Abstract Branching for Quantified Formulas

Marco Benedetti*
LIFO, Université d'Orléans, France
Marco.Benedetti@univ-orleans.fr

Abstract

We introduce a novel search-based decision procedure for Quantified Boolean Formulas (QBFs), called *Abstract Branching*. As opposed to standard search-based procedures, it escapes the burdensome need for branching on both children of every universal node in the search tree. This is achieved by branching on existential variables only, while admissible universal assignments are *inferred*. Running examples and experimental results are reported.

Introduction

The language of Quantified Boolean Formulas (QBFs) allows us to wonder about the validity of statements like

$$\exists a \forall b \exists c.(a \vee b \vee c) \wedge (b \vee \neg c) \wedge (a \vee \neg b \vee \neg c) \wedge (\neg a \vee b)$$

where we ask if a truth value (TRUE or FALSE) exists for a such that for both truth values of b a truth value for c exists such that the given conjunction of constraints (or clauses) is invariably satisfied. Every problem that can be stated as a two-player finite game can be modeled in QBF. The QBF validity problem is itself a game between the $\exists$ player, who tries to satisfy every constraint, and the $\forall$ player, doing his best to contradict at least one clause. Many applications exist, like model checking for finite-state systems and conformant planning (see e.g. Rintanen 1999).

QBF instances from applications come out to be unexpectedly difficult to solve (the surprise being engendered by the comparatively large success of SAT solvers in related applications). Such difficulties led some to suggest the presence of inherent deficiencies in the language (Ansótegui, Gomes, & Selman 2005), and others to develop solving paradigms alternative to the classical one.

Classical decision procedures search the AND/OR semantic evaluation tree of the formula, looking for the existence of a suited sub-tree, called *model* or *strategy*. Alternative paradigms replace this search effort with a *solution process* based on quantified resolution or some special kind of skolemization (see the "Previous Work" section).

Such alternatives have been quite successfull. However, by giving up search in favour of inference they switch from time-intensive to memory-intensive computations. Families of instances exist in which even small elements systematically generate out-of-memory conditions. This has renewed the interest in search methods, which guarantee to work in polynomial space (once *learning* is properly restricted).

All the improvements to search-based procedures published over the years share the basic scheme of (Cadoli, Giovanardi, & Schaerf 1998). Namely, no one escapes the need for searching separately both branches of every node of the search tree associated with a universal quantifier.

In this paper, we present a novel search-based QBF decision procedure, called Abstract Branching (AB). As a key feature, it eludes the need for branching over universal variables by *abstracting* over their existence: AB searches at once in the widest possible set of branches (all of them, if possible). Ex-post, it expunges just those for which the current solution is guaranteed not to work. As the search goes on, a set of partial solutions is grown to satisfy more and more branches. If (and only if) all of them happen to be covered after an exhaustive search, the formula is TRUE.

The next section is devoted to a thorough presentation of this idea. Then, we discuss previous work, give details on our implementation, and comment on preliminary experimental results. Some final remarks close the paper.

Notation. We consider QBFs in *prenex conjunctive normal form* (CNF), consisting in a *prefix* with alternations of quantifiers, followed by a *matrix*, i.e. a conjunction of clauses (a clause set). Given a QBF F on variables $var(F)$, we denote by $\widetilde{F}$ its matrix, and by $var_\exists(F)$ ($var_\forall(F)$) the set of existentially (universally) quantified variables in F. Given a clause set G and an assignment Δ to (some of) the variables in G, we denote by $G * \Delta$ the CNF obtained by assigning Δ, i.e. by removing from G each literal which is false in Δ and each clause containing some literal true in Δ. An *empty clause* (contradiction) may result, written $\Box \in G * \Delta$. Or, an empty formula can be obtained, in which case Δ is a *model* for G. By $\mathcal{M}(G)$ we mean the set of all the models of G. A set $A \in 2^{var(F)}$ represent the assignment where $v{=}T$ if $v \in A$, and $v{=}F$ otherwise.

Abstract Branching

We describe abstract branching for contrast and comparison with the search procedure used in DPLL-like QBF solvers. We concentrate on $\forall\exists$*-formulas* exhibiting one single quan-

*The author is supported by "Le STUDIUM®" (France).

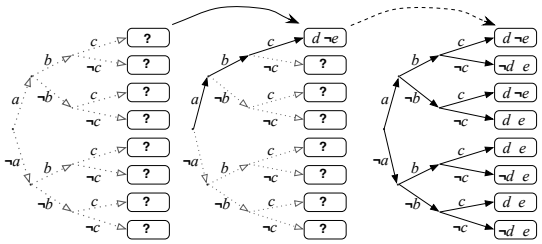


Figure 1: How to decide the QBF (2) according to a classical DPLL search. Dotted paths have to be visited yet.

