Leveraging Structural Analysis for Quantified Boolean Formulae

Joan Thibault Khalil Ghorbal

June 12, 2020

Abstract

We introduce a new normalized representation of Boolean formulas under conjunctive form, along with a normalization process. We term this representation *Reduced Block Triangular Form* (RBTF). Common combinatoric problems such as SAT, #SAT or AllSat can be efficiently solved on an RBTF. Furthermore, RBTF exhibits interesting properties toward improving QBF-solvers. The normalization process allows to iteratively decompose an initial formula into a sequence of subproblems according to an ordered partition of its variables. Preliminary results demonstrate the tractability of our approach.

1 Introduction

Using Shannon's well known representation theorem, one may state that representing a Boolean function with n variables (while assuming uniform distribution) requires (on average) 2^n bits. However, in practice, uniform distribution is not observed and structured functions are more likely to appear. This is especially the case in areas like software/hardware verification and synthesis, where complex functions are usually built from a combination of well-known simpler ones. Various representations of Boolean functions exist. They can be organized into two categories: (1) *description languages* (DL), which are used to describe problems (e.g., CNF, DNF, AIG, QBF) and (2) *representation languages* (RL) which are efficient when performing SAT-related queries, (e.g., BDD, ZDD, MLDD, DZD, d-DNNF).[1]

Conversion between representations is usually expensive, especially from DL to RL where it is usually exponential in term of both computation time and memory consumption. The representation languages listed above mostly focus on functional properties (e.g. useless and canalizing variables or disjoint-support decomposition) while overlooking structural properties like sparsity or variable dependencies.

In this work, we want to leverage structural properties of the initial representation [2]. We focus in particular on lightweight syntactic structure, in contrast to the more involved semantic structure like disjoint support decomposition or bi-decomposition [3]. We introduce a meta-representation, we term RBTF, which is at an intermediary stage between a given DL and a RL. Its key feature is to preserve the structural properties of the DL, while preserving the tractability of some relevant queries, such as SAT AnySat or #SAT (cf. [4] for a detailed list of relevant queries). Indeed, unsatisfiable formulas have a unique RBTF representative, allowing for SAT to be computable in constant time.

For our approach to work properly, we need a RL to be complete (that is, any Boolean function can be represented in the language), and yield $O(2^n)$ time and space operators for conjunctions and existential quantifier elimination (n being the number of variables in the considered

[1] CNF (Conjunctive Normal Form), DNF (Disjunctive Normal Form), AIG (And Inverter Graph), QBF (Quantified Boolean Formula), BDD (Binary Decision Diagram), ZDD (Zero-suppressed Decision Diagram), MLDD (Multi-Layer Decision Diagram), DSD (Disjoint Support Decomposition), d-DNNF (decision Deterministic Negation Normal Form), cf. [1], for relevant references.

formula).These requirements are met by the above-mentioned languages. For convenience, we choose BDD as the targeted representation language, which has the additional property of being canonical. We focus on DL which are conjunctions of formulae.

In the sequel, we briefly introduce RBTF with its associated reduction processes. RBTF is a conjunction in triangular form. As such it limits the impact of the variables to only local parts of the so constructed conjunction. In addition, RBTF enjoys a *local consistency* property: any solution of any conjunct can be extended into a solution of the whole formula. One may compile a QBF formula by interleaving the elimination of existential quantifiers using RBTF-based strategy and straightforward elimination of universal quantifiers.

2 Reduced Block Triangular Form

RBTF and its corresponding reduction processes arise at the intersection of three areas: structural analysis, dynamic programming and symbolic computation.

Firstly, we perform a structural analysis of the formula to determine a tree-shaped partition of the dependency graph of the variables similar to tree-decomposition [5]. Secondly, this partition is used to decompose the initial problem into sub-problems. During the so called *forward reduction process* (FRP), each terminal problem is existentially projected (by computing its existential closure) onto the support set of its ancestor in the tree-shaped partition. This first step allows to condense, in an ordered fashion, all sub-problems into a single one, the solution of which (if any) ensures that the whole system is satisfiable. A second computation phase called the *backward propagation process* (BPP), allows to propagate back the information by existentially projecting the solutions of each sub-problem (starting with the last one) onto its children's support variable in the tree-shaped partition.

