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 19, 2017

Abstract

The Reduced Ordered Binary Decision Diagram (ROBDD) [3, 10] is the state-of-the-art representation for Boolean functions and used in various fields such as logic synthesis, artificial intelligence or combinatorics. However, efficient manipulation of ROBDDs is memory expensive. Several variants exist which allows to capture some properties of Boolean functions in order to simplify their representation and reduce the computation time (e.g. "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 framework that aims at providing sufficient conditions in order to ensure the semi-canonicity of the representation. Using GROBDD, we designed three new variants, namely: "`U`seless variables extraction", "input `N`egation and output `N`egation `I`nvariant extraction" and "1-prediction extraction". These variants have been implemented using OCaml [7] and tested against several set of circuits [5, 11] and CNF formulas [6].

1 Introduction

Nowadays, most critical systems rely on digital circuits: in transports (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 that needs to be proven is the "equivalence checker".

In order to prove that two digital circuits are equivalent, there are two main algorithmic solutions: (1) the DPLL (Davis–Putnam–Logemann–Loveland) algorithm and (2) compilation to ROBDD (Reduced Ordered Binary Decision Diagram). 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). In this report we will focus on ROBDD: a canonical structure which represents a vector of functions as a multi-rooted hash-consed Directed Acyclic Graph (DAG). Compiling a function into a ROBDD is an exponential time (in the number of variables) process in the worst case, however, testing two functions for equality is a constant time operation.

However, unlike the DPLL algorithm, efficiently compiling a function into a ROBDD is memory expensive. Therefore, variants were invented in order to capture some semantic properties of the function, hence, reducing the size of the representation and the computation time. For example, Zero suppressed Binary Decision Diagrams (ZBDDs) are better suited to represent sparse functions. In this report we use the "output negation" variant [2], which extends the reduction rules in order to guarantee canonicity under negation, therefore, allowing 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 $\bar{f} = (x_1, \dots, x_n) \longrightarrow \neg f(\neg x_1, \dots, \neg x_n)$, therefore the reduction works similarly to the "output negation").

However, simply merging two variants may break the canonicity (e.g. ZBDD and "output negation"). Therefore, we designed the `GROBDD` (Generalized Reduction of Ordered Binary Decision Diagram) framework as a way of (1) defining sufficient constraints on allowed transformation that ensures the structure to be canonical (2) finding new variants using a systematic approach.

In order to show the effectiveness of our approach, we introduce three new variants: "`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 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.

2 Notations

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

Let A be a set of elements and $a, b \in \mathbb{N}$ a pair of integers, we denote $X_{[\![a,b]\!]} \in A^{b-a+1}$ a vector of $(b-a+1)$ elements of A and A_k its $(k-a+1)$-th element (with $k \in [\![a,b]\!]$).

We denote the set of Booleans $\mathbb{B} = \{0, 1\}$. The set of Boolean vectors 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 $f_{|b} = (x_1, \dots, x_n) \longrightarrow f(b, x_1, \dots, x_n)$. 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 functions of arity n. We denote $f \star g$ the Boolean function of arity $n+1$ defined by $f \star g = (x_0, x_1, \dots, x_n) = x_0 \longrightarrow_S f(x_1, \dots, x_n), g(x_1, \dots, x_n)$. 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 $\forall (x_1, \dots, x_n) \in \mathbb{B}^n, f(x_1, \dots, x_{i-1}, 0, x_i, \dots, x_n) = f(x_1, \dots, x_{i-1}, 1, x_i, \dots, x_n)$

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 $\forall (x_1, \dots, x_n), f(x_1, \dots, x_{i-1}, t, x_i, \dots, x_n) = r$.

Definition 5 *Polarity-Phase Invariant*
Let f be a Boolean function of arity n, and $X = (y, z_1, \dots, z_n) \in \mathbb{B}^{n+1}$ be a vector of $n+1$ Booleans. We say that X is a polarity-phase invariant of f, iff $\forall (x_1, \dots, x_n) \in \mathbb{B}^n, f(x_1, \dots, x_n) = y \oplus f(x_1 \oplus z_1, \dots, x_n \oplus z_n)$. 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 a row-echelon matrix of Booleans.