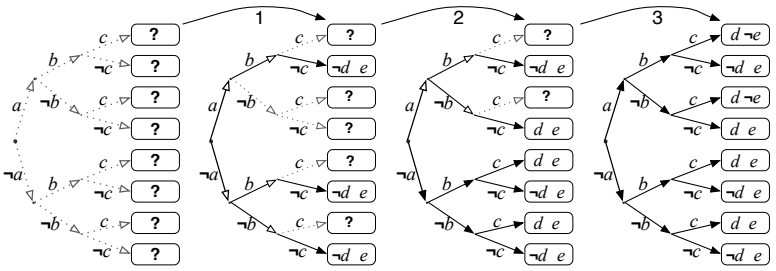


Figure 2: Abstract Branching's key idea: First propose existential assignments, then reason on universals. Solid paths represent neutralized scenarios.

tifier alternation (this restriction will be removed soon):

$$\forall U \exists E.\widetilde{F}(U,E) \tag{1}$$

At the request of deciding the validity of (1), standard search-based solvers select some assignment Δ to the universal variables U, then look for a model of $\widetilde{F} * \Delta$ (the *cofactored* matrix). The latter step is equivalent to solve a SAT problem. If no such model exists, the formula is declared to be FALSE. Otherwise, some other universal assignment is considered. The formula is claimed to be TRUE when *all* the universal assignments have been considered without ever failing to satisfy the matrix[1].

Such procedure descends from the definition of validity of a $\forall\exists$ formula. By using the term *scenario* (after Chen 2004) to mean any total assignment to the universal variables it is

Proposition 1 *A formula $\forall U \exists E.\widetilde{F}$ is TRUE iff for every scenario Δ over U, the formula $\widetilde{F} * \Delta$ is satisfiable.*

By choosing any arbitrary order for the universal variables, the space of universal scenarios fits into a complete binary tree with $|U|$ levels and $2^{|U|}$ branches, where each branch identifies a single scenario. DPLL-like solvers visit this tree in a depth-first manner, trying to label each leaf with the proper satisfying assignment, so to obtain a *model* (sometimes also called *strategy* or *certificate*) for the formula. The necessary and sufficient condition for the labeled tree to be a model is that the assignment/scenario along each branch, joined with the assignment inside the label reached via that branch, is invariably a model for the matrix.
As an example, let us consider the following TRUE formula.

$$\begin{aligned}\forall a \forall b \forall c \exists d \exists e.(a \vee \neg d \vee e) \wedge (b \vee c \vee e) \wedge (\neg b \vee c \vee \neg d) \wedge \\ (a \vee c \vee \neg d \vee \neg e) \wedge (\neg a \vee \neg c \vee \neg e) \wedge (\neg a \vee b \vee d) \wedge \\ (\neg c \vee d \vee \neg e) \wedge (\neg b \vee d \vee e) \wedge (a \vee \neg c \vee d \vee e)\end{aligned} \tag{2}$$

The initial situation is depicted in the leftmost picture in Figure 1, where the existential assignments we are looking for are represented as empty labels to be filled in. Suppose the scenario we visit is $\{a{=}T, b{=}T, c{=}T\}$ (uppermost branch). The leaf we reach corresponds to the formula

$$F * \{a{=}T, b{=}T, c{=}T\} = (\neg e) \wedge (d \vee \neg e) \wedge (d \vee e)$$

which is satisfied by the posing $\{d{=}T, e{=}F\}$: We label the first leaf by this assignment. We happen to succeed in repeating this process for every branch, so that in the end (rightmost picture in Figure 1) we know that (2) is TRUE.

Abstract Branching: The intuition

Abstract branching entirely reverses the classical search perspective, in favor of this one: First, guess some (total) assignment Γ to the *existential* variables. Then, ask: Which scenarios this existential assignment is "good" for? I.e.: to which leaves in the tree we can safely attach the label Γ?

The answer is: to all the leaves reached by universal assignments (scenarios) Δ such that $F * \Delta$ is satisfied by Γ, i.e. all the branches associated with models of $F * \Gamma$. This solution strategy descends from a definition of validity alternative (but equivalent) to the one given in Proposition 1.

Proposition 2 *A formula $\forall U \exists E.\widetilde{F}$ is TRUE iff a set $\mathcal{A}$ of assignments to the existential variables E exists such that $\cup_{\alpha \in \mathcal{A}} \mathcal{M}(\widetilde{F} * \alpha)$ contains every scenario over U.*

Let us consider again the QBF (2). This time, we ignore universal variables and pick some assignment to $var_\exists(F) = \{d, e\}$, for example $\Gamma = \{d{=}F, e{=}T\}$. How do we decide where to attach this label? We observe that the formula

$$F * \{d{=}F, e{=}T\} = (a \vee c) \wedge (\neg a \vee \neg c) \wedge (\neg a \vee b) \wedge (\neg c) \tag{3}$$

by construction only mentions universal variables, so that each time we assign a, b, and c to satisfy (3) we identify some scenario where Γ is the (or at least is one) right label.

In our example, the models of (3) are $\{a{=}F, b{=}T, c{=}F\}$, $\{a{=}F, b{=}F, c{=}F\}$ and $\{a{=}F, b{=}F, c{=}T\}$. This information allows us to move the first step in Figure 2.

Let us call *neutralized* those scenarios for which we have a valid label (solid lines in Figure 2). Once recognized, neutralized scenarios need no further consideration in our attempt to discover a model, as by construction assignments along neutralized paths cannot lead to a contradiction.

Suppose the next existential assignment we pick is $\Gamma' = \{d{=}T, e{=}T\}$. This time we ask: Which paths among the *not-yet-neutralized* ones Γ' is good for? The answer is computed as before. It is depicted in Figure 2 (second step).