Both FRP and BPP are based on the two following properties. Let ϕ be a formula and $\mathbf{v}$ a subset of variables, then

1. $\phi = \phi_{\mathbf{v}} \wedge \phi_{\overline{\mathbf{v}}}$, where $\phi_{\mathbf{v}}$ are conjuncts of ϕ which have a common variable with $\mathbf{v}$ and $\phi_{\overline{\mathbf{v}}}$ the remaining conjuncts.

2. ϕ is equivalent to $\phi \wedge \exists \mathbf{v}.\, \phi$, where $\exists \mathbf{v}.\, \phi$ denotes the formula obtained after existentially eliminating the variables in $\mathbf{v}$ from ϕ.

These two rules are applied iteratively with respect to the sequence computed by the structural analysis. Both properties are leveraged to perform FRP, while only the second is used during the BPP.

The FRP algorithm processes the formula by induction with respect to the sequence, as described in Figure 1. On a non-empty partition, the first rule is used to split the formula into $\phi_{\mathbf{v}_1}$ and $\phi_{\overline{\mathbf{v}_1}}$ according to the subset of variables $\mathbf{v}_1$. Then, the second rule is used to project ψ_1 into $\psi_1' := \exists \mathbf{v}_1.\, \psi_1$. The BDD ψ_1' encodes the constraints on $\overline{\mathbf{v}_1}$ that were implicit in $\phi_{\mathbf{v}_1}$. It is therefore added back to the formula $\phi_{\overline{\mathbf{v}_1}}$. At any step, if the unsatisfiable formula $\bot$ is detected, then one can immediately return $\bot$.

```
let rec FRP φ v =
match v with
| () ⟶ (BDD_of φ)
| (v1, v+) ⟶
  let φ_v1 ∧ φ_¬v1 = φ in
  let ψ1 = BDD_of φ_v1 in
   if ψ1 = ⊥ then ⊥ else
    let ψ1' = ∃v1. ψ1 in
    let ψ+ = FRP (ψ1' ∧ φ_¬v1) v+ in
     if ψ+ = ⊥ then ⊥ else
      (ψ1 ∧ ψ+)
```

Figure 1: Pseudo-code for the FRP

The primal graph of a formula, is defined as an undirected graph over the variables, such that there exists an arc between two variables if and only if they appear in a common conjunct.

Example 1 *Let us consider the following formula ϕ in conjunctive form defined over* 12 *variables, which primal graph is depicted in in Figure 2*

$$\begin{aligned}\phi := &(x_1 \vee x_4) \wedge (x_1 \vee x_5) \wedge (\neg x_4 \vee x_5) \wedge (x_2 \vee \neg x_4 \vee \neg x_5)\\ &\wedge (x_2 \vee x_4) \wedge (x_3 \vee x_4 \vee x_5) \wedge (\neg x_3 \vee x_5 \vee x_6) \wedge (\neg x_6 \vee x_9 \vee x_{10})\\ &\wedge (\neg x_9 \vee x_{10} \vee \neg x_{11}) \wedge (\neg x_9 \vee x_{11} \vee x_{12}) \wedge (\neg x_8 \vee x_9 \vee \neg x_{12})\\ &\wedge (x_6 \vee \neg x_7) \wedge (x_7 \vee x_8) \wedge (\neg x_7 \vee x_8)\end{aligned}$$

Figure 3 depicts the components of the intermediary formula obtained by applying FRP with respect to the ordered partition $\mathbf{v} = (\mathbf{v}_A, \mathbf{v}_B, \mathbf{v}_C, \mathbf{v}_D, \mathbf{v}_E)$, where $\mathbf{v}_A = \{x_1, x_2\}$, $\mathbf{v}_B = \{x_3, x_4, x_5\}$, $\mathbf{v}_C = \{x_7\}$, $\mathbf{v}_D = \{x_{10}\}$ and $\mathbf{v}_E = \{x_6, x_8, x_9, x_{11}, x_{12}\}$.

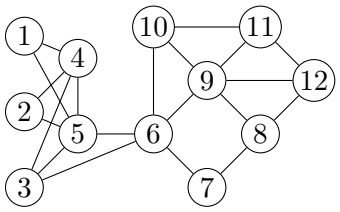

Figure 2: Primal graph of the formula ϕ of Example 1.

