Extended Resolution Simulates Binary Decision Diagrams

Nicolas Peltier

Leibniz-IMAG - CNRS, 46, avenue Felix Viallet, 38031 Grenoble cedex, France, Nicolas.Peltier@imag.fr

Abstract

We prove that binary decision diagrams [1] can be polynomially simulated by the extended resolution rule of [2]. More precisely, for any unsatisfiable formula ϕ, there exists an extended resolution refutation of ϕ where the number of steps is polynomially bounded by the maximal size of the BDDs built from the formulae occurring in ϕ.

Key words: Extended Resolution, Binary Decision Diagrams

1 Introduction

Resolution [3] is the most widely studied approach in propositional theorem proving. Many refinements and variants of the resolution rule have been designed and their theoretical complexity and practical performances have been throughoutly investigated. The most well-known and widely used variant is the *Davis-Putnam-Logemann-Loveland* procedure [4] (although the DPLL procedure is very different at first sight from the resolution method, it can be shown that it is equivalent to tree-like resolution proof procedures). More recent, very efficient, approaches, such as the ones described in [5–7], can also be characterized as variants (refinements) of the resolution method.

Other approaches use **Binary Decision Diagrams** (or BDDs, see for example [1]). Informally speaking, a BDD can be seen as a "graphical" representation of an "if-then-else" formula: the leaves represent the possible values of the function (0 or 1), whereas each node corresponds to a choice point, with two successors depending on the value of a given propositional variable v. A key idea is that common subtrees may be *shared* instead of being duplicated, which may yield very concise representations (using directed acyclic graphs instead of trees). BDDs are built inductively, starting from the atomic subformulae and

using logical operations on BDDs ($\vee, \wedge, \ldots$). The complexity of this procedure depends on the maximal size of the BDDs corresponding to the subformulae occurring in the considered problem.

Practical experimentations have shown that these two approaches – resolution and BDDs – are incomparable. Indeed, there exist benchmarks for which BDD-based systems outperform resolution-based provers by several orders of magnitude (particularly in VLSI design [8]), whereas resolution is much better on other examples (see for instance [9]). Recently [10], it has been shown that these two techniques are also *incomparable* from a *theoretical* point of view, in the sense that none of them can *polynomially simulate* the other one. There exist sequences of formulae for which the length of the shortest resolution proof is exponential w.r.t. the size of the corresponding BDDs and conversely, there exist formulae having a short resolution proof but containing a subformula with a very complex BDD.

As stated in [10], a very natural question arises: is it possible to extend the resolution rule in such a way that it simulates BDDs ? In the present paper, we show that the resolution method, augmented by the *extension rule* originally defined by Tseitin [2], polynomially simulates BDDs in the following sense: for any unsatisfiable formula ϕ, there exists a refutation proof of ϕ in which the number of (resolution and extension) steps is *polynomially bounded* by the maximal size of the reduced BDDs corresponding to the subformulae occurring in ϕ (of course the reduced BDD of ϕ itself is 0 if ϕ is unsatisfiable). It is worth mentioning that the usual algorithm to compute the reduced OBDD of a formula is polynomially bounded by the same maximum [1].

2 Some basic notions

In this section we briefly review some basic notions and notations that are necessary for the understanding of our work.

Formulae are built as usual on a set of *propositional variables* $\mathcal{P}$, using the (complete) set of connectives $\vee, \wedge, \Leftrightarrow, \neg$. $\phi \Rightarrow \psi$ is used as a shortcut for the formula $\neg\phi \vee \psi$.

A *literal* is either a propositional variable or the negation of a propositional variable. A *clause* is a finite set of literals (interpreted as a disjunction). The empty clause is denoted by $\Box$.

For any formula ϕ, we denote by $SF(\phi)$ the set of formulae occurring in ϕ.

An *interpretation* is a subset of $\mathcal{P}$. The notions of model, satisfiability, ... are

defined as usual.

A *resolvent* of two clauses C, D is a clause of the form $(C \setminus \{a\}) \cup (D \setminus \{\neg a\})$ where a is a variable s.t. $a \in C$ and $\neg a \in D$.

If ϕ is a formula, then $|\phi|$ denotes the size of ϕ (i.e. number of symbols occurring in ϕ).

Extended Resolution

We use the same definition of the extension rule as in [11]. Informally, the idea is to extend the resolution calculus by introducing new propositional variables during proof search. These variables may be seen as new "names" for a propositional "lemma" (not necessarily occurring in the original formula). More precisely, if S is a clause set and a, b are variables occurring in S, then an *extension* of S is obtained by adding to S the following clauses:

$$\{\neg c \vee \neg a \vee \neg b, a \vee c, b \vee c\}$$

where a, b are variables and c is a new variable, not occurring in S. These three clauses express the fact that $c \Leftrightarrow (\neg a \vee \neg b)$. c may be seen as a new "name" for the formula $\neg a \vee \neg b$. By repeated applications of this rule one can generate any propositional formula.

An *extended derivation* from S is a sequence $(S_0, \ldots, S_n)$ of clause sets s.t. $S_0 = S$ and for any $i \in [1..n]$ S_i is obtained from S_{i-1} by resolution or extension. As well known a clause S is unsatisfiable iff there exists a *refutation* of S i.e. a derivation from S to a clause set containing $\Box$. Note that all the extension steps can be performed before the resolution steps (this is obviously not restrictive).

It is well known [12,13] that extended resolution is much more powerful than resolution (w.r.t. proof complexity). In particular, it polynomially simulates the most powerful known proof systems for propositional logic (see for instance [14]).

Binary Decision Diagrams

We recall some basic notions about BDDs. We only provide a short overview of the main definitions and results. The interested reader should refer for example to [1] for a more detailed presentation.

A *binary decision diagram* (BDD for short) is a dag (directed acyclic graph)

with a unique root such that each node is labeled either by a propositional variable or by the truth value 1 or 0. Any node labeled by a variable has two successors, a 0-successor and a 1-successor. Nodes labeled by 1 or 0 have no successor. A BDD is said to be *ordered* w.r.t. a total ordering $<$ among propositional variables iff for any node Δ labeled by a propositional variable v, the labels of the successors of Δ are strictly lower than v.

If Δ is a BDD then $var(\Delta)$ denotes the label of Δ, Δ^1 and Δ^0 denote the 1-successor and 0-successor of Δ respectively (or *nil* if Δ is labeled by 0 or 1). We denote by 0 and 1 two BDDs labeled by a 0 and 1 respectively (no confusion is possible). We write $\Delta \equiv \Delta'$ iff $var(\Delta) = var(\Delta')$, $\Delta^0 = \Delta'^0$ and $\Delta^1 = \Delta'^1$.

We denote by $|\Delta|$ the size of Δ (number of nodes).

A BDD can be used to denote a boolean function: If $\Delta \equiv 0$ (resp. 1) then the truth value of Δ in any interpretation I is 0 (resp. 1). Otherwise, the value of Δ in an interpretation I is identical to the value of Δ^v, where v denotes the truth value of $var(\Delta)$ in I.

The notion of models, satisfiability etc can be extended to BDD.

There exist two simplification rules useful for reducing the size of the BDDs (without affecting their semantics).

- If for a given node Δ, the 0-successor and 1-successor of Δ are identical, then Δ is useless and can be removed. Any link to Δ in the BDD is replaced by a link to Δ^1 $(= \Delta^0)$. This simplification rule is called "elimination". We write $\Delta \rightarrow_{elim} \Delta'$ if Δ' is obtained from Δ by an elimination step.
- If the BDD contains two nodes Δ, Δ' s.t. $\Delta \equiv \Delta'$ then obviously the boolean functions corresponding to Δ and Δ' are identical hence one of the nodes is deleted and replaced by the other one. This simplification rule is called "merging". We write $\Delta \rightarrow_{merge} \Delta'$ if Δ' is obtained from Δ by a merging step.

Any BDD irreducible w.r.t. the two above simplification rules is said to be "reduced". It can be shown that reduced ordered BDDs are unique (up to a renaming of the nodes).

If particular, if ϕ is a formula, there exists a unique reduced ordered BDD $bdd(\phi)$ equivalent to ϕ.

Given a BDD Δ one can compute a BDD $\Delta' = \neg\Delta$ s.t. the truth value of Δ' is the negation of the truth value of Δ. Similarly, given two BDDs Δ_1 and Δ_2 one can compute a BDD $\Delta = \Delta_1 \vee \Delta_2$ (resp. $\Delta_1 \wedge \Delta_2$, $\Delta_1 \Leftrightarrow \Delta_2$) s.t. the truth value of Δ is the disjunction (resp. conjunction, equivalence) of the

truth values of Δ_1 and Δ_2. More precisely, we have (see [1] for details):

