams. In *Circuits and Systems, 1998. ISCAS '98. Proceedings of the 1998 IEEE International Symposium on*, volume 6, pages 159–162 vol.6, May 1998.

[9] S. Minato, N. Ishiura, and S. Yajima. Shared binary decision diagram with attributed edges for efficient boolean function manipulation. In *27th ACM/IEEE Design Automation Conference*, pages 52–57, Jun 1990.

[10] Fabio Somenzi. Binary decision diagrams. 1999.

[11] S. Yang. Logic synthesis and optimization benchmarks user guide: Version 3.0. Technical report, 1991.

Contents

7 Annexes

7.1 GROBDD: Buildable (definition of B)

Let X be a function, we denote I_X the identifier of an hypothetical node whose semantic interpretation is X.

We define B an algorithm over $\mathbb{Y}$ which respect the signature:

$$\begin{aligned}&\mathtt{B} : \mathbb{Y}_{n_0,m} \times \mathbb{I}_{n_0} \longrightarrow \mathbb{Y}_{n_1,m} \times \mathbb{I}_{n_1} \longrightarrow \\ &\quad | \ \text{ConsNode} \ \ \mathbb{Y}_{n',m} \times (\mathbb{Y}_{n_x,n'} \times \mathbb{I}_{n_x}) \times (\mathbb{Y}_{n_y,n'} \times \mathbb{I}_{n_y}) \\ &\quad | \ \text{Merge} \ \ \mathbb{Y}_{n_z,m} \times \mathbb{I}_{n_z}\end{aligned}$$

(with $x, y, z \in \{0, 1\}$)
Furthermore, for all $(\gamma_g, I_g, \gamma_h, I_h) \in \mathbb{Y}_{n_0,m} \times \mathbb{I}_{n_0} \times \mathbb{Y}_{n_1,m} \times \mathbb{I}_{n_1}$,

$$\mathtt{B}(\gamma_g, I_g, \gamma_h, I_h) = \mathtt{ConsNode}(\gamma, (\gamma', I_X), (\gamma'', I_Y)) \Rightarrow f = \rho\,(\gamma)\,(\rho\,(\gamma')\,(X) \star \rho\,(\gamma'')\,(Y))$$

$$\mathtt{B}(\gamma_g, I_g, \gamma_h, I_h) = \mathtt{Merge}(\gamma''', I_Z) \Rightarrow f = \rho(\gamma''')(Z)$$

(with $X, Y, Z \in \{g, h\}$ and $f = \rho(\gamma_g)(g) \star \rho(\gamma_h)(y)$)

Definition 8 *a node is B-stable*
Let G be a GROBDD, we denote v an internal node of G. We denote $\gamma_0 = v.if0.\gamma, I_0 = v.if0.node, \gamma_1 = v.if1.\gamma$ and $I_1 = v.if1.node$. The node v is said B-stable iff $\mathtt{B}(\gamma_0, I_0, \gamma_1, I_1) = \mathtt{ConsNode}(Id, (\gamma_0, I_0), (\gamma_1, I_1))$.

Constraint 1 *B is B-stable*

$$\begin{aligned}\forall \gamma_f, I_f, \gamma_g, I_g, \mathtt{B}(\gamma_f, I_f, \gamma_g, I_g) = \mathtt{ConsNode}(\gamma, (\gamma_0, I_0), (\gamma_1, I_1)) \\ \Rightarrow \mathtt{B}(\gamma_0, I_0, \gamma_1, I_1) = \mathtt{ConsNode}(Id, (\gamma_0, I_0), (\gamma_1, I_1))\end{aligned}$$

Informally, when B returns a node, this node is B-stable.

Constraint 2 $\mathbb{Y}-$`node` *are B-stable*

1. *We assume all $\mathbb{Y}-$`node` v to be B-stable*

$$\mathtt{B}(v.\gamma_0, v.I_0, v.\gamma_1, v.I_1) = \mathtt{ConsNode}(Id, (v.\gamma_0, v.I_0), (v.\gamma_1, v.I_1))$$

Constraint 3 *B is S-free preserving*
The algorithm B is said S-free preserving iff for all $\mathbb{Y}-$`node` v, $\phi(v)$ is S-free.