3 Reduced Ordered Binary Decision Diagram (ROBDD) and Canonicity

Definition of Binary Decision Diagram (BDD)

A Binary Decision Diagram is a directed acyclic graph $(V \cup T, \Psi \cup E)$ representing a vector of Boolean functions $F_{[\![1,k]\!]}$ 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:

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

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

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 functions, the equality test between two of this function is a constant time operation. Canonicity is ensured by using an association table (usually a hash table), which maps all nodes to an identifier (usually a pointer to the node itself). Therefore, before creating a new node, all procedure must first check that it does not already exists. If it does, the procedure retrieves its identifier, otherwise, generates a new identifier and adds the pair node-identifier into the association table. The association table is usually called *unique table.*

Zero Suppressed Binary Decision Diagram (ZBDD) The main difference between ROBDD and ZBDD stands in how they chose to remove nodes. In a ROBDD, nodes *node* with *node*.`if0` identical to *node*.`if1` are removed. In a ZBDD, nodes *node* with *node*.`if0` pointing to the terminal node 0 are removed. Additionally, as ZBDD and "output negation" are not compatible, there are two terminal nodes respectively denoted 0 and 1).

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

Creating new canonical variants is a painful task for several reasons, such as: (1) the combination of two canonical variants is not necessarily one, (2) proving that a variant is canonical from scratch needs to turn a local property into a global one and (3) we have no idea on what a variant should look like.

We designed the Generalized Reduction of Ordered Binary Decision Diagram (`GROBDD`) framework, in order to properly define what a model is. This definition would help us defining sufficient conditions on the variant which ensure the structure to be canonical (or semi-canonical). In this section, we formally define `GROBDD`s, then deduce a set of sufficient properties which ensure the canonicity (or the semi-canonicity) of a `GROBDD`.

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}. Furthermore, it allows to "copy" functions at (almost) no cost as the expression $f(x_1, x_3, x_5, x_7)$ and $f(x_0, x_1, x_3, x_5)$ share the same structure. This variant is a generalization of the "shifting variables" [9] variant.

The "`NNI` extract" variant allows output and inputs negation, therefore making the structure invariant under input and output negation (which ensures that there is at most 2^{2^n-n} nodes of arity n). However, keeping the structure canonical, requires to compute the set of polarity-phase invariants of a manipulated function (defined in Section 1). This was done by generalizing the work of Burch and Long [4] from phase-invariants (only inputs) to polarity-phase invariants (output and inputs). Thankfully, we can "extract" polarity-phase invariants into edges saving one variable per dimension of the linear space. This variant is a generalization of the following variants: "output negation" [2], "input negation" [9] and "dual edge" [8].

The "`X` extract" variant allows to "extract" 1-predictions (defined in Section 1). This variant is a generalization of ZBDD[1] and is compatible with "output negation" and "`U` extract" but not with "`NNI` extract".

Due to the limited length of this report, we will only detail the "`U` extract" variant (cf. Section 5) and will not detail further the "`NNI` extract" and "`X` extract" variants.

4.1 Definition of `GROBDD`

4.1.1 structure

A `GROBDD` is a directed acyclic graph $(V \cup T, \Psi \cup E)$ representing a vector of Boolean functions $F_{[\![1,k]\!]}$ is defined similarly to a ROBDD. Nodes are partitioned

[1]ZBDD is special case of "`X` extract", with the only extractable 1-prediction being $(0, 0, 0)$

into two sets: the set of internal nodes V and the set of terminal nodes T. Every internal node $v \in V$ has a field `index` which represents a unique identifier associated to each node and two outgoing edges respectively denoted `if0` and `if1`. 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`.

4.1.2 semantic interpretation

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:

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

We assume the function ϕ defined on all terminals T (we always assume, all terminals to have a different interpretation through ϕ). We define the set of transformations $\mathbb{T} = \{\rho(\gamma) \mid \gamma \in \mathbb{Y}\}$ as the set of semantic interpretation of transformation descriptors (and $\forall n, m \in \mathbb{N}, \mathbb{T}_{n,m} = \mathbb{T} \cap (\mathbb{F}_n \longrightarrow \mathbb{F}_m)$)..