The set of neutralized scenarios keeps on enlarging monotonically. By using any systematic procedure for generating existential candidates we are guaranteed to encounter only one out of two possible outcomes: Either each scenario is eventually neutralized (the formula is TRUE by Proposition 2), or we run out of existential candidates before such condition is met (the formula is FALSE by Proposition 2). In

[1] Such worst-case behaviour is alleviated by look-back techniques (e.g. conflict-directed backjumping, model caching) that re-use information gathered from searching in previous branches.

our working example, it suffices to consider one more existential candidate, namely $\{d{=}T, e{=}F\}$ to neutralize every scenario: The formula is TRUE (last step in Figure 2).

Some details need to be filled out before any realistic algorithm emerges out of the search strategy just sketched. Let us just briefly outline the plan of exposition.

Backtrack search. DPLL-like algorithms grow total assignments out of partial ones, within a backtrack-based search procedure. Also, they don't wait for assignments to be total before reasoning about them. Key properties—such as being sufficient to satisfy the matrix, or to contradict some clause—are checked on the partially specified candidate. Does this apply to abstract branching?

The answer—made formal in the next section—comes from the following analysis. We cannot tell scenarios neutralized by some candidate Γ from non-neutralized ones until $F * \Gamma$ "loses" every existential variable, which might not happen until Γ is total. Still, if we build Γ step by step we can realize that some scenarios will *never be neutralized* by any extension of the current partial candidate. Let us consider again the QBF (2). We start with the partial assignment $\Gamma = \{d{=}F\}$, so to obtain $F * \Gamma = F'$, with $F' = (b \vee c \vee e) \wedge (\neg a \vee \neg c \vee \neg e) \wedge (\neg a \vee b) \wedge (\neg c \vee \neg e) \wedge (\neg b \vee \neg e)$

We isolate the maximal subset $F'_\forall \subseteq F'$ in which existential variables are no longer mentioned. In our case, it is $F'_\forall = (\neg a \vee b)$. This set has two essential properties:

1. It excludes the possibility to neutralize every scenario inconsistent with $F'_\forall$ itself. No superset of Γ can be used to label branches identified by non-models of $F'_\forall$.
2. $F'_\forall$ is contained not only in F', but in any co-factored matrix $F * \Gamma'$ we obtain by extending Γ to $\Gamma' \supseteq \Gamma$. So, whatever the presence of $F'_\forall$ is excluding, it stays excluded from every candidate specializing Γ.

This observation leads us to design a least-commitment optimistic search procedure. It starts by considering every scenario as *neutralizable*, so long as no evidence exists of the contrary. As existential candidates grow, they cut away some scenarios from the set of those we can neutralize. If such set becomes empty, the current existential candidate has no extension leading to a valid label. Hence, we backtrack. Otherwise, we obtain the empty matrix, and accumulate neutralized scenarios in an enlarging set.

Multiple alternations. The abstract branching technique is quite intuitive for the simple $\forall\exists$ case. It generalizes nicely to solve general QBFs with any number of quantifier alternations. Before we address the problem in the full generality, we need to develop some formal notions. This will be done in the Section "*Formalization*".

Forward Inferences. QBF and SAT solvers to carry out fast forward inferences by performing *unit clause propagation* after each branching step. The abstract-branching framework preserves the power of forward reasoning, as shown in the "*Forward Reasoning*" section.

Data Structures. Solvers for quantified formulas rely on tailored, well-engineered data structures. The objects we manipulate blend clause sets with sets of scenarios (subsets of a powerset). We sort out an appropriate solution from the literature in the "*Data Structures*" section.

Formalization

Definition 1 (Quantification Structure) *Given a finite set of propositional symbols V, a* quantification structure *(QS) over V is a quadruple $\langle V_\exists, V_\forall, \delta, \mathit{dom}\rangle$ where $V_\exists$ and $V_\forall$ are a partition of V ($V_\exists \cup V_\forall = V$ and $V_\exists \cap V_\forall = \emptyset$), $\delta : V_\exists \to [0, 1, \ldots, |V_\forall|]$ and $\mathit{dom} : V_\exists \to 2^{V_\forall}$ are two functions such that $\delta(x) \leq \delta(y) \implies \mathit{dom}(x) \subseteq \mathit{dom}(y)$.*

A quantification structure is used to capture all the relevant information about how universal and existential variables relate to one another in a closed quantified formula. The intuition is that a variable $e \in V_\exists$ is at *depth* $\delta(e)$ when it is dominated (i.e. preceded in the left-to-right prefix order) by $\delta(e)$ different universal scopes, which together define a dom-inating subset $\mathit{dom}(e)$ for e.

The functions $\delta(\cdot)$ and $\mathit{dom}(\cdot)$ in a quantification structure $\langle V_\exists, V_\forall, \delta, \mathit{dom}\rangle$ are extended to any clause set G with $var(G) \subseteq V_\exists \cup V_\forall$, as $\delta(G) \doteq \min_{v \in var(G) \cap V_\exists} \delta(v)$ and $\mathit{dom}(G) \doteq \mathit{dom}(\delta(G))$ respectively. The intuition is that formulas are treated as if they were their shallowest existential variable (i.e. the more external one in their prefix).