Constraint 4 *B is A-reduction preserving*
The algorithm B is said A-reduction preserving iff

$$\forall v, w \in \mathbb{Y}-\mathtt{node}, \phi(v) \sim_A \phi(w) \Rightarrow v = w$$

7.2 GROBDD: Reduction Rules and Semi-Canonicity Theorem

We define a GROBDD model as the triple $(\mathbb{Y}, \mathtt{C}, \mathtt{B})$. A valid model must satisfy all the previously mentioned constraints.

In addition to the previous constraints, we define two reduction rules:

1. The syntactical reduction : all sub-graphs are different up to graph-isomorphism (i.e. all identical sub-graphs are merged)
2. The local semantic reduction : all internal node $v \in V$ is B-stable.
3. All node has at least one incoming edge.

A GROBDD is said reduced if it satisfies the reduction rules. In this section we prove:

1. For all vector of Boolean function F it exists a reduced GROBDD G representing it.
2. A reduced GROBDD G is semi-canonical, defined as :
   (a) For all node $v \in V$, $\phi(v)$ is S-free.
   (b) The set $X = \{\phi(v_1), \ldots, \phi(v_N)\}$ representing the set of the semantic interpretation of the set of internal nodes V is A-reduced.
   (c) $\forall(\gamma, I, \gamma', I') \in (\mathbb{Y}_{n,m} \times \mathbb{I}_n)^2, \psi(\{\gamma = \gamma, node = I\}) = \psi(\{\gamma = \gamma', node = I'\}) \Rightarrow (\gamma, I) = (\gamma', I')$ (with I and I' being indexes of nodes in G).
3. Between two reduced GROBDD G and G' representing the same vector of Boolean functions F, it exists a one-to-one mapping $\sigma : V \longrightarrow V'$ such that $\forall v, v' \in V \times V', \sigma(v) = v' \Rightarrow (\exists a \in A_*, \phi(v) = \rho(a)(\phi(v'))$.

7.2.1 Existence

We inductively define the procedure E with:

- $\forall b \in \mathbb{F}_0, \mathtt{E}(b) = \mathtt{E}_0(b)$
- Let f be a Boolean function of arity n (with $n \geq 1$). Let G be a GROBDD representing functions f_0 (the negative restriction of f according to its first variable) and f_1 (the positive restriction of f according to its first variable.) by using the procedure E on f_0 and f_1. Let G' be the output of the Cons procedure on G, in order to create $f = f_0 \star f_1$ (expansion theorem).
  $E(f) = G'$ The GROBDD G' satisfies the reduction rules:
  - If no node is created, the proof is straightforward.
  - If a node is created, this node is syntactically unique by definition of Cons and is B-stable (as B is B-stable)

By construction, the procedure E (generalized to accept a vector of function as input) returns a GROBDD satisfying the reduction rules.

7.2.2 Semi-Canonical

Let G be a reduced `GROBDD`.

S-free and A-reduced For all node $v \in V$, we denote $h(v) = max(h(v.if0.node), h(v.if1.node))$ with $\forall t \in T, h(t) = 0$. For all $n \in \mathbb{N}$, we define $V_n = \{v \in V \mid h(v) \leq n\}$ For all $n \in \mathbb{N}$, we define the recurrence hypothesis $H(n)$:

- For all $v \in V_n$, $\phi(v)$ is S-free.
- The set of Boolean functions $X = \{\phi(v) \mid v \in V_n\}$ is A-reduced.

Initialization We prove $H(0)$ using the constraints that (1) terminals are S-free and (2) the set of terminal nodes is A-reduced.