- $\neg\Delta \stackrel{\text{def}}{=} 1$ if $\Delta = 0$.
- $\neg\Delta \stackrel{\text{def}}{=} 0$ if $\Delta = 1$.
- If $var(\Delta) = v$ then $var(\neg\Delta) \stackrel{\text{def}}{=} v$, $(\neg\Delta)^1 \stackrel{\text{def}}{=} \neg\Delta^1$ and $(\neg\Delta)^0 \stackrel{\text{def}}{=} \neg\Delta^0$.
- $\Delta_1 \vee \Delta_2 \stackrel{\text{def}}{=} \Delta_2$ if $\Delta_1 \equiv 0$;
- $\Delta_1 \vee \Delta_2 \stackrel{\text{def}}{=} \Delta_1$ if $\Delta_2 \equiv 0$;
- $\Delta_1 \vee \Delta_2 \stackrel{\text{def}}{=} 1$ if $\Delta_i \equiv 1$ for some $i = 1, 2$;
- $\Delta_1 \wedge \Delta_2 \stackrel{\text{def}}{=} 0$ if $\Delta_i \equiv 0$ for some $i = 1, 2$;
- $\Delta_1 \wedge \Delta_2 \stackrel{\text{def}}{=} \Delta_2$ if $\Delta_1 \equiv 1$;
- $\Delta_1 \wedge \Delta_2 \stackrel{\text{def}}{=} \Delta_1$ if $\Delta_2 \equiv 1$;
- $\Delta_1 \Leftrightarrow \Delta_2 \stackrel{\text{def}}{=} \Delta_2$ if $\Delta_1 \equiv 1$.
- $\Delta_1 \Leftrightarrow \Delta_2 \stackrel{\text{def}}{=} \Delta_1$ if $\Delta_2 \equiv 1$.
- $\Delta_1 \Leftrightarrow \Delta_2 \stackrel{\text{def}}{=} \neg\Delta_2$ if $\Delta_1 \equiv 0$.
- $\Delta_1 \Leftrightarrow \Delta_2 \stackrel{\text{def}}{=} \neg\Delta_1$ if $\Delta_2 \equiv 0$.
- If $var(\Delta_1) = var(\Delta_2) = v$ then $var(\Delta_1 \star \Delta_2) \stackrel{\text{def}}{=} v$, $(\Delta_1 \star \Delta_2)^1 \stackrel{\text{def}}{=} {\Delta_1}^1 \star {\Delta_2}^1$ and$(\Delta_1 \star \Delta_2)^0 \stackrel{\text{def}}{=} {\Delta_1}^0 \star {\Delta_2}^0$ (where $\star = \vee, \wedge, \Leftrightarrow$).
- If $var(\Delta_1) > var(\Delta_2)$ then: $var(\Delta_1 \star \Delta_2) = var(\Delta_1)$, $(\Delta_1 \star \Delta_2)^1 \stackrel{\text{def}}{=} {\Delta_1}^1 \star \Delta_2$ and $(\Delta_1 \star \Delta_2)^0 \stackrel{\text{def}}{=} {\Delta_1}^0 \star \Delta_2$ (where $\star = \vee, \wedge, \Leftrightarrow$).
- If $var(\Delta_1) < var(\Delta_2)$ then: $var(\Delta_1 \star \Delta_2) = var(\Delta_2)$, $(\Delta_1 \star \Delta_2)^1 \stackrel{\text{def}}{=} \Delta_1 \star {\Delta_2}^1$ and $(\Delta_1 \star \Delta_2)^0 \stackrel{\text{def}}{=} \Delta_1 \star {\Delta_2}^0$ (where $\star = \vee, \wedge, \Leftrightarrow$).

Note that the obtained BDDs are not reduced in general. They have to be reduced afterwards using the merging and elimination rules. For instance if Δ_1, Δ_2 are the BDD corresponding to p and $\neg p$ respectively, then we have $var(\Delta_1 \wedge \Delta_2) = p$, $(\Delta_1 \wedge \Delta_2)^1 = (\Delta_1 \wedge \Delta_2)^0 = 0$, hence $\Delta_1 \wedge \Delta_2$ can be reduced to 0 by elimination.

The two properties stated in the proposition below will be useful in the following.

Proposition 1 *Let Δ_1, Δ_2 be two BDDs. Let $\star = \vee, \wedge, \Leftrightarrow$.*

(1) The size of $\Delta_1 \star \Delta_2$ is at most $|\Delta_1| \times |\Delta_2|$.
(2) The number of merging and elimination steps required to compute the reduced ordered BDD corresponding to the non-reduced OBDD $\Delta_1 \star \Delta_2$ is bounded by $|\Delta_1 \star \Delta_2|$.

Proof:

(1) By definition, any BDD occurring in $\Delta_1 \star \Delta_2$ is of the form $\Delta'_1 \star \Delta'_2$, where Δ'_1, Δ'_2 occur respectively in Δ_1 and Δ_2. Thus the size of $\Delta_1 \star \Delta_2$ is bounded by $|\Delta_1| \times |\Delta_2|$.
(2) This is immediate since the elimination and merging rules strictly decrease the number of nodes.

□

Transformation into clausal form

The first problem that we have to solve for simulating BDDs is the transformation into clausal form. It is well-known that the standard clausification algorithm is exponential. We use the (structure-preserving) clausal transformation algorithm firstly introduced in [2], based on a renaming of the subformulae. Of course, there exist many useful refinements of this algorithm (see for instance [15,16]) but we prefer to use a simpler version, adapted in order to better suit our purposes.

We introduce a function Cl mapping each formula ϕ into a sat-equivalent set of clauses $Cl(\phi)$. Let $\psi \rightarrow p_\psi$ be a function mapping the formulae in $SF(\phi) \cup \{\neg\psi \mid \psi \in SF(\phi)\} \cup \{\psi \Rightarrow \psi', \psi' \Rightarrow \psi, \neg(\psi \Rightarrow \psi'), \neg(\psi' \Rightarrow \psi) \mid (\psi \Leftrightarrow \psi') \in SF(\phi)\}$ to pairwise distinct propositional variables not occurring in ϕ.

First, we define inductively the following function mapping each formula ϕ to a clause set $Def(\phi)$ defining the predicate symbols p_ϕ and $p_{\neg\phi}$ corresponding to ϕ and $\neg\phi$ (only the implication $p_\phi \Rightarrow \phi$ is needed).

If p is a propositional variable then $Def(p) \stackrel{\text{def}}{=} \{\neg p_{\neg p} \vee \neg p, \neg p_p \vee p\}$.

Otherwise:

$$
\begin{aligned}
Def(\phi \vee \psi) \stackrel{\text{def}}{=} & \{\neg p_{\phi\vee\psi} \vee p_\phi \vee p_\psi, \neg p_{\neg(\phi\vee\psi)} \vee p_{\neg\phi}, \neg p_{\neg(\phi\vee\psi)} \vee p_{\neg\psi}\} \\
& \cup Def(\phi) \cup Def(\psi). \\
Def(\phi \wedge \psi) \stackrel{\text{def}}{=} & \{\neg p_{\phi\wedge\psi} \vee p_\phi, \neg p_{\phi\wedge\psi} \vee p_\psi, \neg p_{\neg(\phi\wedge\psi)} \vee p_{\neg\phi} \vee p_{\neg\psi}\} \\
& \cup Def(\phi) \cup Def(\psi). \\
Def(\phi \Leftrightarrow \psi) \stackrel{\text{def}}{=} & \{\neg p_{\phi\Leftrightarrow\psi} \vee p_{\phi\Rightarrow\psi}, \neg p_{\phi\Leftrightarrow\psi} \vee p_{\psi\Rightarrow\phi}, \\
& \neg p_{\neg(\phi\Leftrightarrow\psi)} \vee p_{\neg(\phi\Rightarrow\psi)} \vee p_{\neg(\psi\Rightarrow\phi)}, \\
& \neg p_{\phi\Rightarrow\psi} \vee p_{\neg\phi} \vee p_\psi, \neg p_{\neg(\phi\Rightarrow\psi)} \vee p_\phi, \neg p_{\neg(\phi\Rightarrow\psi)} \vee p_{\neg\psi}, \\
& \neg p_{\psi\Rightarrow\phi} \vee p_{\neg\psi} \vee p_\phi, \neg p_{\neg(\psi\Rightarrow\phi)} \vee p_\psi, \neg p_{\neg(\psi\Rightarrow\phi)} \vee p_{\neg\phi}, \\
& \} \\
& \cup Def(\phi) \cup Def(\psi).
\end{aligned}
$$

$$
Def(\neg\phi) \stackrel{\text{def}}{=} \{\neg p_{\neg\neg\phi} \vee p_\phi\} \cup Def(\phi).
$$

Intuitively, $Def(\phi)$ expresses the fact that for any subformula ψ occurring in

ϕ, we have: $p_\psi \Rightarrow \psi$ and $p_{\neg\psi} \Rightarrow \neg\psi$. Then we define:

$$Cl(\phi) \stackrel{\text{def}}{=} Def(\phi) \cup \{p_\phi\}.$$

Lemma 1 *Let ϕ be a formula. ϕ is satisfiable iff $Cl(\phi)$ is satisfiable. Moreover the size of $Cl(\phi)$ is polynomial w.r.t. the size of ϕ.*

3 Simulating Binary Decision Diagrams by Resolution

We need to introduce some further notations and definitions, essentially useful for improving the readability of the forthcoming proofs.

If C is a clause and S a set of clauses, we denote by $C \vee S$ the set of clauses $\{C \vee D \mid D \in S\}$. Obviously, $C \vee S$ contains exactly $|S|$ clauses (where $|S|$ denotes the number of clauses in S).

A *BDD-naming* is a *partial* function γ mapping BDDs to clause sets, s.t. for any $\Delta \in dom(\gamma)$:

- either $\gamma(\Delta) = \{\Box\}$ and $\Delta \equiv 0$;
- or $\gamma(\Delta) = \emptyset$ and $\Delta \equiv 1$;
- or $\gamma(\Delta) = \{\{v\}\}$, where v is a propositional variable (we have possibly $\Delta \equiv 0, 1$).

We denote by $named(\gamma)$ the set of BDDs Δ s.t. $\gamma(\Delta) = \{\{v\}\}$. In this case the variable v is denoted by $l_\gamma(\Delta)$. v can be seen as a "name" associated to the BDD Δ.

Note that the propositional variable v occurring in the BDD-naming is not related to the variable labeling Δ. v can be seen as a "name" given to the BDD Δ.

Let γ be a BDD-naming. We write $S \lhd_\gamma \Delta$ iff:

- Either $\Delta \equiv 1$ and $S = \emptyset$;
- Or $\Delta \equiv 0$ and $S = \{\Box\}$;
- Or $\Delta^1, \Delta^0 \in dom(\gamma)$, $S = (\neg a \vee \gamma(\Delta^1)) \cup (a \vee \gamma(\Delta^0))$, where $a = var(\Delta)$.