Definition 2 (Abstract Formula) *An* abstract formula *(AF) defined over the QS $\langle V_\exists, V_\forall, \delta, \mathit{dom}\rangle$ is a couple $\langle \mathcal{U}, F\rangle$ where F is a clause set, $var(F) \subseteq V_\exists \cup V_\forall$, and $U \subseteq 2^{\mathit{dom}(F)}$.*

Abstract formulas can be seen as a generalization of QBFs. The validity of a QBF requires the existence of a tree of satisfying assignments shaped after the prefix. Conversely, in the AF $\langle \mathcal{U}, F\rangle$ we are only interested in the validity of the set of QBFs with matrices $\{F * \Delta, \Delta \in \mathcal{U}\}$ where $\mathcal{U}$ represents some subset of all the possible truth assignments to the variables in $\mathit{dom}(F)$.

Definition 3 (Association with QBFs) *The QS associated with a QBF $F = Q_0 V_0 \ldots Q_n V_n . \widetilde{F}$ is defined by $V_\exists = var_\exists(F)$, $V_\forall = var_\forall(F)$, $\delta(v) = \lfloor i/2 \rfloor$ if $v \in V_i \cap V_\exists$, and $\mathit{dom}(v) = \{w \in V_i | Q_i = \forall, v \in V_j, 0 \leq i < j\}$. The AF associated with the QBF F is defined over the quantification structure associated with F as $\langle 2^{\mathit{dom}(\widetilde{F})}, \widetilde{F}\rangle$.*

For example, the QS associated with $\forall a \forall b \exists c \forall d \exists e \exists f.\ \widetilde{M}(a, b, c, d, e, f)$ is defined by $V_\exists = \{c, e, f\}$, $V_\forall = \{a, b, d\}$, $\delta(c) = 1$, $\delta(e) = \delta(f) = 2$, $\mathit{dom}(c) = \{a, b\}$, and $\mathit{dom}(e) = \mathit{dom}(f) = \{a, b, d\}$. The AF for the same QBF is $\langle \{\emptyset, \{a\}, \{b\}, \{ab\}\}, \widetilde{M}(a, b, c, d, e)\rangle$.

Given a QS $\langle V_\exists, V_\forall, \delta, \mathit{dom}\rangle$ and a clause set G with $var(G) \subseteq V_\exists \cup V_\forall$, we denote by $G_\forall \subseteq G$ the maximal subset of G such that $var(G_\forall) \subseteq V_\forall$, and by $G_{\forall\exists}$ its complement to G. Essentially, $G_\forall \cup G_{\forall\exists}$ is a partition of G which separates clauses only mentioning universal variables ($G_\forall$) from those also mentioning existential variables ($G_{\forall\exists}$).

Definition 4 (Abstract Star Operator) *Given an abstract formula $\langle \mathcal{U}, F\rangle$ and a literal l with $\delta(l) = \delta(F)$, we define $\langle \mathcal{U}, F\rangle * l \doteq \langle \mathcal{U}', F'\rangle$ where $F' = (F * l)_{\forall\exists}$ and $\mathcal{U}' = \{u \in 2^{\mathit{dom}(F')}$ such that $u \cap \mathit{dom}(F) \in \mathcal{U}$, and $\square \notin (F * l)_\forall\}$*

The intuition is that the abstract star operator assigns a truth value to some variable e in the "outermost" existential scope of an AF (by the condition $\delta(l) = \delta(F)$) *abstracting* over the

existence of unassigned universal variables to the left of e. As a result, it produces a simpler abstract formula $\langle \mathcal{U}', F' \rangle$ that no longer mentions the variable e in F', and whose set of universal scenarios $\mathcal{U}'$ has been properly shrunk to exclude cases inconsistent with the assignment over e. Also, $\mathcal{U}'$ is extended to belong to a wider space than $\mathcal{U}$'s one, if $\delta(F') > \delta(F)$. Indeed, $\mathcal{U}' \subseteq 2^{dom(F')}$ and $dom(F) \subseteq dom(F')$.

Let us denote by $\mathcal{N}\downarrow^{\forall}A$ the *universal projection* of $\mathcal{N} \subseteq 2^B$ over $A \subseteq B$, defined as $\mathcal{N} \downarrow^{\forall} A = \{x \in 2^A | \forall y \in 2^{B \setminus A}. x \cup y \in \mathcal{N}\}$. The set $x \in 2^A$ belongs to $\mathcal{N}\downarrow^{\forall}A$ only if we find in $\mathcal{N}$ every set obtained by extending x with zero or more elements from $B \setminus A$. We give semantics to abstract formulas through the concept of *neutralized* scenarios.