4.1.3 definition: A `GROBDD` model

A `GROBDD` model is a 7-uplet $(T, \mathbb{Y}, \rho, \texttt{C}, \mathbb{I}, \texttt{B}, \mathcal{P})$:

1. T: The set of all terminal nodes
2. $\mathbb{Y}$: The set of all transformation descriptors
3. ρ: The semantic interpretation of a transformation descriptor
4. $\texttt{C} : \mathbb{Y} \longrightarrow \mathbb{Y} \longrightarrow \mathbb{Y}$: The `C`omposition operator, ensuring that $\rho(\texttt{C}(\gamma, \gamma')) = \rho(\gamma') \circ \rho(\gamma)$ (`C` can be a partial function).
5. $\mathbb{I}$: The set of identifiers.
6. $\texttt{B} : (\mathbb{Y} \times \mathbb{I}) \times (\mathbb{Y} \times \mathbb{I}) \longrightarrow$
   $\mid \texttt{ConsNode}\ \mathbb{Y} \times (\mathbb{Y} \times \mathbb{I}) \times (\mathbb{Y} \times \mathbb{I})$
   $\mid \texttt{Merge}\ \mathbb{Y} \times \mathbb{I}$
   The `B`uilding operator, ensuring that:
   (a) $\texttt{B}((\gamma, I_g), (\gamma, I_h)) = \texttt{ConsNode}(\gamma, (\gamma'_X, I_X), (\gamma'_Y, I_Y))$
   $\Rightarrow f = \rho(\gamma)\,(\rho(\gamma'_X)(X) \star \rho(\gamma'_Y)(Y)))$
   (b) $\texttt{B}((\gamma, I_g), (\gamma, I_h)) = \texttt{Merge}(\gamma'_Z, I_Z) \Rightarrow f = \rho(\gamma'_Z)(Z)$
   (with $f = \rho(\gamma_g)(g) \star \rho(\gamma_h)(h)$), (`B` can be a partial function)
7. $\mathcal{P} : V \longrightarrow \mathbb{B}$: The "local semantic reduction" predicate.

The `cons` procedure is defined by:

```
let cons grobdd arc0 arc1 = match
    $\texttt{B}((arc0.\gamma, arc0.\texttt{node.index}), (arc1.\gamma, arc1.\texttt{node.index}))$ with
  | ConsNode $(\gamma, (\gamma', I'), (\gamma'', I''))$ $\longrightarrow$
    let $N$ = $\{\texttt{if0} = \{\gamma = \gamma', \texttt{node} = I'\}, \texttt{if1} = \{\gamma = \gamma'', \texttt{node} = I''\}\}$ in
    let $I$ = if ($N$ exists in the unique table of grobdd)
      then ($N$'s identifier)
      else
      (generate a new identifier $I$ for $N$ and create the
          association ($I$, $N$) into grobdd's unique table) in
```

```
        {γ = γ, node = I}
    | Merge (γ,I) ⟶ {γ = γ, node = I}
```

Properties 6.a and 6.b on the Building operator are necessary and sufficient to ensure that

```
ψ(cons grobdd arc0 arc1) = (ψ arc0) ⋆ (ψ arc1)
```

4.1.4 Reduction rules

We define three reduction rules:

1. All sub-graphs are different up to graph-isomorphism (i.e. all identical sub-graphs are merged)
2. All internal nodes satisfy the "local semantic reduction" predicate.
3. All nodes have at least one incoming edge.

A GROBDD is said reduced iff it satisfies the reduction rules.

4.2 Constraints

We make a list of sufficient conditions on models which ensure the (semi-)canonicity of the structure. Most of them are conjectured to be necessary.