Note that $\gamma(\Delta^1)$ and $\gamma(\Delta^0)$ are clause sets. Thus S contains $2, 1$ or 0 clause(s) depending on $\Delta, \gamma(\Delta^1)$ and $\gamma(\Delta^0)$.

If for any BDD Γ occurring in $named(\gamma)$, $l_\gamma(\Gamma)$ is equivalent to Δ and if $S \lhd_\gamma \Delta$ then it is clear that S and Δ must be equivalent (i.e. must have the same truth value in any interpretation).

A clause set S *encodes* a BDD Δ w.r.t. a BDD-naming γ iff the following conditions hold:

- $\Delta \in \mathit{named}(\gamma)$;
- For any $\Gamma \in \mathit{named}(\gamma)$, there exists a clause set $S(\Gamma) \lhd_\gamma \Gamma$ s.t. the clause set $\neg l_\gamma(\Gamma) \vee S(\Gamma)$ is a subset of S.

From this definition, we see that a clause set S encodes Δ w.r.t. γ iff it satisfies the following properties.

- If Δ is labeled by a propositional variable a and the 1-successor of Δ is Δ', then S must contain the clause $\neg p \vee \neg a \vee q$, where p, q are the names of Δ, Δ' respectively (i.e. $p = l_\gamma(\Delta)$, $q = l_\gamma(\Delta')$).
- If Δ is labeled by a propositional variable a and the 0-successor of Δ is Δ', then S must contain the clause $\neg p \vee a \vee q$, where p, q are the names of Δ, Δ'.
- If Δ is labeled by a propositional variable a and the 1-successor of Δ is 0, then S must contain the clause $\neg p \vee \neg a$, where p is the name of Δ.
- If Δ is labeled by a propositional variable a and the 0-successor of Δ is 0, then S must contain the clause $\neg p \vee a$, where p is the name of Δ.

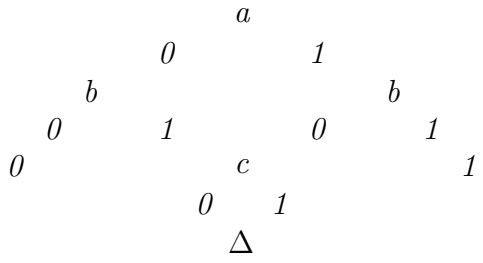
Note that S may contain other clauses that those specified above. In particular, if $S' \supseteq S$ and S encodes Δ w.r.t. γ, then S' also encodes Δ w.r.t. γ.

It follows from these definitions that if S encodes Δ then for any BDD Γ occurring in Δ, $\Gamma \in \mathit{dom}(\gamma)$. Moreover, for any $\Gamma \in \mathit{named}(\gamma)$, either $I \not\models \neg l_\gamma(\Gamma)$ or Γ and $S(\Gamma)$ have the same truth value in I. In particular, if $\Gamma \equiv 0$ then either $\gamma(\Gamma) = \Box$ or $\neg l_\gamma(\Gamma)$ must occur in S.

This definition allows one to encode BDDs into clause sets. Now, we show that we can simulate all the standard operations on BDDs, i.e. elimination, merging, conjunction and disjunction, in polynomial time, using only the resolution and extension rules.

First we show that one can simulate the *elimination* rule. The idea is very simple: it suffices to apply the resolution rule on the variable corresponding to the eliminated node. Before giving the technical details, we provide a simple illustrating example.

Example 1 *Let us consider the following BDD.*



This BDD is encoded by the following clause set:

1 $\neg p \vee a \vee q_1$

2 $\neg p \vee \neg a \vee q_2$

3 $\neg q_1 \vee b$

4 $\neg q_1 \vee \neg b \vee r$

5 $\neg q_2 \vee b \vee r$

6 $\neg r \vee c \vee s$

7 $\neg r \vee \neg c \vee s$

where $s = l_\gamma(\Delta)$.

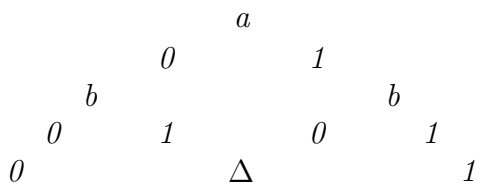
By resolving the clauses 6 *and* 7 *we obtain:* 8 $\neg r \vee s$

By resolving the clause 4 *and* 5 *with clause* 8 *we eliminate all the occurrences of* r *and replace them by* s.

9 $\neg q_1 \vee \neg b \vee s$

10 $\neg q_2 \vee b \vee s$

The reader can check that the clause set $\{1, 2, 3, 9, 10\}$ *encodes the following BDD.*



This BDD is obviously obtained from the initial one by applying the elimination rule.

Lemma 2 *Let Δ be a BDD. Let S be a clause set encoding Δ w.r.t. a BDD-naming γ. Let Δ' be a BDD s.t. $\Delta \rightarrow_{elim} \Delta'$.*

One can generate from S in at most $2 \times |\Delta'| + 1$ resolution steps a clause set S' s.t. S' encodes Δ' w.r.t. a BDD-naming γ' s.t. $\gamma(\Delta) = \gamma'(\Delta')$.

Proof: Since $\Delta \rightarrow_{elim} \Delta'$, there exists a BDD Γ in Δ s.t. $\Gamma^1 = \Gamma^0$ and $\Gamma \not\equiv 1, 0$. Since S encodes Δ w.r.t. γ, $\Gamma \in dom(\gamma)$. Since $\Gamma \not\equiv 1, 0$ this implies that $\Gamma \in named(\gamma)$. Since S encodes Δ w.r.t. γ this implies that S contains the clause sets $\neg l_\gamma(\Gamma) \vee \neg p \vee \gamma(\Gamma^1)$ and $\neg l_\gamma(\Gamma) \vee p \vee \gamma(\Gamma^0)$ where $p = var(\Gamma)$. We have $\Gamma^0 = \Gamma^1$. Thus by resolution on p we get the clause $\neg l_\gamma(\Gamma) \vee \gamma(\Gamma^1)$ (denoted by $(\star)$ in the following).

Now, let Ω be a BDD occurring in Δ'. By definition, Ω is obtained from a (unique) BDD $anc(\Omega)$ occurring in Δ by replacing at most one occurrence of the BDD Γ by Γ^1 $(= \Gamma^0)$. We define the following BDD-naming: $\gamma'(\Omega) \stackrel{\text{def}}{=} \gamma(anc(\Omega))$, for any Ω occurring in Δ'.

Clearly, $\Delta' \in named(\gamma')$. Now, we prove, by induction on the size of the BDD, that for any Ω occurring in Δ' s.t. $\Omega \in named(\gamma')$, one can construct a clause set $\neg l_{\gamma'}(\Omega) \vee S'$ s.t. $S' \lhd_{\gamma'} \Omega$.

Obviously $anc(\Omega)$ occurs in Δ. If $anc(\Omega) \notin named(\gamma)$ then by definition $\Omega \notin named(\gamma')$ which is impossible. Thus $anc(\Omega) \in named(\gamma)$ whence S contains a clause set $\neg l_\gamma(anc(\Omega)) \vee S''$ s.t. $S'' \lhd_\gamma anc(\Omega)$. If $anc(\Omega) \equiv 1$ then $\Omega \equiv 1$ hence the proof is completed (it suffices to take $S' = \emptyset$).

If $anc(\Omega) \equiv 0$ then $S'' = \Box$ and $\Omega \equiv 0$. Thus $S'' \lhd_\gamma \Omega$. Moreover $\gamma'(\Omega) = \gamma(anc(\Omega))$. Thus S contains the clause $\neg l_\gamma(\Omega) \vee \Box$ where $\Box \lhd_{\gamma'} \Omega$ and the proof is completed.

Otherwise, let $p = var(anc(\Omega))$. By definition $S'' = \neg p \vee \gamma(anc(\Omega)^1) \cup p \vee \gamma(anc(\Omega)^0)$.

If $anc(\Omega)^1 \neq \Gamma$, then $anc(\Omega)^1 = anc(\Omega^1)$. In this case $\gamma(anc(\Omega)^1) = \gamma(anc(\Omega^1)) = \gamma'(\Omega^1)$. Hence S contains the clause set $\neg l_\gamma(anc(\Omega)) \vee \neg p \vee \gamma'(\Omega^1)$.

If $anc(\Omega)^1 = \Gamma$ then S contains $\neg l_\gamma(anc(\Omega)) \vee \neg p \vee \gamma(\Gamma)$. Moreover $\gamma(\Gamma) = l_\gamma(\Gamma)$. By applying the resolution rule between this clause set and the clause $\neg l_\gamma(\Gamma) \vee \gamma(\Gamma^1)$ constructed before (see $(\star)$) we get: $\neg l_\gamma(anc(\Omega)) \vee \neg p \vee \gamma(\Gamma^1)$, i.e. $\neg l_{\gamma'}(\Omega) \vee \neg p \vee \gamma'(\Gamma^1)$.

Thus in all the cases $\neg l_{\gamma'}(\Omega) \vee \neg p \vee \gamma'(\Gamma^1)$ can be generated. Similarly we

obtain the clause set $\neg l_{\gamma'}(\Omega) \vee p \vee \gamma'(\Gamma^0)$. Hence we have obtained a clause set $\neg l_{\gamma'}(\Omega) \vee S'$ s.t. $S'' \lhd_{\gamma'} \Omega$, which completes the proof.