Induction Let $n \in \mathbb{N}$, we assume $\forall k \leq n, H(k)$. Let v be a node of depth $n+1$, thus the depth of $v.if0.node$ and $v.if1.node$ is lower than n (we can apply the recurrence hypothesis). Therefore, the quadruple $\bar{v} = (v.if0.\gamma, v.if0.node, v.if1.\gamma, v.if1.node)$ is a $\mathbb{Y}-$`node`. Thus, using the constraints that `B` is S-free preserving, we prove that $\phi(v) = \phi(\bar{v})$ is S-free. Let v' be a node of depth $k \leq n+1$, we can prove that the quadruple $\bar{v'} = (v'.if0.\gamma, v'.if0.node, v'.if1.\gamma, v'.if1.node)$ is a $\mathbb{Y}-$`node`. Therefore, we can use the constraint that `B` is A-reduction preserving to prove that $\phi(\bar{v}) \sim_A \phi(\bar{v'}) \Rightarrow \phi(\bar{v}) = \phi(\bar{v'})$ However, $\phi(v) = \phi(\bar{v})$ and $\phi(v') = \phi(\bar{v'})$ Thus, $\forall v, w \in V_{n+1}, \phi(v) \sim_A \phi(v') \Rightarrow \phi(v) = \phi(v')$. Thus, the set of Boolean function $X = \{\phi(v) \mid v \in V_{n+1}\}$ is A-reduced. Therefore $(\wedge_{k \leq n} H(k)) \Rightarrow H(n+1)$.

Using the strong recurrence theorem, we prove that $\forall n \in \mathbb{N}, H(n)$. Therefore proving properties (2.a) "all nodes are S-free" and (2.b) "the set $X = \{\phi(v_1), \ldots, \phi(v_N)\}$ is A-reduced".

Semantic Reduction We prove the property "$\forall(\gamma, I, \gamma', I') \in (\mathbb{Y}_{n,m} \times \mathbb{I}_n)^2, \psi(\{\gamma = \gamma, node = I\}) = \psi(\{\gamma = \gamma', node = I'\}) \Rightarrow (\gamma, I) = (\gamma', I')$ (with I and I' being indexes of nodes in G)" by induction on $n \in \mathbb{N}$ the arity of $f = \psi(\{\gamma = \gamma, node = I\})$.

Initialization The induction property holds for $n = 0$:
Let $f \in \mathbb{F}_0$, we assume it exists a quadruple $(\gamma, I, \gamma', I') \in (\mathbb{Y}_{n,m} \times \mathbb{I}_n)^2$ such that $f = \psi(\{\gamma = \gamma, node = I\}) = \psi(\{\gamma = \gamma', node = I'\})$. However, $\mathbb{Y}_{0,0} = \{Id_0\}$, therefore, γ and γ' are asymmetric transformation descriptors. Hence, $\rho(a)(\phi(I)) = \rho(a')(\phi(I'))$, thus $\phi(I) \sim_A \phi(I')$. Using the constraint, that terminals are A-reduced and canonical, we have that $\phi(I) = \phi(I')$, thus $I = I'$.

Induction Let $k \in \mathbb{N}$, we assume the induction property holds for all $n \leq k$, let prove it holds for $n = k+1$. Let f be a Boolean function, we assume it exists a quadruple $(\gamma, I, \gamma', I') \in (\mathbb{Y}_{n,m} \times \mathbb{I}_n)^2$ such that $f = \psi(\{\gamma = \gamma, node = I\}) = \psi(\{\gamma = \gamma', node = I'\})$. We decompose γ and γ' to their symmetric and asymmetric components: $\gamma = s \circ a$ and $\gamma' = s' \circ a'$. Using the S-uniqueness constraint, on $f = \rho(s)(\rho(a)(\phi(I))) = \rho(s')(\rho(a')(\phi(I')))$, we have $s = s'$ and $\rho(a)(\phi(I)) = \rho(a')(\phi(I'))$. Therefore, $\phi(I) \sim_A \phi(I')$, however, we

proved that the set of the semantic interpretations of the nodes is A-reduced, thus $\phi(I) = \phi(I')$ Using the induction hypothesis (as $\phi(I)$ as an arity strictly smaller than $k+1$, thus smaller than k), we deduce that $I = I'$.