4.2.1 Transformation Set

The set of transformation descriptors must be uniquely mapped to the set of transformations[2]: $\forall t \in T, \exists! \gamma \in \mathbb{Y}, t = \rho(\gamma)$. Transformations produce bigger functions than their input[3]: $\forall n > m \in \mathbb{N}, \mathbb{T}_{n,m} = \emptyset$. Transformations are composable: $\forall n \leq m \leq l \in \mathbb{N}, \forall t \in \mathbb{T}_{n,m}, \forall t' \in \mathbb{T}_{m,l}, t' \circ t \in \mathbb{T}_{n,l}$. Hence, for all integer $n \in \mathbb{N}$, $\mathbb{T}_{n,n}$ is stable by composition. For all integer $n \in \mathbb{N}$, we define $A_n = \mathbb{T}_{n,n}$ the set of asymmetric transformations, such transformations are one-on-one mapping of Boolean functions $\mathbb{F}_n$ of arity n. Thus, for all pair $n, m \in \mathbb{N}$ of integers, we define the set of symmetric transformations $S_{n,m} \subset \mathbb{T}_{n,m}$, such that (1) $\forall t \in \mathbb{T}_{n,m}, \exists!(s,a) \in (S_{n,m} \times A_n), t = s \circ a$[4] and (2) $\forall n \in \mathbb{N}, S_{n,n} = \{Id_n\}$. Let $f \in \mathbb{F}_m (m \in \mathbb{N})$ be a Boolean function, we say that f is S-free iff $\forall n < m, \nexists (t, f') \in (\mathbb{T}_{n,m} \times \mathbb{F}_n), t(f') = f$. We denote $\mathbb{F}^S$ the set of S-free functions and for all integer $n \in \mathbb{N}$, $\mathbb{F}^S_n = \mathbb{F}^S \cap \mathbb{F}_n$ the set of S-free functions of arity n. For all function $f \in \mathbb{F}_m (m \in \mathbb{N})$, $\exists!(t, f') \in (\mathbb{T} \times \mathbb{F}^S), t(f') = f$ (conjectured necessary), we respectively denote $\bar{f}$ such a f' and $t_{\bar{f} \to f}$ such a t (if f is S-free, $\bar{f} = f$ and $t_{\bar{f} \to f} = Id$). We say that two Boolean functions $f, f' \in \mathbb{F}_n (n \in \mathbb{N})$ are A-equivalent iff $\exists t \in A_n, f = t(f')$, we denote $\sim_A$ this equivalence relation (it two functions are not A-equivalent, they are A-distinct). We say that a Boolean function $f \in \mathbb{F}_n (n \in \mathbb{N})$ is A-invariant free iff $\forall a \in A_n, f \neq a(f)$, we

[2]assuming otherwise, it exists a pair of distinct transformation descriptors that have the same semantic interpretation, therefore, we can create two structures which are distinct but equivalent

[3]assuming otherwise, let $t \in \mathbb{T}_{n,m}$ (with $n > m$) be such a transformation, t is not injective($\#\mathbb{F}_n > \#\mathbb{F}_m$), it exists a pair of distinct functions which have the same image f through t, therefore f may have several representations

[4]remark: splitting transformations into their symmetric and asymmetric components was our "Eureka!" to solve the problem of properly defining semi-canonicty, i.e. a form of canonicity which allow nodes to be A-equivalent instead of strictly identical between to distinct representations.

denote $\mathbb{F}^A \subset \mathbb{F}$ the set of A-invariant free function and $\mathbb{F}^A_n = \mathbb{F}^A \cap \mathbb{F}_n$ the set of A-invariant free function of arity n. We set the constraint that all S-free function are A-invariant free: $\mathbb{F}^S \subset \mathbb{F}^A$.

4.2.2 Terminals

Terminal nodes are interpreted as Boolean function of arity 0, thus, S-free. Terminal nodes are correct iff (1) $A_0 = \{Id\}$, $T = \{0, 1\}$, $\phi(0) = () \to 0$ and $\phi(1) = () \to 1$ OR (2) $A_0 \neq \{Id\}$, $T = \{0\}$ and $\phi(0) = () \to 0$. Terminals are S-free ($\forall t \in T, \phi(t) \in \mathbb{F}^S_0$) and A-distinct ($\forall t \neq t' \in T, \phi(t) \not\sim_A \phi(t')$). We assume defined the function $E_0 : \mathbb{F}_0 \longrightarrow E$, such that $\forall b \in \mathbb{F}_0, \psi(E_0(b)) = b$.

4.2.3 Building operator `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.

Definition 7 *a node is `B`-stable*
Let G be a `GROBDD`, we denote v an internal node of G. We denote $t_0 = v.\texttt{if0}.\gamma, I_0 = v.\texttt{if0}.\texttt{node}, t_1 = v.\texttt{if1}.\gamma$ and $I_1 = v.\texttt{if1}.\texttt{node}$. The node v is said B-stable iff $\texttt{B}(t_0, I_0, t_1, I_1) = \texttt{ConsNode}(Id, (t_0, I_0), (t_1, I_1))$. We define the "local semantic reduction" predicate $\mathcal{P}$, by for all node $v \in V$, $\mathcal{P}v$ is true iff v is B-stable.