Definition 5 (Neutralized Set) *The* neutralized set *associated with the AF $\langle \mathcal{U}, F \rangle$, written* $\text{NEUTR}(\langle \mathcal{U}, F \rangle)$*, is a subset of $\mathcal{U}$ inductively defined as follows:*

1. *if $\square \in F$, then* $\text{NEUTR}(\langle \mathcal{U}, F \rangle) = \emptyset$
2. *if $F = \emptyset$, then* $\text{NEUTR}(\langle \mathcal{U}, F \rangle) = \mathcal{U}$
3. *otherwise,* $\text{NEUTR}(\langle \mathcal{U}, F \rangle) = \mathcal{N}\downarrow^{\forall} dom(F)$*, where* $\mathcal{N} = \text{NEUTR}(\langle \mathcal{U}, F \rangle * v) \cup \text{NEUTR}(\langle \mathcal{U}, F \rangle * \neg v)$*, and $v \in V_\exists$ is any variable such that $\delta(v) = \delta(F)$.*

We call fully neutralized *any AF with* $\text{NEUTR}(\langle \mathcal{U}, F \rangle) = \mathcal{U}$.

Notice that all the definitions given so far are for general QBFs (not just $\forall\exists$ alternation), and that the universal projection operator takes care of multiple alternations. It plays no role in $\forall\exists$ formulas (where the $dom(.)$ function by definition always evaluates to the same powerset), but is the key to joining neutralized sets from higher universal depths. Indeed, suppose that a subset $\mathcal{U}^+ \subseteq \mathcal{U}$ of $\mathcal{F} = \langle \mathcal{U}, F \rangle$ is neutralized in $\mathcal{F}^+ = \langle \mathcal{U}, F \rangle * e$, while $\mathcal{U}^- \subseteq \mathcal{U}$ is neutralized in $\mathcal{F}^- = \langle \mathcal{U}, F \rangle * \neg e$. So long as $\delta(\mathcal{F}) = \delta(\mathcal{F}^+) = \delta(\mathcal{F}^-)$ we simply join these two sets: They complement each other in their ability to neturalize scenarios. But, if $\delta(\mathcal{F}) < \delta(\mathcal{F}^+)$ and/or $\delta(\mathcal{F}) < \delta(\mathcal{F}^-)$ (i.e. we have assigned the last existential variable in the present scope so the next one lay at a higher alternation depth), we can accept the joint neutralization effort of the two sub-instances with respect to those cases only that are completely neutralized in $dom(F)$. This amounts to cut away via the projection operators all the scenarios that have been only partially (or not at all) neutralized.

Theorem 1 *The QBF F is TRUE iff the abstract formula associated with F is fully neutralized.*

A neutralized-set computation procedure built after Definition 5 is given in Algorithm 1 (the explanation of Line 1 is deferred until the next section). It can be used as a decision procedure for the QBF F by calling it on the abstract formula associated with F by Definition 3. Such invocation is meant to check whether the input formula is fully neutralized (which happens iff the algorithm returns $2^{dom(F)}$). By Theorem 1 this answers the validity problem over F in a sound and complete way. Figure 3 illustrates a sample run of the algorithm.

Exploiting forward inferences

Let us call $\exists$*-unit clause* a clause mentioning exactly one existential literal. We may think of a unit clause $C = \{l\}$ in some CNF formula F as a clause such that $\square \in F * \neg l$ (i.e. a literal whose negation causes an immediate contradiction). By analogy, we give the following definition.

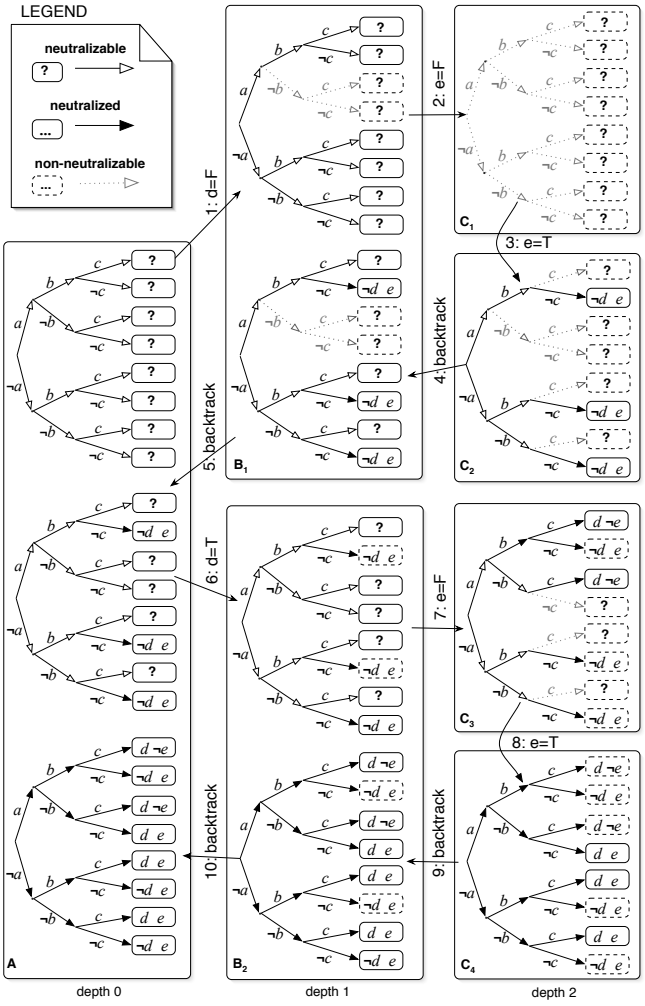


Figure 3: AB solves the QBF (2). The first component of each AFs is depicted, the second is the original formula in A, the empty formula in C_{1-4}, the clause set $(b \vee c \vee e) \wedge (\neg a \vee \neg c \vee \neg e) \wedge (\neg c \vee \neg e) \wedge (\neg b \vee e) \wedge (a \vee \neg c \vee e)$ in B_1, and $(a \vee e) \wedge (b \vee c \vee e) \wedge (a \vee c \vee \neg e) \wedge (\neg a \vee \neg c \vee \neg e)$ in B_2.