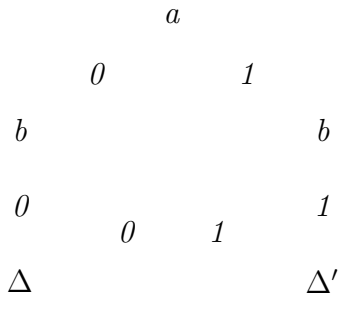
One resolution steps is needed for generating $\neg l_\gamma(\Gamma) \vee \gamma(\Gamma^1)$, then at most two resolution steps is needed for each symbol in $dom(\Delta')$. Thus the number of resolution steps is bounded by $2 \times |\Delta'| + 1$ (the extension rule is not needed). □

Now we show the *merging* rule can be simulated.

Again, before proving the general result, we consider a simple example.

Example 2 *We consider a BDD encoded by the following clause set.*

1	$\neg p \vee a \vee q_1$
2	$\neg p \vee \neg a \vee q_2$
3	$\neg q_1 \vee b \vee r_1$
4	$\neg q_1 \vee \neg b \vee r_2$
5	$\neg q_2 \vee b \vee r_1$
6	$\neg q_2 \vee \neg b \vee r_2$



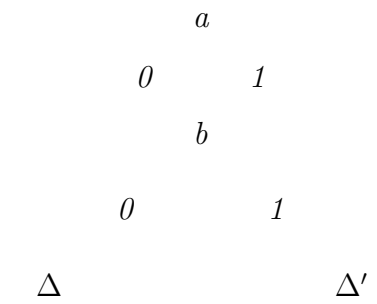
At this point, we apply the extension rule in order to introduce a new variable q s.t. $q \Leftrightarrow (q_1 \vee q_2)$. This yields the clauses

7	$q \vee \neg q_1$
8	$q \vee \neg q_2$
9	$\neg q \vee q_1 \vee q_2$

Hence by resolution:

10	$\neg p \vee a \vee q$	*(res 1,7)*
11	$\neg p \vee \neg a \vee q$	*(res 2,8)*
12	$\neg q \vee b \vee r_1$	*(res 9,3,5)*
13	$\neg q \vee \neg b \vee r_2$	*(res, 9,4,6)*

The obtained clause set encodes the following BDD (obtained by applying the merging rule on the initial one):



Lemma 3 *Let Δ be a BDD. Let S be a clause set encoding Δ w.r.t. a BDD-naming γ. Let Δ' be a BDD s.t. $\Delta \rightarrow_{merge} \Delta'$.*

One can construct in at most $10 + 2 \times |\Delta'|$ resolution or extension steps a clause set S' s.t. S' encodes Δ' w.r.t. a BDD-naming γ' s.t. $\gamma(\Delta) = \gamma'(\Delta')$.

Proof: By definition, Δ contains two equivalent BDDs Γ_1 and Γ_2 and any BDD Ω in Δ' is obtained from a BDD $anc(\Omega)$ in Δ by replacing the BDD Γ_2 by Γ_1.

First we show how to define the BDD-naming γ. We distinguish two cases.

- $\Gamma_i \in named(\gamma)$, for any $i = 1, 2$.
  By definition, S contains two clause sets of the form $\neg l_\gamma(\Gamma_i) \vee S_i$ $(i = 1, 2)$ where $S_i \lhd_\gamma \Gamma_i$. Since Γ_1 and Γ_2 are equivalent, it is clear that we must have $S_1 = S_2$.
  Let $i = 1, 2$. We apply the extension rule on the variable $l_\gamma(\Gamma_i)$ (the rule is applied with $a = b = l_\gamma(\Gamma_i)$). This introduces a new variable p_i (not occurring in S) and the following clauses: $p_i \vee l_\gamma(\Gamma_i), \neg p_i \vee \neg l_\gamma(\Gamma_i)$.
  Then we apply the extension rule on the variables p_1 and p_2. This introduces a new variable p (not occurring in S) and the following clauses: $p_i \vee p$ $(i = 1, 2)$, $\neg p \vee \neg p_1 \vee \neg p_2$. By resolving these last clauses with the clauses $p_i \vee l_\gamma(\Gamma_i)$ and $\neg p_i \vee \neg l_\gamma(\Gamma_i)$, we get $\neg l_\gamma(\Gamma_i) \vee p$, $\neg p \vee l_\gamma(\Gamma_1) \vee l_\gamma(\Gamma_2)$.
  By resolving the last clause with $\neg l_\gamma(\Gamma_i) \vee S_i$ $(i = 1, 2)$, we get $\neg p \vee S_1$ (since $S_1 = S_2$). This takes (exactly) 2 extension steps and (at most) 8 resolution steps (since $|S_1| \leq 2$).
  We define a BDD-naming γ' as follows. $\gamma'(\Omega) \stackrel{\text{def}}{=} \gamma(anc(\Omega))$ if $\Omega \neq \Gamma_i$ and $\gamma'(\Gamma_1) \stackrel{\text{def}}{=} p$.
- $\Gamma_i \notin named(\gamma)$ for some $i = 1, 2$. In this case we have either $\Gamma_i \equiv 1$ or $\Gamma_i \equiv 0$. In both cases we define $\gamma'(\Omega) \stackrel{\text{def}}{=} \gamma(anc(\Omega))$ for any $\Omega \neq \Gamma_i$ and $\gamma(\Gamma_1) = \gamma(\Gamma_i)$.

We know that $\Delta' \in named(\gamma')$. Now, we prove that for any BDD Ω in Δ' s.t. $\Omega \in named(\gamma')$ one can generate a clause set $\neg\gamma'(\Omega) \vee S'$ s.t. $S' \lhd_{\gamma'} \Omega$.

By definition S contains a clause set $\neg l_\gamma(anc(\Omega)) \vee S''$ where $S'' \lhd_\gamma anc(\Omega)$.

If $\Omega = \Gamma_1$ then the proof is obvious, in this case (since $\Omega \in named(\gamma')$) we must have $\Gamma_1, \Gamma_2 \in named(\gamma)$ hence we have generated a clause set $\neg p \vee S_1$. Otherwise, we have $anc(\Omega) \neq \Gamma_i$ thus $\gamma(anc(\Omega)) = \gamma'(\Omega)$.

If $anc(\Omega) \equiv 1$ then $\Omega \equiv 1$ hence the proof is immediate (it suffices to take $S' = \emptyset$).

If $anc(\Omega) = 0$ then $S'' = \Box$. Moreover $\Omega \equiv 0$ hence $S'' \lhd_{\gamma'} \Omega$ and the proof is completed.

Otherwise, let $q = var(anc(\Omega))$. S'' contains a clause $\neg q \vee \gamma(anc(\Omega)^1)$. If $anc(\Omega)^1 \neq \Gamma_i$, then $anc(\Omega^1) = anc(\Omega)^1$. Thus S'' contains the clause $\neg q \vee \gamma(anc(\Omega^1)) = \neg q \vee \gamma'(\Omega^1)$. Therefore the clause set $l_\gamma(anc(\Omega)) \vee \neg q \vee \gamma'(\Omega^1) = l_{\gamma'}(\Omega) \vee \neg q \vee \gamma'(\Omega^1)$ can be generated.

If $anc(\Omega)^1 = \Gamma_i$, then $\Omega^1 = \Gamma_1$. We distinguish two cases.

If there exists j s.t. $\Gamma_j \notin named(\gamma)$, then by definition we have $\Gamma_1 \equiv \Gamma_2 \equiv 1$ or $\Gamma_1 \equiv \Gamma_2 \equiv 0$ and $\gamma'(\Omega^1) = \gamma(\Gamma_j)$. If $\Gamma_i \notin named(\gamma)$ then $\gamma(\Gamma_i) = \gamma(\Gamma_j) = \gamma'(\Omega^1)$ hence $\neg l_\gamma(anc(\Omega)) \vee S''$ contains the clause $\neg l_\gamma(anc(\Omega)) \vee \neg q \vee \gamma'(\Omega^1)$. If $\Gamma_i \in named(\gamma)$ then by definition S contains a clause $\neg l_\gamma(\Gamma_i) \vee S'''$ where $S''' \lhd_\gamma \Gamma_i$. We have $S''' = \{\{\gamma'(\Omega^1)\}\}$. Thus by resolution from the clauses $\neg l_\gamma(anc(\Omega)) \vee \neg q \vee l_\gamma(anc(\Omega)^1)$ and $\neg l_\gamma(\Gamma_i) \vee S'''$ we get $\neg l_\gamma(anc(\Omega)) \vee \neg q \vee \gamma'(\Omega^1)$.

If $\Gamma_j \in named(\gamma)$ for any $j = 1, 2$, then by applying the resolution rule between $\neg l_{\gamma'}(\Omega) \vee \neg q \vee \gamma(anc(\Omega)^1)$ and the clause $\neg l_\gamma(\Gamma_i) \vee p$ generated above, we get $\neg l_{\gamma'}(\Omega) \vee \neg q \vee p$ i.e. $\neg l_{\gamma'}(\Omega) \vee \neg q \vee \gamma'(\Gamma_1) = \neg l_{\gamma'}(\Omega) \vee \neg q \vee \gamma'(\Omega^1)$.

Thus in all the cases the clause set $\neg l_{\gamma'}(\Omega) \vee \neg q \vee \gamma'(\Omega^1)$ can be generated. By symmetry we can also generate the clause set $\neg l_{\gamma'}(\Omega) \vee q \vee \gamma'(\Omega^0)$.

This takes at most 2 resolution steps for each BDD Ω in Δ'.

$\Box$

Lemmata 4 and 5 show that one can compute the disjunction and conjunction (respectively) of two BDDs. The idea is – given two BDDs Δ_1, Δ_2 encoded by a clause set w.r.t. a BDD-naming γ – to use the extension rule to generate the formula $l_\gamma(\Delta_1) \vee l_\gamma(\Delta_2)$ (resp. $l_\gamma(\Delta_1) \wedge l_\gamma(\Delta_2)$). Afterwards, the resolution rule is used to generate the clauses encoding $\Delta_1 \vee \Delta_2$ and $\Delta_1 \wedge \Delta_2$.