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

4.3 `GROBDD` are (Semi-)Canonical

We refer the reader to Annexes (Section 7.2) for a proof that under previous constraints, a `GROBDD` has the following properties:

1. For all vector of Boolean function F there 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 $v, v' \in V$, $\phi(v)$ and $\phi(v')$ are A-distinct.
   (c) If two edges a and a' represent the same function, then $a.\gamma = a'.\gamma \wedge a.\texttt{node} = a'.\texttt{node}$.
3. Between two reduced `GROBDD` G and G' representing the same vector of Boolean functions F, there exists a one-to-one mapping $\sigma : V \longrightarrow V'$ such that $\forall v, v' \in V \times V', \sigma(v) = v' \Rightarrow (\exists a \in A_*, \phi(v) = \rho(a)(\phi(v'))$.

Furthermore, we can prove that, if the set of unique identifiers has no order $\mathbb{I}$ (i.e. the operator $<$ is not defined on $\mathbb{I}$) then the structure is canonical.

4.4 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. We implemented five variants respectively denoted `N` (equivalent to ROBDD with "output negation"), `Z` (equivalent to ZBDD), `NU` ("output negation" and "`U` extract"), `NNI` ("`NNI` extract"), `NU-X` (`NU` and "`X` extract"). In the next section, we (1) introduce the variant `NU` and (2) compare the variants `Z`, `NU`, `NNI` and `NU-X` against `N`.

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 there 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, we can solve the problem once and then, by creating the appropriate edge, replicating the solution without creating new nodes.
- Given a function, we can know which variables are useless, just by looking at the transformation descriptor, thus without going through the whole structure.

5.2 Definition

Definition of the Transformation Descriptor Set (TDS) We define the Transformation Descriptor Set (TDS) of the variant `NU` : "output Negation" + "`U` extract" by: $\mathbb{Y}_{n,m} = \{\{neg = b, sub = s\} \mid b \in \mathbb{B}, s \in \mathbb{B}^m, \sum_i s_i = n\}$

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 of indexes i for which $\gamma.sub_i$ 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.

	`lgsynth91`		`iscas99`		`uf20-91`		`uf50-218`	
variants	**nodes**	**mem.**	**nodes**	**mem.**	**nodes**	**mem.**	**nodes**	**mem.**
`Z`	+233%	+233%	+162%	+162%	-41%	-41%	-42%	-42%
`NU`	-26%	-39%	-25%	-20%	-3%	+7%	-3%	+22%
`NNI`	-60%	-63%	-56%	-49%	-30%	-10%	-39%	+5%
`NU-X`	-64%	-67%	-55%	-46%	-96%	-95%	-97%	-96%

Table 1: Average improvement in term of number of nodes and estimated memory cost. In columns "`nodes`" we have the relative number of nodes and in columns "**mem.**" the relative estimated memory cost.

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 model `NU` 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 of variant specific details about 6 100 lines of code: `N` (318 lines), `Z` (201 lines), `NU` (728 lines), `NNI` (1 939 lines) and `NU-X` (2 587 lines). Various useful tools are implemented (about 4 300 lines). All in all, this project is about 12 400 lines long. We did not spent much time on minimizing the memory cost of our implementation, therefore, the estimated memory cost introduced in the next section should be considered carefully (at most) as an indicator. One reason, is that the size of nodes in a ROBDD is constant (e.g. 22 bytes in Minoto et al. implementation [9]), however, the size of nodes in a `GROBDD` is variable (e.g. with the variant `NU`, a node which semantic interpretation is of arity n, costs about $(1 + 2n)$ bits). Having nodes of variable size should complexify the memory management of a memory aware implementation of `GROBDD`s.

5.3.2 Results

We compare the number of nodes and the estimated memory cost of variants `Z`, `NU`, `NNI` and `NU-X` relatively to `N`. This results were obtained on four different benchmarks: `lgsynth91` [11], `iscas99` [5], `uf20-91` [6] and `uf50-218` [6].