Therefore, applying the strong induction theorem, we prove the property (2.c).

7.3 Model NU: proof of semi-canonicity

transformations are canonical

Let $\gamma', \gamma'' \in \mathbb{Y}_{n,m}$ be a pair of transformation descriptors, such that for all function $f \in \mathbb{F}_n$, $\rho(\gamma')(f) = \rho(\gamma'')(f)$. Let $f \in \mathbb{F}_n$ a function with no useless variable. We denote $S_{\gamma'} = (i'_1, \ldots, i'_n)$ and $S_{\gamma''} = (i''_1, \ldots, i''_n)$. We absurdly assume that it exists an index k such that $i'_k \notin S_{\gamma''}$. Therefore, in one hand, the i'_k-th variable of $\rho(\gamma')(f)$ is useless. In the other hand, the i'_k-th variable of $\rho(\gamma'')(f)$ is mapped to some variable of f which by definition has no useless variable, therefore, the i_k-th variable of $\rho(\gamma'')(f)$ is not useless. Hence, it leads to a contradiction as $\rho(\gamma'')(f) = \rho(\gamma'')(f)$ and that useless and not useless are incompatible states. Therefore, it does not exist such an index k, thus, $\gamma'.sub = \gamma''.sub$. We absurdly assume that $\gamma'.neg \neq \gamma''.neg$, thus $\gamma'.neg = \neg\gamma''.neg$. Hence, it leads to a contradiction as $\rho(\gamma')(f)(\vec{0}) = \rho(\gamma'')(f)(\vec{0})$ and $\rho(\gamma')(f)(\vec{0}) = \neg\rho(\gamma'')(f)(\vec{0})$. Therefore, $\gamma'.neg = \gamma''.neg$. Hence, $\gamma' = \gamma''$.

verified constraints on transformations

Separable Let $\gamma \in \mathbb{Y}_{n,m}$ be a transformation descriptor, we denote $S_\gamma = (i_1, \ldots, i_n)$. We define the function $\triangle_\gamma$: $\mathbb{B}^m \longrightarrow \mathbb{B}^n$ by $\forall x \in \mathbb{B}^m, \triangle_\gamma\ (x_1, \ldots, x_m) = (x_{i_1}, \ldots, x_{i_n})$. And define the function $\triangledown_\gamma : \mathbb{B}^m \longrightarrow \mathbb{B} \longrightarrow \mathbb{B}$ by $\forall x \in \mathbb{B}^m, y \in \mathbb{B}, \triangledown_\gamma(x, y) = \gamma.neg \oplus y$. Hence, $\forall \gamma \in \mathbb{Y}_{n,m}, \forall f \in \mathbb{F}_n, \forall x \in \mathbb{B}^m, \rho(\gamma)(f)(x) = \triangledown_\gamma(x, f(\triangle_\gamma\ (x)))$.

Composable Let $\gamma \in \mathbb{Y}_{n,m}$ and $\gamma' \in \mathbb{Y}_{m,l}$, we denote $S_\gamma = (i_1, \ldots, i_n)$ and we denote $S_{\gamma'} = (i'_1, \ldots, i'_m)$. We define $C(\gamma, \gamma') = \gamma''$ with $\gamma'' = \{neg = \gamma.neg \oplus \gamma'.neg, sub = (i'_{i_1}, i'_{i_2}, \ldots, i'_{i_n})\}$. We can prove that $\forall f \in \mathbb{F}_n, \rho(\gamma')(\rho(\gamma)(f)) = \rho(C(\gamma, \gamma'))(f)$.