Definition 6 (Abstract Unit Clause) *An $\exists$-unit clause $C \in F$ with existential literal l is a* partial unit clause *for the abstract formula $\langle \mathcal{U}, F \rangle$, $\mathcal{U} \neq \emptyset$, with $\langle \mathcal{U}, F \rangle * \neg l = \langle \mathcal{U}', F' \rangle$ if $\mathcal{U}' \subset \mathcal{U}$. It is a* total *unit clause if, in addition, $\mathcal{U}' = \emptyset$.*

The algorithmic intuition is that an abstract (partial) unit clause identifies "bad" branching choices that immediately contract the neutralizable set. If we contradict a total unit, such set becomes suddenly empty, and the procedure has to backtrack. We don't need to wait until we encounter the empty set: We foresee by forward inference that a total unit is a *necessary* consequence given the current state of the search, and just *propagate* this information.

Definition 7 (Abstract unit clause propagation) *The abstract formula $\mathcal{F}'$ obtained via* abstract unit clause propa-

Algorithm 1: absBranching

input : An abstract formula $\langle \mathcal{U}, F \rangle$
output: A subset of $2^{dom(F)}$

1 $\langle \mathcal{U}, F \rangle \leftarrow AUCP(\langle \mathcal{U}, F \rangle)$;
2 **if** $\square \in F$ **then**
  // No hope to neutralize further scenarios
3 **return** $\emptyset$;
4 **else if** F *is empty* **then**
  // Fully neutralized subformula
5 **return** U;
  else
  // Inductive case: branch over two sub-problems
6 $e \leftarrow$ one variable in $V_\exists$ with $\delta(F) = \delta(e)$;
7 $U^+ \leftarrow$ absBranching$(\langle \mathcal{U}, F \rangle * \{e{=}T\})$;
8 $U^- \leftarrow$ absBranching$(\langle \mathcal{U}, F \rangle * \{e{=}F\})$;
9 **return** $(U^+ \cup U^-)\downarrow^{\forall} dom(F)$;

gation *(AUCP) from the abstract formula $\mathcal{F}$, written $\mathcal{F}' = AUCP(\mathcal{F})$, is defined as $AUCP(\mathcal{F}) = AUCP(\mathcal{F} * l)$ if l is the unique existential literal of some total unit clause in $\mathcal{F}$, and as $AUCP(\mathcal{F}) = \mathcal{F}$ if no such clause exists or $\square \in \mathcal{F}$.*

Abstract unit propagation prevents us from exploring subspaces where no scenario can be neutralized, so we state

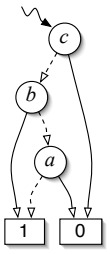
Property 1 *For every abstract formula $\mathcal{F}$, $AUCP(\mathcal{F})$ is fully neutralized iff $\mathcal{F}$ is fully neutralized.*

This property justifies line 1 in the pseudo-code. To recognize total unit clauses it suffices to keep on watching $\exists$-unit clauses, and check whether the scenarios they would cut away if contradicted are covering the whole working set. If so, they are total unit by Definition 6 and can be propagated. For example, three $\exists$-unit clauses on e are present after the first step in Figure 3: $\neg b \vee e$, $a \vee \neg c \vee e$, and $b \vee c \vee e$. It is easy to see that—if contradicted on e—these clauses would prune every scenario where either b=T or a=F, thus cancelling the whole working set as depicted after Step 1. So, we propagate $e{=}T$ by AUCP and jump after Step 3 straightaway.

Data Structures

Algorithm 1 manipulates AFs, i.e. couples made up of a set of scenarios and of a set of clauses. The former element of the couple undergoes *join*, *intersection* and *projection* operations (lines 7-9), the latter is subject to a slight variation over subsumption/resolution transformations (lines 7,8).

- Sets of scenarios are represented via *reduced ordered binary decision diagrams* (Bryant 1986) (ROBDDs, or just BDDs). Such representation may be exponentially more succinct than explicit ones, like the trees used so far (Wegener 2000). The BDD way of representing subsets of a powerest 2^S is to associate an *if-then-else* decision variable with each element in S, then to construct a directed acyclic graph representing the characteristic function of the subset. A path in the diagram leading to the sink node "1" represents the subsets containing (resp. missing) all the elements associated with a *then*-decision (resp. *else*-decision), while the presence of other elements is not constrained. In our case, decision variables are associated one-to-one with universal variables.



For example, the BDD aside represents the universal scenarios we observe after Step 3 in Figure 3: Solid (dashed) arrows denote then-branches (else-branches). The decision order is c, b, a. BDDs are ideally suited to representing *broad* scenarios: The initial one, that takes into account every assignment, is represented by a tiny BDD with no decision node and only one then-arc heading for the 1 sink. In reduced/ordered diagrams all paths encounter variables in the same order, and no two nodes represent the same set (each one has a *canonic* representation). This allows for information sharing among abstract formulas at different stack depths in Algorithm 1. The operations on sets we need (conjunction, projection, etc.) can be performed efficiently on the involved BDDs.