Lemma 4 *Let S be a clause set. Assume that S encodes two BDDs Δ_1 and Δ_2 w.r.t. γ. If S contains a clause of the form $\neg u \vee \gamma(\Delta_1) \vee \gamma(\Delta_2)$, then one can generate a clause set S' in at most $14 \times |\Delta_1| \times |\Delta_2|$ resolution or extension*

steps s.t. S' encodes $\Delta_1 \vee \Delta_2$ w.r.t. a BDD-naming γ' s.t. $l_{\gamma'}(\Delta_1 \vee \Delta_2) = u$.

Proof: For each pair $(\Gamma_1, \Gamma_2) \neq (\Delta_1, \Delta_2)$ s.t. Γ_i occurs in Δ_i and $\Gamma_i \in \mathit{named}(\Delta_i)$, we apply the extension rule with $a = b = l_\gamma(\Gamma_i)$. This yields the clauses $p_i(\Gamma_1, \Gamma_2) \vee l_\gamma(\Gamma_i)$ and $\neg p_i(\Gamma_1, \Gamma_2) \vee \neg l_\gamma(\Gamma_i)$ (for $i = 1, 2$) where $p_i(\Gamma_1, \Gamma_2)$ is a new propositional variable. Then by applying again the extension rule on the variables $p_1(\Gamma_1, \Gamma_2), p_2(\Gamma_1, \Gamma_2)$ we obtain $q(\Gamma_1, \Gamma_2) \vee p_i(\Gamma_1, \Gamma_2)$ and $\neg q(\Gamma_1, \Gamma_2) \vee \neg p_1(\Gamma_1, \Gamma_2) \vee \neg p_2(\Gamma_1, \Gamma_2)$. By resolution we get: $q(\Gamma_1, \Gamma_2) \vee \neg l_\gamma(\Gamma_i)$ $(i = 1, 2)$ and $\neg q(\Gamma_1, \Gamma_2) \vee l_\gamma(\Gamma_1) \vee l_\gamma(\Gamma_2)$ i.e. $\neg q(\Gamma_1, \Gamma_2) \vee \gamma(\Gamma_1) \vee \gamma(\Gamma_2)$.

This takes 2 extension steps and 4 resolution step for each pair (Γ_1, Γ_2), hence $6 \times |\Delta_1| \times |\Delta_2|$ steps.

Any BDD occurring in $\Delta_1 \vee \Delta_2$ is of the form $\Gamma_1 \vee \Gamma_2$ where Γ_i occurs in Δ_i (with possibly $\Gamma_i \equiv 1, 0$). We define γ' as the extension of γ to the BDD of the form $\Gamma_1 \vee \Gamma_2$ where Γ_i occurs in Δ_i satisfying the following property: If $\Gamma_1, \Gamma_2 \in \mathit{named}(\gamma)$ then $\gamma'(\Gamma_1 \vee \Gamma_2) \stackrel{\text{def}}{=} \{\{q(\Gamma_1, \Gamma_2)\}\}$ if $(\Gamma_1, \Gamma_2) \neq (\Delta_1, \Delta_2)$ and $\gamma'(\Delta_1 \vee \Delta_2) \stackrel{\text{def}}{=} \{\{u\}\}$.

Note that γ' is well defined since $\Gamma_1 \vee \Gamma_2 \neq \Gamma'_1 \vee \Gamma'_2$ if $\Gamma_1 \neq \Gamma'_1$ or $\Gamma_2 \neq \Gamma'_2$.

By definition, for any Γ_1, Γ_2 occurring in Δ_1, Δ_2 respectively, if $\Gamma_i \in \mathit{named}(\gamma)$ for any $i = 1, 2$ then the clause $\neg l_{\gamma'}(\Gamma_1 \vee \Gamma_2) \vee l_\gamma(\Gamma_1) \vee l_\gamma(\Gamma_2)$ has been generated. Moreover, if $\Gamma_1, \Gamma_2 \neq \Delta_1, \Delta_2$ we have also generated two clauses of the form $l_{\gamma'}(\Gamma_1 \vee \Gamma_2) \vee \neg l_\gamma(\Delta_i)$ for any $i = 1, 2$ (this is not true for $\Gamma_1, \Gamma_2 = \Delta_1, \Delta_2$ since in this case $l_{\gamma'}(\Gamma_1 \vee \Gamma_2) = u$).

By definition, since $\Delta_1, \Delta_2 \in \mathit{named}(\gamma)$ we have $\Delta_1 \vee \Delta_2 \in \mathit{named}(\gamma')$.

Let $\Gamma \in \mathit{named}(\gamma')$. We prove that one can construct a clause set $\neg l_{\gamma'}(\Gamma) \vee S'$ s.t. $S' \lhd_{\gamma'} \Gamma$. By definition Γ is of the form $\Gamma_1 \vee \Gamma_2$ where Γ_i occurs in Δ_i $(i = 1, 2)$.

If $\Gamma_1 \equiv 1$ or $\Gamma_2 \equiv 1$ then $\Gamma_1 \vee \Gamma_2 \equiv 1$, hence the proof is immediate (it suffices to take $S' = \emptyset$).

If $\Gamma_1 \equiv 0$ then $\Gamma \stackrel{\text{def}}{=} \Gamma_2$. Since $\Gamma \in \mathit{named}(\gamma')$ we must have $\Gamma_2 \in \mathit{named}(\gamma)$. By definition S contains a clause set $\neg l_\gamma(\Gamma_2) \vee S_2$ where $S_2 \lhd_\gamma \Gamma_2$. Moreover $\gamma(\Gamma) = \gamma'(\Gamma)$. The same holds if $\Gamma_2 \equiv 0$.

Now assume that $\Gamma_i \not\equiv 0, 1$. Then $\Gamma_i \in \mathit{named}(\gamma)$ hence S contains two clause sets $\neg l_\gamma(\Gamma_i) \vee S_i$ where $S_i \lhd_\gamma \Gamma_i$. Let $r_i = \mathit{var}(\Gamma_i)$. We assume, w.l.o.g. that $r_1 \geq r_2$ (the other case is symmetric). By definition, $\mathit{var}(\Gamma_1 \vee \Gamma_2) = r_1$. We have $S_i = (\neg r_i \vee \gamma({\Gamma_i}^1)) \cup (r_i \vee \gamma({\Gamma_i}^0))$.

We distinguish two cases.

- If $r_1 = r_2 = r$, then $(\Gamma_1 \vee \Gamma_2)^1 = {\Gamma_1}^1 \vee {\Gamma_2}^1$.

  By applying the resolution rule between $\neg l_\gamma(\Gamma_i) \vee S_i$ and the clause $\neg q(\Gamma_1 \vee \Gamma_2) \vee l_\gamma(\Gamma_1) \vee l_\gamma(\Gamma_2)$ generated above, we obtain $\neg q(\Gamma_1 \vee \Gamma_2) \vee \neg r \vee \gamma({\Gamma_1}^1) \vee \gamma({\Gamma_2}^1)$ i.e. $\neg l_{\gamma'}(\Gamma_1 \vee \Gamma_2) \vee \neg r \vee \gamma({\Gamma_1}^1) \vee \gamma({\Gamma_2}^1)$.

  Assume that ${\Gamma_1}^1$ and ${\Gamma_2}^1$ occur in $named(\gamma)$. Then, since ${\Gamma_i}^1 \neq \Delta_i$ we have generated the clauses $l_{\gamma'}({\Gamma_1}^1 \vee {\Gamma_2}^1) \vee \neg l_\gamma({\Gamma_i}^1)$ $(i = 1, 2)$ thus after two resolution steps we obtain: $\neg l_{\gamma'}(\Gamma_1 \vee \Gamma_2) \vee \neg r \vee \gamma'({\Gamma_1}^1 \vee {\Gamma_2}^1)$ i.e. $\neg l_{\gamma'}(\Gamma_1 \vee \Gamma_2) \vee \neg r \vee \gamma'((\Gamma_1 \vee \Gamma_2)^1)$.

  If ${\Gamma_1}^1 \notin named(\gamma)$ then we have either ${\Gamma_1}^1 \equiv 0$ or ${\Gamma_1}^1 \equiv 1$. If ${\Gamma_1}^1 \equiv 1$ then $(\Gamma_1 \vee \Gamma_2)^1 \equiv 1$ hence $\gamma'(\Gamma_1 \vee \Gamma_2) = \emptyset$. Thus the clause set $\neg l_{\gamma'}(\Gamma_1 \vee \Gamma_2) \vee \neg r \vee \gamma'((\Gamma_1 \vee \Gamma_2)^1)$ is empty. If ${\Gamma_1}^1 \equiv 0$, then $\gamma'((\Gamma_1 \vee \Gamma_2)^1) \stackrel{\text{def}}{=} \gamma({\Gamma_2}^1)$. Moreover $\gamma(\Gamma_1) = \Box$ thus the clause $\neg l_{\gamma'}(\Gamma_1 \vee \Gamma_2) \vee \neg r \vee \gamma({\Gamma_1}^1) \vee \gamma({\Gamma_2}^1)$ is equivalent to $\neg l_{\gamma'}(\Gamma_1 \vee \Gamma_2) \vee \neg r \vee \gamma'((\Gamma_1 \vee \Gamma_2)^1)$.