Decomposable We denote $A_n = \mathbb{B} \times \{1\}^n \subset \mathbb{Y}_{n,n}$ (with $n \in \mathbb{N}$) and $S_{n,m} = \{0\} \times \{x \in \mathbb{B}^m \mid \sum_k x_k = n\}$. We can prove that $\forall \gamma \in \mathbb{Y}_{n,m}, \exists a \in A_m, \exists! s \in S_{n,m}, \gamma = a \circ s$. We can notice that, for all $n \in \mathbb{N}$, $\rho(A_n) = \{Id, \neg\}$.

an S-free function is A-invariant free

We absurdly assume that it exists a Boolean function $f \in \mathbb{F}_n$ without useless variable (i.e. S-free), and a pair of distinct asymmetric transformations $a, a' \in A$ such that $\rho(a)(f) = \rho(a')(f)$. As A_n is of cardinal 2, we enumerate all cases, which lead to a contradiction as $f \neq \neg f$. Therefore f is A-invariant free.

terminal nodes are S-free

The function $\phi(0) = () \longrightarrow 0$ has no variable therefore, no useless variable, thus is S-free.

the set of semantic interpretation of terminal nodes is A-reduced

There is only one terminal node, therefore the set is A-reduced.

7.3.1 Definition of the building algorithm B

We define the building algorithm B as follow:

```
1  B(γ0, I0, γ1, I1){
2     if (I0 = I1) ∧ γ0 = γ1 then{
3        Merge {γ = {neg = γ0.neg, sub = (0, γ0.sub1, ..., γ0.subn)}, node = I0}
4     }else{
5        we define γ, γ'0, γ'1 by:
6           γ.neg = γ0
7           γ'0.neg = 0
8           γ'1.neg = γ0 ⊕ γ1
9           Sγ = {i1 < ... < i_m'} = Sγ0 ∪ Sγ1
10          Sγ'0 = {j1 < ... < jn} the indexes of Sγ0 in Sγ.
11          Sγ'1 = {k1 < ... < k_n'} the indexes of Sγ1 in Sγ.
12       ConsNode (γ, (γ'0, I0), (γ'1, I1))
13    }
14 }
```

The proof that B is correct is left to the reader. Furthermore, we prove that $S_{\gamma'_0} \cup S_{\gamma'_1} = \{1, \ldots, m'\}$.

B is B-stable We have to prove that

$$\texttt{B}(\gamma_0, I_0, \gamma_1, I_1) \in \mathbb{Y}_{n,m}) = \texttt{ConsNode}(\gamma, (\gamma'_0, I'_0), (\gamma'_1, I'_1))$$

$$\Rightarrow \texttt{B}(\gamma'_0, I'_0, \gamma'_1, I'_1) = \texttt{ConsNode}(Id, (\gamma'_0, I'_0), (\gamma'_1, I'_1))$$

We absurdly assume that $\texttt{B}(\gamma'_0, I'_0, \gamma'_1, I'_1) \neq \texttt{ConsNode}(Id, (\gamma'_0, I'_0), (\gamma'_1, I'_1))$:

- We absurdly assume that $\texttt{B}(\gamma'_0, I'_0, \gamma'_1, I'_1) = \texttt{Merge}(\gamma, I)$. Therefore, by definition of B, $\gamma'_0 = \gamma'_1$ and $I'_0 = I'_1$, thus $\gamma_0 = \gamma_1$ and $I_0 = I_1$, hence it leads to a contradiction as $\texttt{B}(\gamma_0, I_0, \gamma_1, I_1) \in \mathbb{Y}_{n,m}) = \texttt{Merge}\ldots$.
- Therefore, $\texttt{B}(\gamma'_0, I'_0, \gamma'_1, I'_1) = \texttt{ConsNode}(\gamma', (\gamma''_0, I''_0), (\gamma''_1, I''_1))$:
  - We absurdly assume that $\gamma' \neq Id$. However, $\gamma'.neg = \gamma'_0.neg = 0$ (by definition of B, therefore $\gamma'.sub \neq (1, \ldots, 1) \in \mathbb{B}^n$. Thus, it exists i such that $\gamma'.sub_i = 0$, thus, it exists an index k' such that $k' \notin S_{\gamma'_0} \cup S_{\gamma'_1}$, which leads to a contradiction as $S_{\gamma'_0} \cup S_{\gamma'_1} = \{1, \ldots, m'\}$.
  - Therefore, $\gamma' = Id$. Thus, $\gamma''_0 = \gamma'_0$, $\gamma''_1 = \gamma'_1$, $I''_0 = I'_0$ and $I''_1 = I'_1$. Hence it leads to a contradiction as we assumed that $\texttt{B}(\gamma'_0, I'_0, \gamma'_1, I'_1) \neq \texttt{ConsNode}(Id, (\gamma'_0, I'_0), (\gamma'_1, I'_1))$.