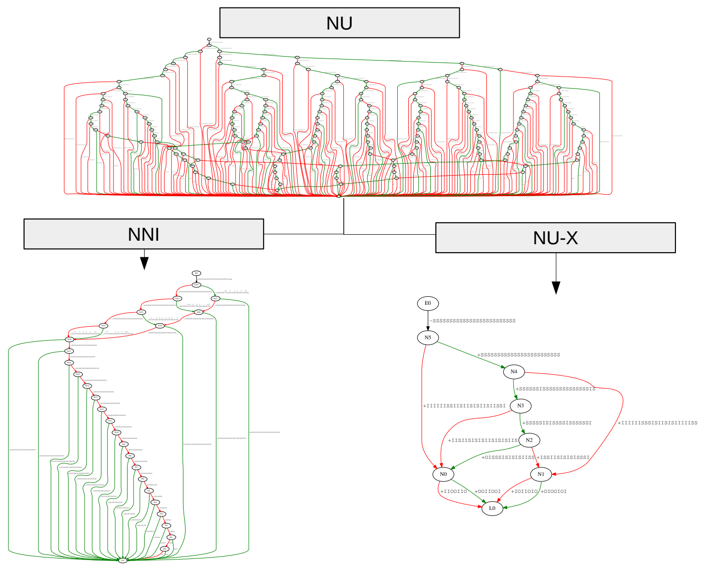

Figure 1: Solution of the 5-queens problem, using variants: `NU`, `NNI` and `NU-X`.

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 circuit into a simpler to parse And-Inverter-Graph (AIG). The benchmark `uf20-91` (respectively `uf50-218`) represents a set of 1 000 satisfiable CNF formulas with 20 (respectively 50) variables and 91 (respectively 218) clauses. Results are given relatively to the model `N`.

5.3.3 The N-Queens problem

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 of logarithmic size per queen).

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 use an integer ($\log n$ Booleans) to represent its position in the colon. We call this representation the "pseudo-linear version", with integer being represented as interleaved Boolean vectors starting from the most significant bits. Numbers of nodes for both version on each model are summarized in Table 2, we display a representation of the solutions of the 5-queens problem 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). We introduced three new variants: "`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). We defined five `GROBDD` models denoted: `N` (equivalent to ROBDD with "output negation"), `Z` (equivalent to ZBDD), `NU` ("output negation" and "`U` extract"), `NNI` ("`NNI` extract"), `NU-X` (`NU` and "`X` extract") and implemented them in DAGaml [7] an OCaml software.

Experimentation shows (Table 1) significant reduction of the number of nodes for both circuits [6, 11] and CNF formulas [5] when using our variants.

Future work will be focused 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.

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`)

Here are the remaining constraints on the building operator `B`. Let X be a function, we denote I_X the identifier of an hypothetical node whose semantic interpretation is X.

Constraint 1 *`B` is `B`-stable*

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

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

Definition 8 $\mathbb{Y}-\mathtt{node}$
A $\mathbb{Y}-\mathtt{node}_{l,m,n}$ is a quadruple $(\gamma_0, I_0, \gamma_1, I_1) \in \mathbb{Y}_{l,n} \times \mathbb{I}_l \times \mathbb{Y}_{m,n} \times \mathbb{I}_m$. Let v be a $\mathbb{Y}-\mathtt{node}$, we denote $v.\gamma_0$ (respectively $v.I_0$, $v.\gamma_1$ and $v.I_1$) the first (respectively second, third and fourth) components of v.

1. *For all $\mathbb{Y}-\mathtt{node}$ v, we always assume that functions $\phi(v.I_0)$ and $\phi(v.I_1)$ are `S`-free.*
2. *All pair f, f' of functions in $\{\phi(v.I_0) \mid v \in \mathbb{Y}-\mathtt{node}\} \cup \{\phi(v.I_1) \mid v \in \mathbb{Y}-\mathtt{node}\}$ are A-distinct.*
3. *$\mathbb{Y}-\mathtt{nodes}$ v are `B`-stable*

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

We extend the definition of ϕ, with : for all $\mathbb{Y}-\mathtt{node}$ v, $\phi(v) = \rho(v.\gamma_0)(\phi(v.I_0)) \star \rho(v.\gamma_1)(\phi(v.I_1))$.

Constraint 2 *`B` is S-free preserving*
The algorithm `B` is said S-free preserving iff for all $\mathbb{Y}-\mathtt{node}$ v, $\phi(v)$ is S-free.

Constraint 3 *`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

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 $h(v) = max(h(v.\texttt{if0.node}), h(v.\texttt{if1.node}))$ 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.
- For all pair of nodes $v, v' \in V_n$, $\phi(v)$ and $\phi(v')$ are A-distinct.

Initialization We prove $H(0)$ using the constraints that respective semantic interpretation of terminal nodes are (1) S-free and (2) A-distinct.