  The same holds if ${\Gamma_2}^1 \notin named(\gamma)$. Thus in each case, the clause set $\neg l_{\gamma'}(\Gamma_1 \vee \Gamma_2) \vee \neg r \vee \gamma'((\Gamma_1 \vee \Gamma_2)^1)$ can be generated.
- Now assume that $r_1 > r_2$. In this case $(\Gamma_1 \vee \Gamma_2)^1 = {\Gamma_1}^1 \vee \Gamma_2$. By applying the resolution rule between $\neg l_\gamma(\Gamma_1) \vee S_1$, and $\neg l_{\gamma'}(\Gamma_1 \vee \Gamma_2) \vee \gamma(\Gamma_1) \vee \gamma(\Gamma_2)$ we obtain $\neg l_{\gamma'}(\Gamma_1 \vee \Gamma_2) \vee \neg r_1 \vee \gamma({\Gamma_1}^1) \vee \gamma(\Gamma_2)$.

  If ${\Gamma_1}^1$ occurs in $named(\gamma)$, then we have generated the clauses $l_{\gamma'}({\Gamma_1}^1 \vee \Gamma_2) \vee \neg l_\gamma({\Gamma_1}^1)$ and $l_{\gamma'}({\Gamma_1}^1 \vee \Gamma_2) \vee \neg l_\gamma(\Gamma_2)$ thus after two resolution steps we obtain: $\neg l_{\gamma'}(\Gamma_1 \vee \Gamma_2) \vee \neg r_1 \vee \gamma'({\Gamma_1}^1 \vee \Gamma_2)$ i.e. $\neg \gamma'(\Gamma_1 \vee \Gamma_2) \vee \neg r_1 \vee \gamma'((\Gamma_1 \vee \Gamma_2)^1)$.

  Otherwise, we have either ${\Gamma_1}^1 \equiv 1$, hence in this case $\neg l_{\gamma'}(\Gamma_1 \vee \Gamma_2) \vee \neg r_1 \vee \gamma'(\Gamma_1 \vee {\Gamma_2}^1)$ is empty, or ${\Gamma_1}^1 = 0$ and in this case ${\Gamma_1}^1 \vee \Gamma_2 = \Gamma_2$ and $\gamma({\Gamma_1}^1) = \Box$ thus $\neg l_{\gamma'}(\Gamma_1 \vee \Gamma_2) \vee \neg r_1 \vee \gamma({\Gamma_1}^1) \vee \gamma(\Gamma_2)$ is equivalent to $\neg l_{\gamma'}(\Gamma_1 \vee \Gamma_2) \vee \neg r_1 \vee \vee \gamma'({\Gamma_2}^1 \vee \Gamma_2)$.

In both cases the clause set $\neg l_{\gamma'}(\Gamma_1 \vee \Gamma_2) \vee \neg var(\Gamma_1 \vee \Gamma_2) \vee \gamma'((\Gamma_1 \vee \Gamma_2)^1)$ has been generated, in at most 4 resolution steps. By symmetry we also generate the clause set $\neg l_{\gamma'}(\Gamma_1 \vee \Gamma_2) \vee var(\Gamma_1 \vee \Gamma_2) \vee \gamma'((\Gamma_1 \vee \Gamma_2)^0)$ Hence we have generate a clause set $\neg l_{\gamma'}(\Gamma_1 \vee \Gamma_2) \vee S''$ where $S'' \lhd_{\gamma'} \Gamma_1 \vee \Gamma_2$. This takes at most 8 resolution steps for each pair (Γ_1, Γ_2). $\Box$

The relation $\sim$ is inductively defined as follows: we write $\Delta_1 \sim \Delta_2$ if either $\Delta_1 \in \{1, 0\}$ or $\Delta_2 \in \{1, 0\}$ or $l_{\Delta_1} = l_{\Delta_2}$ and ${\Delta_1}^1 \sim {\Delta_2}^1$ and ${\Delta_1}^0 \sim {\Delta_2}^0$.

Informally $\Delta_1 \sim \Delta_2$ if Δ_1, Δ_2 only differ by their constant nodes.

Lemma 5 *Let S be a clause set. Assume that S encodes two BDDs Δ_1 and Δ_2 w.r.t. a BDD-naming γ. If S contains two clauses of the form $\neg u \vee \gamma(\Delta_i)$ $(i = 1, 2)$, then one can generate a clause set S' in at most $14 \times |\Delta_1| \times |\Delta_2|$ resolution or extension steps s.t. S' encodes $\Delta_1 \wedge \Delta_2$ w.r.t. a BDD-naming γ' s.t. $l_{\gamma'}(\Delta_1 \wedge \Delta_2) = u$. Moreover if $\Delta_1 \sim \Delta_2$ then the construction requires only $14 \times \min(|\Delta_1|, |\Delta_2|)$ steps.*

Proof: For each pair $(\Gamma_1, \Gamma_2) \neq (\Delta_1, \Delta_2)$ s.t. Γ_i occurs in Δ_i and $\Delta_i \in$

named(γ), we apply the extension rule on the variables $l_\gamma(\Gamma_1)$ and $l_\gamma(\Gamma_2)$.

This yields the clauses $p(\Gamma_1, \Gamma_2) \vee \gamma(\Gamma_i)$ $(i = 1, 2)$ and $\neg p(\Gamma_1, \Gamma_2) \vee \neg l_\gamma(\Gamma_1) \vee \neg l_\gamma(\Gamma_2)$. Then by applying again the extension rule with $a = b = p(\Gamma_1, \Gamma_2)$ we obtain $q(\Gamma_1, \Gamma_2) \vee p(\Gamma_1, \Gamma_2)$ and $\neg q(\Gamma_1, \Gamma_2) \vee \neg p(\Gamma_1, \Gamma_2)$. By resolution we get: $\neg q(\Gamma_1, \Gamma_2) \vee \gamma(\Gamma_i)$ $(i = 1, 2)$ and $q(\Gamma_1, \Gamma_2) \vee \neg l_\gamma(\Gamma_1) \vee \neg l_\gamma(\Gamma_2)$. This takes 2 extension steps and 4 resolution steps.

We define γ' as an extension of γ to the BDDs of the form $\Gamma_1 \wedge \Gamma_2$ where for any $i = 1, 2$, Γ_i occurs in Δ_i and $\Gamma_i \in$ *named*(γ) satisfying the following properties: $\gamma'(\Gamma_1 \wedge \Gamma_2) \stackrel{\text{def}}{=} q(\Gamma_1, \Gamma_2)$ if $(\Gamma_1, \Gamma_2) \neq (\Delta_1, \Delta_2)$ and $\gamma'(\Delta_1 \wedge \Delta_2) \stackrel{\text{def}}{=} u$. Notice that γ' is well defined since $\Gamma_1 \wedge \Gamma_2 \neq \Gamma'_1 \wedge \Gamma'_2$ if $\Gamma_1 \neq \Gamma'_1$ or $\Gamma_2 \neq \Gamma'_2$.

By definition, since $\Delta_1, \Delta_2 \in$ *named*(γ) we have $\Delta_1 \wedge \Delta_2 \in$ *named*(γ'). We prove that for any $\Gamma \in$ *named*(γ'), one can construct a clause set $\neg l_{\gamma'}(\Gamma_1 \wedge \Gamma_2) \vee S'$ s.t. $S' \lhd_{\gamma'} \Gamma_1 \wedge \Gamma_2$. By definition any $\Gamma \in$ *named*(γ') is of the form $\Gamma_1 \wedge \Gamma_2$ where Γ_i occurs in Δ_i and $\Gamma_i \in$ *named*(γ') $(i = 1, 2)$. S must contain two clause sets of the form $\neg l_\gamma(\Gamma_i) \vee S_i$ where $S_i \lhd_\gamma \Gamma_i$.

If $\Gamma_1 \equiv 0$ then $\Gamma_1 \wedge \Gamma_2 = 0$. Moreover, we have $S_1 = \Box$, thus $\neg l_\gamma(\Gamma_1)$ occurs in S. By resolving this clause with $\neg l_{\gamma'}(\Gamma_1 \wedge \Gamma_2) \vee \gamma(\Gamma_i)$ we get $\neg l_{\gamma'}(\Gamma_1 \wedge \Gamma_2)$, hence the proof is completed (since $\Box \lhd_{\gamma'} \Gamma_1 \wedge \Gamma_2$). The same holds if $\Gamma_2 \equiv 0$.

If $\Gamma_1 \equiv 1$ then $\Gamma_1 \wedge \Gamma_2 = \Gamma_2$. From $\neg l_\gamma(\Gamma_2) \vee S_2$ and $\neg l_{\gamma'}(\Gamma_1 \wedge \Gamma_2) \vee \gamma(\Gamma_2)$ we get $\neg l_{\gamma'}(\Gamma_1 \wedge \Gamma_2) \vee S_2$. Moreover, $S_2 \lhd_{\gamma'} \Gamma_2$ hence the proof is completed. The same holds if $\Gamma_2 \equiv 1$.

Now assume that $\Gamma_i \not\equiv 0, 1$. Let $r_i = var(\Gamma_i)$. By definition S_i is of the form $\neg r_i \vee \gamma({\Delta_i}^1) \cup r_i \vee \gamma({\Delta_i}^0)$ We assume, w.l.o.g. that $r_1 \geq r_2$ (the other case is symmetric). By definition, $var(\Gamma_1 \wedge \Gamma_2) = r_1$.

We distinguish two cases.