- Clause set representation is based on *lazy data structures* (Zhang 1997), in the QBF-specific version of (Gent *et al.* 2004). We designed a variant of the latter which addresses two additional issues:
  - Clauses with universal literals only have to be immediately recognized and removed, according to Def. 4.
  - Unit clases that are not total have to be taken on a watched list of partial unit clauses, where they wait to possibly become total as a side effect of a contraction of the working scenario, according to Def. 6.

Relation to Previous Work

Search-based QBF solvers are based on the seminal paper (Cadoli, Giovanardi, & Schaerf 1998), where trivial truth/falsity tests, monotone literals and forced assignments rules are introduced as well. Major enhancements have been (1) learning/caching of (in)validity search outcomes for sub-formulas (Letz 2002; Giunchiglia, Narizzano, & Tacchella 2002; Zhang & Malik 2002), (2) *lazy data structures* (Giunchiglia, Narizzano, & Tacchella 2001), and (3) partial unfolding and quantifier inversion (Rintanen 2001). Recently, a data-structure level integration between a search-based QBF procedure and a SAT solver, called "S-QBF" has been shown to improve the state of the art on some families (Samulowitz & Bacchus 2005). In another recent work (Remshagen & Truemper 2005) a specialized search-based algorithm (*only* working on the "Q-ALL" class of formulas) was shown to outperform other approaches. Some strong solvers *not* based on search emerged in the last years. We mention Quantor (Biere 2004), QMRES (Pan & Vardi 2004), and sKizzo (Benedetti 2005), and refer the reader to (Benedetti 2004) for a thorough discussion.

As opposed to these "alternative" approaches, AB is still based on searching the space of assignments, but we avoid branching on universal variables: Admissible values for universals are *inferred*. AB guarantees by construction to generate each existential assignment *at most once*, while usual procedures may be redundant in this respect. Furthermore, on instances with a universal outermost scope AB is able to provide a natural measure of advancement during long runs: the percentage of scenarios already neutralized. Also, if such instances are unsatisfiable, all the non-neutralized branches are eventually computed (instead of just one).

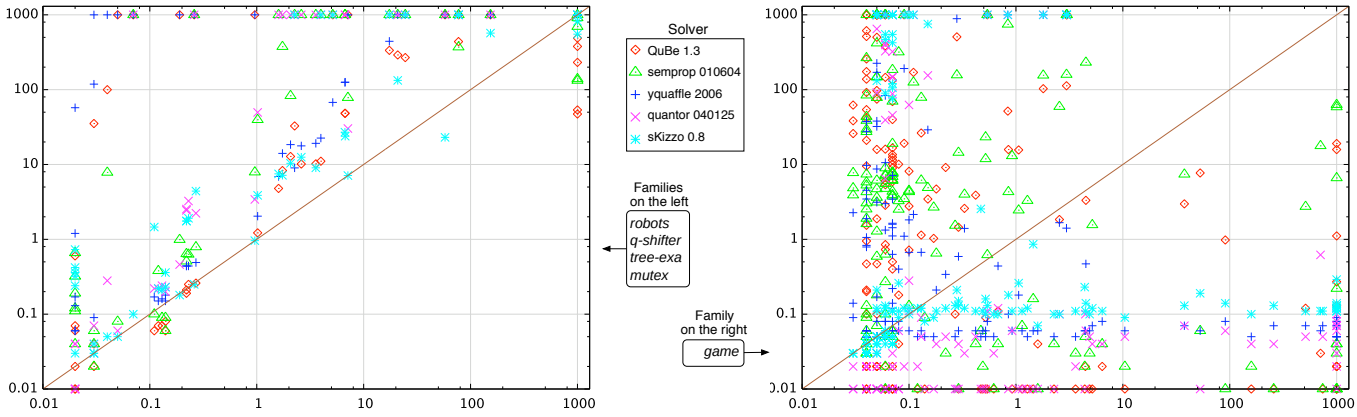


Figure 4: AB performance on $\forall\exists$ families. The x axis gives runtime (sec) for AB, the y axis for the others. Timeout is 1000s.

Implementation and Experimentation

We implemented AB within our existing solver sKizzo, downloadable from (Benedetti 2006). As a preliminary experimental evaluation we compare AB with state-of-the-art solvers (including sKizzo itself as it was *before* the inclusion of AB) on all the families in the QBFLIB with $\forall\exists$ alternation (*mutex*, *shifter*, *robots*, *tree*, *game*, totaling to 192 instances). AB solved 91.7% of the test-set in 1000s, followed by QuBE (87.0%), sKizzo (85.4%), semprop (84.9%), yquaffle (84.9%), and Quantor (78.1%). Seven instances were solved by one solver only, six of which by AB. The runtime distribution (Figure 4) is quite favorable to AB for the first four families. It is more scattered for the last family (*game*): AB comes out to be highly *complementary* to other approaches, i.e. it solves quickly instances difficult for the others, and vice-versa. Indeed, while no solver (and no combination of solvers missing AB) manages to complete the whole family, any solver running in parallel with AB easily decides it all. For example, while both AB and sKizzo encounter instances requiring hundreds of seconds and they sometimes timeout, their parallel version completely decides the *game* family in $24.68s$, with an average time per instance of $0.17s$ and a $2.99s$ worst case. Table 1 exemplifies some improvements over the state of the art.