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 $v.\texttt{if0.node}$ and $v.\texttt{if1.node}$ is lower than n (we can apply the recurrence hypothesis). Therefore, the quadruple $\bar{v} = (v.\texttt{if0}.\gamma, v.\texttt{if0.node}, v.\texttt{if1}.\gamma, v.\texttt{if1.node})$ is a $\mathbb{Y}-\texttt{node}$. 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 $\bar{v'} = (v'.\texttt{if0}.\gamma, v'.\texttt{if0.node}, v'.\texttt{if1}.\gamma, v'.\texttt{if1.node})$ is a $\mathbb{Y}-\texttt{node}$. 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, $\forall v, w \in V_{n+1}, \phi(v) \sim_A \phi(v') \Rightarrow \phi(v) = \phi(v')$. Hence, the respective semantic interpretations of nodes in V_n are A-distinct. Therefore $\forall n \in \mathbb{N}, H(n) \Rightarrow H(n+1)$. Using the recurrence theorem, we prove that $\forall n \in \mathbb{N}, H(n)$. Therefore, proving properties (2.a) and (2.b).

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 there 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 there 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 $S_{\gamma'} = (i'_1, \dots, i'_n)$ and $S_{\gamma''} = (i''_1, \dots, i''_n)$. We absurdly assume that there 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 $S_\gamma = (i_1, \dots, i_n)$. We define the function $\triangle_\gamma$: $\mathbb{B}^m \longrightarrow \mathbb{B}^n$ by $\forall x \in \mathbb{B}^m, \triangle_\gamma\ (x_1, \dots, x_m) = (x_{i_1}, \dots, x_{i_n})$. And define the function $\nabla_\gamma : \mathbb{B}^m \longrightarrow \mathbb{B} \longrightarrow \mathbb{B}$ by $\forall x \in \mathbb{B}^m, y \in \mathbb{B}, \nabla_\gamma(x, y) = \gamma.neg \oplus y$. Hence, $\forall \gamma \in \mathbb{Y}_{n,m}, \forall f \in \mathbb{F}_n, \forall x \in \mathbb{B}^m, \rho(\gamma)(f)(x) = \nabla_\gamma(x, f(\triangle_\gamma\ (x)))$.

Composable Let $\gamma \in \mathbb{Y}_{n,m}$ and $\gamma' \in \mathbb{Y}_{m,l}$, we denote $S_\gamma = (i_1, \dots, i_n)$ and we denote $S_{\gamma'} = (i'_1, \dots, i'_m)$. We define $C(\gamma, \gamma') = \gamma''$ with $\gamma'' = \{neg = \gamma.neg \oplus \gamma'.neg, sub = (i'_{i_1}, i'_{i_2}, \dots, i'_{i_n})\}$. 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 there 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.

7.3.1 Definition of the building algorithm B

We define the building algorithm B as follow:

```
B($\gamma_0, I_0, \gamma_1, I_1$){
  if $(I_0 = I_1) \wedge (\gamma_0 = \gamma_1)$ then{
   Merge $\{\gamma = \{neg = \gamma_0.neg, sub = (0, \gamma_0.sub_1, \dots, \gamma_0.sub_n)\}, node = I_0\}$
  }else{
   we define $\gamma, \gamma_0', \gamma_1'$ by:
     $\gamma.neg = \gamma_0$
     $\gamma_0'.neg = 0$
     $\gamma_1'.neg = \gamma_0 \oplus \gamma_1$
     $S_\gamma = \{i_1 < ... < i_{m'}\} = S_{\gamma_0} \cup S_{\gamma_1}$
     $S_{\gamma_0'} = \{j_1 < ... < j_n\}$ the indexes of $S_{\gamma_0}$ in $S_\gamma$.
     $S_{\gamma_1'} = \{k_1 < ... < k_{n'}\}$ the indexes of $S_{\gamma_1}$ in $S_\gamma$.
   ConsNode $(\gamma, (\gamma_0', I_0), (\gamma_1', I_1))$
  }
}
```

The proof that B is correct is left to the reader. Furthermore, we prove that $S_{\gamma_0'} \cup S_{\gamma_1'} = \{1, \dots, m'\}$.

B is B-stable We have to prove that

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

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

We absurdly assume that $\mathtt{B}(\gamma_0', I_0', \gamma_1', I_1') \neq \mathtt{ConsNode}(Id, (\gamma_0', I_0'), (\gamma_1', I_1'))$:

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

Therefore, B is B-stable.

B is S-free preserving

Let $v = (\gamma_0, I_0, \gamma_1, i_1)$ be a $\mathbb{Y}-$`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, there 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, there 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 $v = v'$.