- If $r_1 = r_2 = r$, then $(\Gamma_1 \wedge \Gamma_2)^1 = {\Gamma_1}^1 \wedge {\Gamma_2}^1$.
  By applying the resolution rule between $\neg l_\gamma(\Gamma_i) \vee S_i$ and $\neg l_{\gamma'}(\Gamma_1 \wedge \Gamma_2) \vee \gamma(\Gamma_i)$ we obtain $\neg l_{\gamma'}(\Gamma_1 \wedge \Gamma_2) \vee \neg r \vee \gamma({\Gamma_i}^1)$.
  If ${\Gamma_1}^1$ and ${\Gamma_2}^1$ occur in *named*(γ) then we have generated the clauses $l_{\gamma'}({\Gamma_1}^1 \wedge {\Gamma_2}^1) \vee \neg l_\gamma({\Gamma_1}^1) \vee \neg l_\gamma({\Gamma_2}^1)$ thus after two resolution steps we obtain: $\neg l_{\gamma'}(\Gamma_1 \wedge \Gamma_2) \vee \neg r \vee \gamma'({\Gamma_1}^1 \wedge {\Gamma_2}^1)$ i.e. $\neg l_{\gamma'}(\Gamma_1 \vee \Gamma_2) \vee \neg r \vee \gamma'((\Gamma_1 \wedge \Gamma_2)^1)$.
  If ${\Gamma_1}^1 \notin$ *named*(γ) then we have either ${\Gamma_1}^1 \equiv 1$ or ${\Gamma_1}^1 \equiv 0$.
  If ${\Gamma_1}^1 \equiv 0$ then $(\Gamma_1 \wedge \Gamma_2)^1 \equiv 0 \equiv {\Gamma_1}^1$. If ${\Gamma_1}^1 \equiv 1$, then $(\Gamma_1 \wedge \Gamma_2)^1 = {\Gamma_2}^1$. In both cases, there exists $j = 1, 2$ s.t. $(\Gamma_1 \wedge \Gamma_2)^1 = {\Gamma_j}^1$. But in this case $\neg l_{\gamma'}(\Gamma_1 \wedge \Gamma_2) \vee \neg r \vee \gamma({\Gamma_j}^1)$ is equivalent to $\neg l_{\gamma'}(\Gamma_1 \wedge \Gamma_2) \vee \neg r \vee \gamma((\Gamma_1 \vee \Gamma_2)^1)$ hence the proof is completed.
  The same holds if ${\Gamma_2}^1 \notin$ *named*(γ).
- Now assume that $r_1 > r_2$. In this case $(\Gamma_1 \wedge \Gamma_2)^1 = {\Gamma_1}^1 \wedge \Gamma_2$. By applying

the resolution rule between $\neg l_\gamma(\Gamma_1) \vee S_1$, and $\neg l_{\gamma'}(\Gamma_1 \wedge \Gamma_2) \vee l_\gamma(\Gamma_1)$ we obtain $\neg l_{\gamma'}(\Gamma_1 \wedge \Gamma_2) \vee \neg r_1 \vee \gamma({\Gamma_1}^1)$. Moreover, we also have generated the clause $\neg l_{\gamma'}(\Gamma_1 \wedge \Gamma_2) \vee l_\gamma(\Gamma_2)$.

If ${\Gamma_1}^1 \in \mathit{named}(\gamma)$, we have generated the clause $l_{\gamma'}({\Gamma_1}^1 \wedge \Gamma_2) \vee \neg l_\gamma({\Gamma_1}^1) \vee \neg l_\gamma(\Gamma_2)$ (since ${\Gamma_1}^1 \neq \Delta_1$) thus after two resolution steps we obtain $\neg l_{\gamma'}(\Gamma_1 \wedge \Gamma_2) \vee \neg r_1 \vee \neg l_{\gamma'}({\Gamma_1}^1 \wedge \Gamma_2)$.

If ${\Gamma_1}^1 \notin \mathit{named}(\gamma)$ then ${\Gamma_1}^1$ must be equal to 1 or 0 and the proof follows as in the previous case.

In both cases the clause $\neg l_{\gamma'}(\Gamma_1 \wedge \Gamma_2) \vee \neg r_1 \vee \gamma'((\Gamma_1 \wedge \Gamma_2)^1)$ has been generated, in at most 4 resolution steps. By symmetry we also generate the clause $\neg l_{\gamma'}(\Gamma_1 \wedge \Gamma_2) \vee \neg r_1 \vee \gamma'((\Gamma_1 \wedge \Gamma_2)^0)$ Hence we have generated a clause set $\neg l_{\gamma'}(\Gamma_1 \wedge \Gamma_2) \vee S''$ where $S'' \lhd_{\gamma'} \Gamma_1 \wedge \Gamma_2$. We used 14 inference steps for each pair (Γ_1, Γ_2) s.t. Γ_i occurs in Δ_i.

If $\Delta_1 \sim \Delta_2$ then we are always in case 1 above. Hence we only need to compute BDDs of the form $\Gamma_1 \wedge \Gamma_2$, where Γ_1, Γ_2 correspond to the BDDs occurring at the *same* position p in Δ_1, Δ_2 respectively. The number of pairs is thus bounded by the number of positions occuring both in Δ_1 and Δ_2. Hence only $14 \times \min(|\Delta_1|, |\Delta_2|)$ steps are needed. □

If ϕ is a formula then we denote by $\mathit{Sbbd}(\phi)$ the maximal size of the reduced BDDs of the formulae occurring in ϕ. More formally: $\mathit{Sbbd}(\phi) \stackrel{\text{def}}{=} \max\{|\mathit{bdd}(\psi)| \mid \psi \in \mathit{SF}(\phi)\}$.

Using the above lemmata, it is easy to show that extended resolution can simulate BDD construction and simplification:

Theorem 1 *Let ϕ be a formula. Using the resolution and extension rules, one can generate from $\mathit{Cl}(\phi)$ a clause set S encoding $\mathit{bdd}(\phi)$ w.r.t. a BDD-naming γ s.t. $\gamma(\phi) = \{\{p_\phi\}\}$. Moreover the total number of steps is bounded by $c \times \mathit{Sbbd}(\phi)^4 \times |\phi|$ for some constant c.*

Proof: We show that this property is true for any formula ψ or $\neg\psi$, s.t. $\psi \in \mathit{SF}(\phi)$ (in particular it is true for ϕ).

The proof is by induction on the set of formulae.

- If ψ is an atom p, then $\mathit{Cl}(\phi)$ must contain a clause of the form $\{\neg p_\psi \vee p\}$. By definition $\mathit{bdd}(\psi)$ is labeled by p and we have $\psi^1 \equiv 1$, $\psi^0 \equiv 0$. Clearly, $\{p\} = \{p \vee \Box\} \cup \neg p \vee \emptyset$ hence $\{p\} \lhd_\gamma \mathit{bdd}(\psi)$. Similarly S contains a clause $\{\neg p_{\neg\psi} \vee \neg p\}$ and $\{\neg p\} \lhd_\gamma \mathit{bdd}(\neg\psi)$. Thus the proof is completed.
- ψ is the negation of a formula ψ'. By induction hypothesis one can construct a clause set encoding ψ' and $\neg\psi'$. Thus one can construct a clause set encoding ψ. Moreover, one can construct a clause set $\neg p_\psi \vee S'$ s.t. $S' \lhd_\gamma$

$bdd(\psi')$. Moreover $Cl(\phi)$ contains a clause $\neg p_{\neg\neg\psi'} \vee p_\psi$. From this clause and the clause set $\neg p_\psi \vee S'$ we obtain $\neg p_{\neg\neg\psi'} \vee S'$, which completes the proof since obviously $bdd(\psi')$ is identical to $bdd(\neg\neg\psi')$ (up to a renaming of the nodes).

- If ψ is the conjunction of two formulae ψ_1 and ψ_2. $Cl(\psi)$ contains two clauses $\neg p_\psi \vee p_{\psi_i}$ (for $i = 1, 2$). By Lemma 5 we generate a clause set S' s.t. S' encodes $bdd(\psi_1) \wedge bdd(\psi_2)$ w.r.t. a BDD-naming γ' s.t. $\gamma'(\psi_1 \wedge \psi_2) = p_\psi$. By Lemmata 2 and 3 we can obtain from S' a clause set S'' encoding the reduced form of $bdd(\psi_1) \wedge bdd(\psi_2)$ (i.e. $bdd(\psi)$) w.r.t. a BDD-naming γ'' s.t. $\gamma''(bdd(\psi)) = \gamma'(bdd(\psi_1) \wedge bdd(\psi_2)) = p_\psi$. Since the number of merging and elimination steps is necessarily bounded by the size of $bdd(\psi_1) \vee bdd(\psi_2)$ (see Point 2 in Proposition 1) and since the size of $bdd(\psi_1) \vee bdd(\psi_2)$ is at most $Sbbd(\phi)^2$ (by Point 2 in Proposition 1) it is clear that the total number of rules is polynomially bounded by $Sbbd(\phi)$. The construction of $bdd(\psi_1 \wedge \psi_2)$ takes at most $14 \times Sbbd(\phi)^2$ inference steps, then according to Lemma 2 and Lemma 3 its reduction takes at most $10 + 2 \times Sbbd(\phi)^2$ steps for each reduction rule, hence at most $10 \times Sbbd(\phi)^2 + 2 \times Sbbd(\phi)^4$ steps (since according to Proposition 1, the number of rules is bounded by $Sbbd(\phi)^2$). Thus the number of steps is bounded by $c \times Sbbd(\phi)^4$ for some constant c sufficiently high.