Therefore, B is B-stable.

B is S-free preserving

Let $v = (\gamma_0, I_0, \gamma_1, i_1)$ be a $\mathbb{Y}-\texttt{node}$, therefore, the Boolean functions $\phi(I_0)$ and $\phi(I_1)$ are S-free and $\phi(I_0) \sim_A \phi(I_1) \Rightarrow I_0 = I_1$ We absurdly assume that the Boolean function $\phi(v) = \rho(\gamma_0)(\phi(I_0)) \star \rho(\gamma_1)(\phi(I_1))$ is not S-free, thus, it exists an index k such that the k-th variable of $\phi(v)$ is useless.

- If $k = 0$, then, $\rho(\gamma_0)(\phi(I_0)) = \rho(\gamma_1)(\phi(I_1))$. Using the S-uniqueness constraint ($\phi(I_0)$ and $\phi(I_1)$ being S-free), we prove that $\phi(I_0) \sim_A \phi(I_1$, thus $I_0 = I_1$. Hence, leading to a contradiction as the $\mathbb{Y}-$`node` v is assumed to be `B`-stable.

- If $k > 0$, then, the $(k-1)$-th variables of the Boolean functions $\rho(\gamma_0)(\phi(I_0))$ and $\rho(\gamma_1)(\phi(I_0))$ are useless. Using the S-uniqueness constraint ($\phi(I_0)$ and $\phi(I_1)$ being S-free), we prove that $\gamma_0.sub_{k-1} = \gamma_1.sub_{k-1} = 0$. Hence, leading to a contradiction as the $\mathbb{Y}-$`node` v is assumed to be `B`-stable.

`B` is A-reduction preserving

Let $v = (\gamma_0, I_0, \gamma_1, I_1)$ and $v' = (\gamma'_0, I'_0, \gamma'_1, I'_1)$ be $\mathbb{Y}-$`nodes`, such that $\phi(v) \sim_A \phi(v')$. Thus, it exists an asymmetric transformation descriptor a such that $\phi(v) = \rho(a)(\phi(v'))$ Therefore, the Boolean functions $\phi(I_0), \phi(I_1), \phi(I'_0)$ and $\phi(I'_1)$ are S-free and the set of Boolean functions $\{\phi(I_0), \phi(I_1), \phi(I'_0), \phi(I'_1)\}$ is A-reduced. Thus, using the expansion theorem, $\rho(\gamma_0)(\phi(I_0)) = \rho(a \circ \gamma'_0)(\phi(I'_0))$ and $\rho(\gamma_1)(\phi(I_1)) = \rho(a \circ \gamma'_1)(\phi(I'_1))$. Using the S-uniqueness constraint, we prove that $\phi(I_0) \sim_A \phi(I'_0)$ and $\phi(I_1) \sim_A \phi(I'_1)$ (and $\gamma_0 = a \circ \gamma'_0$ and $\gamma_1 = a \circ \gamma'_1$), thus, using the A-reduction property, $I_0 = I'_0$ and $I_1 = I'_1$.

Therefore, $v' = (\gamma'_0, I_0, \gamma'_1, I_1)$. We proved that $\gamma_0 = a \circ \gamma'_0$, however, v and v' and `B`-stable, thus $\gamma_0.neg = \gamma'_0.neg = 0$, thus $a = Id$. Hence, $\gamma_1 = \gamma'_1$. We proved that $v = v'$.