Instance	S-QBF	Quaffle	QuBE	Quantor	sKizzo	A. B.
game20_20_40_2	440.94	—	98.26	0.08	—	1.52
game20_25_25_1	309.46	—	369.50	—	—	0.03
game20_25_25_2	125.29	—	2874.96	—	—	0.01
game20_25_25_3	40.06	—	1150.51	—	—	0.02
game20_25_25_4	222.13	—	1651.43	—	—	4.15
game20_25_50_1	221.74	—	1657.63	—	—	4.16
game50_25_25_1	64.22	—	1869.70	—	—	0.02
game50_25_25_3	4.13	—	—	—	—	0.02
game50_25_25_4	1.63	—	51.48	—	—	0.02
game100_25_25_2	0.73	—	—	9.26	0.07	0.01
game100_25_25_3	0.63	4.06	—	0.04	0.02	0.01
game150_25_25_1	0.00	0.00	—	0.01	0.01	0.00
game150_25_25_2	4.22	4.34	—	0.01	0.02	0.00
game150_25_25_4	0.30	208.79	—	0.01	0.01	0.00
robots1_5_2_72.7	221.70	19.64	1385.68	—	—	3.22
robots1_5_2_42.7	672.14	288.06	565.01	—	—	106.82
robots1_5_2_61.6	268.29	99.34	424.87	—	—	21.90

Table 1: Results obtained on some difficult instances. All the columns but the last two are taken from (Samulowitz & Bacchus 2005), where they exemplify cases difficult for search-based solvers and unsolvable by others. A comparable experimental setting is used for sKizzo/AB. "—" is time/mem-out (5000s/3Gb).

Conclusions and Future Work

We introduced a new algorithm for deciding QBFs which radically departs from previous approaches. In a first basic implementation it is already improving the state of the art on some families. As a future work, we plan to elude yet another of QBF solvers' weaknesses, i.e. the need for branching according to the left-to-right order of scopes.

Acknowledgments. We thank the anonymous reviewers for their suggestions, and Arnaud Lallouet for saving us quite some time we re-invested in writing this paper.

References

Ansótegui, C.; Gomes, C. P.; and Selman, B. 2005. The Achilles' Heel of QBF. In *Proc. of AAAI'05*.

Benedetti, M. 2004. sKizzo: a QBF Decision Procedure based on Propositional Skolemization and Symbolic Reasoning. Technical Report 04-11-03, ITC-irst.

Benedetti, M. 2005. Evaluating QBFs via Symbolic Skolemization. In *Proc. of LPAR'04*.

Benedetti, M. 2006. `sKizzo`'s web site, `http://sKizzo.info`.

Biere, A. 2004. Resolve and Expand. In *Proc. of SAT'04*.

Bryant, R. E. 1986. Graph-based algorithms for Boolean function manipulation. *IEEE Transaction on Computing* C-35(8):677–691.

Cadoli, M.; Giovanardi, A.; and Schaerf, M. 1998. An Algorithm to Evaluate Quantified Boolean Formulae. In *Proc. of AAAI/IAAI '98*.

Chen, H. 2004. Quantified Constraint Satisfaction and Bounded Treewidth. In *Proc. of ECAI '04*.

Gent, I.; Giunchiglia, E.; Narizzano, M.; Rowley, A.; and Tacchella, A. 2004. Watched Data Structures for QBF Solvers. *Proc. of SAT'03*.

Giunchiglia, E.; Narizzano, M.; and Tacchella, A. 2001. QuBE: A system for deciding Quantified Boolean Formulas Satisfiability. In *Proc. of IJCAR'01*.

Giunchiglia, E.; Narizzano, M.; and Tacchella, A. 2002. Learning for quantified boolean logic satisfiability. *Proc. of AAAI02*.

Letz, R. 2002. Lemma and model caching in decision procedures for quantified boolean formulas. In *Proc. of TABLEAUX'02*.

Pan, G., and Vardi, M. 2004. Symbolic Decision Procedures for QBF. In *Proc. of CP'04*.

Remshagen, A., and Truemper, K. 2005. An Effective Algorithm for the Futile Questioning Problem. *JAR* 34(1):31–47.

Rintanen, J. 1999. Construction Conditional Plans by a Theorem-prover. *JAIR* 10:323–352.

Rintanen, J. 2001. Partial implicit unfolding in the Davis-Putnam procedure for quantified boolean formulae. In *Proc. of LPAR'01*.

Samulowitz, H., and Bacchus, F. 2005. Using SAT in QBF. *Proc. of the CP05*.

Wegener, I. 2000. *Branching Programs and Binary Decision Diagrams*. Monographs on Discrete Mathematics and Applications. SIAM.

Zhang, L., and Malik, S. 2002. Conflict driven learning in a quantified Boolean Satisfiability solver. *Proc. of ICCAD02*.

Zhang, H. 1997. Sato: An efficient propositional prover. *Proc. of CADE'07* 272–275.