  Similarly, $Cl(\psi)$ must contain a clause $\neg p_{\neg\psi} \vee p_{\neg\psi_1} \vee p_{\neg\psi_2}$. Thus according to Lemma 4 we generate a clause set S' encoding $bdd(\neg\psi_1) \vee bdd(\neg\psi_2)$. By Lemma 2 and 3 we can obtain the reduced form of $bdd(\neg\psi_1) \vee bdd(\neg\psi_2)$, i.e. $bdd(\neg\psi_1 \vee \neg\psi_2) = bdd(\neg\psi)$. Again, the number of steps is bounded by $c \times Sbbd(\phi)^4$.
- If ψ is the disjunction of two formulae ψ_1 and ψ_2. $Cl(\psi)$ contains a clause $\neg p_\psi \vee p_{\psi_1} \vee p_{\psi_2}$. By Lemma 4 we generate a clause set S' s.t. S' encodes $bdd(\psi_1) \vee bdd(\psi_2)$ w.r.t. a BDD-naming γ' s.t. $\gamma'(\psi_1 \vee \psi_2) = p_\psi$. By Lemmata 2 and 3 we can obtain from S' a clause set S'' encoding the reduced form of $bdd(\psi_1) \vee bdd(\psi_2)$ (i.e. $bdd(\psi)$) w.r.t. a BDD-naming γ'' s.t. $\gamma''(bdd(\psi)) = \gamma'(bdd(\psi_1) \vee bdd(\psi_2)) = p_\psi$.

  Similarly, $Cl(\psi)$ must contain two clauses $\neg p_{\neg\psi} \vee p_{\neg\psi_i}$ for $i = 1, 2$. Thus according to Lemma 5 we generate a clause set S' encoding $bdd(\neg\psi_1) \wedge bdd(\neg\psi_2)$. By Lemma 2 and 3 we can obtain the reduced form of $bdd(\neg\psi_1) \wedge bdd(\neg\psi_2)$, i.e. $bdd(\neg\psi_1 \wedge \neg\psi_2) = bdd(\neg\psi)$.

  As for conjunctions, the total number of steps in bounded by $c \times Sbbd(\phi)^4$.
- If $\psi = (\psi_1 \Leftrightarrow \psi_2)$ then $Cl(\psi)$ contains the clauses:

$$\neg p_{\psi_1 \Leftrightarrow \psi_2} \vee p_{\psi_1 \Rightarrow \psi_2}$$

$$\neg p_{\psi_1 \Leftrightarrow \psi_2} \vee p_{\psi_2 \Rightarrow \psi_1}$$

$$\neg p_{\neg(\psi_1 \Leftrightarrow \psi_2)} \vee p_{\neg(\psi_1 \Rightarrow \psi_2)}$$

$$\neg p_{\neg(\psi_1 \Leftrightarrow \psi_2)} \vee p_{\neg(\psi_2 \Rightarrow \psi_1)}$$

  Moreover, it also contains the clauses defining $\psi_1 \Rightarrow \psi_2$ and $\psi_2 \Rightarrow \psi_1$, namely:

$$\neg p_{\psi_1 \Rightarrow \psi_2} \vee p_{\neg\psi_1} \vee p_{\psi_2}$$

$$\neg p_{\neg(\psi_1 \Rightarrow \psi_2)} \vee p_{\psi_1}$$

$$\neg p_{\neg(\psi_1 \Rightarrow \psi_2)} \vee p_{\neg\psi_2}$$

$$\neg p_{\psi_2 \Rightarrow \psi_1} \vee p_{\neg\psi_2} \vee p_{\psi_1}$$

$$\neg p_{\neg(\psi_2 \Rightarrow \psi_1)} \vee p_{\psi_2}$$

$$\neg p_{\neg(\psi_2 \Rightarrow \psi_1)} \vee p_{\neg\psi_1}$$

Using the same principle as in the two previous cases, we apply Lemma 4 to generate a clause set encoding the OBDD Δ_1, Δ_2 corresponding respectively to the formulae $\neg\psi_1 \vee \psi_2$ and $\neg\psi_2 \vee \psi_1$. Similarly, we apply Lemma 5 to construct the OBDD Δ'_1, Δ'_2 corresponding respectively to $\neg\psi_1 \wedge \psi_2$ and $\psi_1 \wedge \neg\psi_2$. As usual, this takes $14 \times Sbbd(\phi)^2$ steps for each construction. The size of the obtained BDDs is bounded by $Sbbd(\phi)^2$. Moreover by construction we have $\Delta_1 \sim \Delta_2 \sim \Delta'_1 \sim \Delta'_2$.

By applying again Lemma 5 we get a clause set encoding the OBBDs $\Delta_1 \wedge \Delta_2$ and $\Delta'_1 \wedge \Delta'_2$ (corresponding respectively to $\psi_1 \Leftrightarrow \psi_2$ and $\neg(\psi_1 \Leftrightarrow \psi_2)$). Since the size of the BDDs $\Delta_1, \Delta_2, \Delta'_1, \Delta'_2$ is bounded by $Sbbd(\phi)^2$ and since $\Delta_1 \sim \Delta_2 \sim \Delta'_1 \sim \Delta'_2$ this takes $14 \times Sbbd(\phi)^2$ inference steps and the size of the obtained BDD is at most $Sbbd(\phi)^2$. By applying Lemma 3 and 2 we obtain the reduced OBDD corresponding to $\psi_1 \Leftrightarrow \psi_2$ and $\neg(\psi_1 \Leftrightarrow \psi_2)$ in at most $(10 + 2 \times Sbbd(\phi)^2) \times Sbbd(\phi)^2$ inference steps.

As we have seen, at most $c \times Sbbd(\phi)^4$ steps are needed for each subformula hence $c \times Sbbd(\phi)^4 \times |\phi|$ are needed for the whole formula ϕ. □

As an important result we have:

Corollary 1 *Let ϕ be an unsatisfiable formula. There exists an extended refutation of $Cl(\phi)$ containing a number of steps polynomially bounded by $Sbbd(\phi)$.*

Proof: This follows immediately from Theorem 1 since $bdd(\phi) = 0$ hence $\neg p_\phi$ can be generated in a polynomial number of steps. But p_ϕ occurs in $Cl(\phi)$. □

4 Conclusion

We have proven that extended resolution polynomially simulates BDDs. As such, this result is mostly of theoretical interest, since extended resolution is not very useful in practice due to the huge branching factor of the extension rule (no successful propositional prover uses this rule). However, our proof also gives a hint of the formulae that can be considered for the extension steps,

which can greatly decrease the search space. A natural follow-up of the present work is to search for other extensions of the resolution method that are both less general and less costly than the extension rule, but still powerful enough to polynomially simulate BDDs.

Recently two new approaches [17,18] have been proposed for solving the SAT problem. These approaches are completely distinct and unrelated, but share some similarities: both are based on breath-first search (in the spirit of the original Davis and Putnam procedure), and use BDD (or more precisely ZDD, Zero-Suppressed Binary Decision Diagrams) for representing the search space (ZDD are used to represents sets of clauses in [17] and sets of clause sets in [18]). Both are strictly more powerful than resolution. It would be interesting to know whether these approaches can be simulated by extended resolution, using the same principle as in the present paper.

References

[1] R. Bryant, Symbolic boolean manipulation with ordered binary decision diagrams, ACM Computing Surveys (1992) 23 (3).

[2] G. S. Tseitin, On the Complexity of Derivation in Propositional Calculus, in: A. Slisenko (Ed.), Studies in Constructive Mathematics and Mathematical Logics, 1968.

[3] J. A. Robinson, A machine-oriented logic based on the resolution principle, J. Assoc. Comput. Mach. 12 (1965) 23–41.

[4] M. Davis, G. Logemann, D. Loveland, A Machine Program for Theorem Proving, Communication of the ACM 5 (1962) 394–397.

[5] J. A. Marques-Silva, K. Sakallah, GRASP: A Search Algorithm for Propositional Satisfiability, IEEE Trans. Computer 48 (5) (1999) 506–521.

[6] M. Sheeran, G. Stålmark, A tutorial on Stålmark's proof procedure for propositional logic, in: Volume 152 of Lecture Notes in Computer Science, Springer Verlag, 1998, pp. 82–99.

[7] J. F. Groote, J. P. Warners, The propositional formula checker HeerHugo, in: I. Gent, H. van Maaren, T. Walsh (Eds.), SAT20000: Highlights of Satisfiability Research in the year 2000, Frontiers in Artificial Intelligence and Applications, Kluwer Academic, 2000, pp. 261–281.

[8] C. Meinel, T. Theobald, Algorithms and Data Structures in VLSI Design: OBDD - Foundations and Application, Springer, 1998.

[9] T. Uribe, M. Stickel, Ordered Binary Decision Diagrams and the Davis-Putnam Procedure, in: J.-P. Jouannaud (Ed.), First conference on Constraints in Computational Logic, 1994.

[10] J. F. Groote, H. Zantema, Resolution and binary decision diagrams cannot simulate each other polynomially, Discrete Appl. Math. 130 (2) (2003) 157–171.

[11] O. Kullmann, On a generalization of extended resolution, Discrete Applied Mathematics 96-97 (1999) 149–176.

[12] S. Cook, The Complexity of Theorem Proving Procedures, in: Proc. of the 3th annual ACM Symposium on the Theory of Computing, 1971, pp. 151–158.

[13] A. Haken, The Intractability of Resolution, Theoretical Computer Science 39 (1985) 297–308.

[14] S. Cook, R. Reckhow, The Relative Efficience of Propositional Proof Systems, The Journal of Symbolic Logic 44 (1) (1979) 36–50.

[15] T. Boy de la Tour, An optimality result for clause form translation, Journal of Symbolic Computation 14 (1992) 283–301.

[16] A. Nonnengart, C. Weidenbach, Computing Small Clause Normal Form, in: A. Robinson, A. Voronkov (Eds.), Handbook of Automated Reasoning, Vol. I, Elsevier Science, 2001, Ch. 6, pp. 335–367.

[17] P. Chatalic, L. Simon, Multi-Resolution on compressed sets of clauses, in: Twelfth International Conference on Tools with Artificial Intelligence (ICTAI'00), 2000, pp. 2–10.
URL `http://www.lri.fr/ simon/recherche/papiers/ictai00.ps.gz`

[18] D. B. Motter, I. L. Markov, A compressed breadth-first search for satisfiability, in: ALENEX, 2002, pp. 29–42.