$\psi_A = (x_1 \vee x_4) \wedge (x_1 \vee x_5) \wedge (x_2 \vee \neg x_4 \vee \neg x_5) \wedge (x_2 \vee x_4)$	$\psi'_A = \exists x_1\, x_2\, \psi_A \equiv \top$
$\psi_B = \psi'_A \wedge (\neg x_4 \vee x_5) \wedge (x_3 \vee x_4 \vee x_5) \wedge (\neg x_3 \vee x_5 \vee x_6)$	$\psi'_B = \exists\, x_3\, x_4\, x_5\, \psi_B \equiv \top$
$\psi_C = (x_6 \vee \neg x_7) \wedge (x_7 \vee x_8) \wedge (\neg x_7 \vee x_8)$	$\psi'_C = \exists\, x_7\, \psi_C \equiv x_8$
$\psi_D = (\neg x_6 \vee x_9 \vee x_{10}) \wedge (\neg x_9 \vee x_{10} \vee x_{11})$	$\psi'_D = \exists\, x_{10}\, \psi_D \equiv \top$
$\psi_E = \psi'_B \wedge \psi'_C \wedge \psi'_D \wedge (\neg x_9 \vee x_{11} \vee x_{12}) \wedge (\neg x_8 \vee x_9 \vee \neg x_{12})$	$\psi'_E = \exists\, x_6\, x_8\, x_9\, x_{11}\, x_{12}\, \psi_E \equiv \top$

formula	depends on	formula	depends on
ψ_A	$\{x_1, x_2, x_4, x_5\}$	ψ'_A	$\{x_4, x_5\}$
ψ_B	$\{x_3, x_4, x_5, x_6\}$	ψ'_B	$\{x_6\}$
ψ_C	$\{x_7, x_6, x_8\}$	ψ'_C	$\{x_6, x_8\}$
ψ_D	$\{x_{10}, x_6, x_9, x_{11}\}$	ψ'_D	$\{x_6, x_9, x_{11}\}$
ψ_E	$\{x_6, x_8, x_9, x_{11}, x_{12}\}$	ψ'_E	$\emptyset$

Figure 3: Details of the *forward reduction process* on ϕ (cf. Example 1), the resulting intermediary formula, is denoted by $\psi := \psi_A \wedge \psi_B \wedge \psi_C \wedge \psi_D \wedge \psi_E$ which is equivalent to the input formula ϕ. Each formula would be represented internally as a BDD. For each index, ψ'_i denotes and expression of the formula obtained after existentially eliminating variables in v_i from ψ_i.

One may notice that the performance of the reduction heavily depends on the quality of the decomposition. We are currently investigating tree decomposition-based techniques to find a sequence that minimizes the overall worst case cost of the reduction.

Extension to QBF RBTF can be extended to QBF formulas, where existentially quantified variables are eliminated using FRP and universally quantified ones are eliminated conjunct-wise using the corresponding operation on BDDs. Furthermore, BPP can also be adapted to the QBF case, to compute back a representation for the solutions.

References

[1] J. Thibault and K. Ghorbal, "Ordered Functional Decision Diagrams," Inria, Research Report RR-9333, 2020. [Online]. Available: https://hal.inria.fr/hal-02512117

[2] S. H. Sæther, J. A. Telle, and M. Vatshelle, "Solving MaxSAT and #SAT on structured CNF formulas," *arXiv:1402.6485 [cs]*, Feb. 2014, arXiv: 1402.6485. [Online]. Available: http://arxiv.org/abs/1402.6485

[3] M. Choudhury and K. Mohanram, "Bi-decomposition of large Boolean functions using blocking edge graphs," in *2010 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov. 2010, pp. 586–591, iSSN: 1092-3152.

[4] A. Darwiche and P. Marquis, "A Knowledge Compilation Map," *1*, vol. 17, pp. 229–264, Sep. 2002. [Online]. Available: https://jair.org/index.php/jair/article/view/10311

[5] N. Robertson and P. D. Seymour, "Graph minors. II. Algorithmic aspects of tree-width," *Journal of Algorithms*, vol. 7, no. 3, pp. 309–322, Sep. 1986. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0196677486900234