A Branching Heuristics for Quantified Renamable Horn Formulas

Sylvie Coste-Marquis, Daniel Le Berre, and Florian Letombe*

CRIL, CNRS FRE 2499
{coste, leberre, letombe}@cril.univ-artois.fr

Abstract. Many solvers have been designed for $\mathcal{QBF}$s, the validity problem for Quantified Boolean Formulas for the past few years. In this paper, we describe a new branching heuristics whose purpose is to promote renamable Horn formulas. This heuristics is based on Hébrard's algorithm for the recognition of such formulas. We present some experimental results obtained by our QBF solver **Qbfl** with the new branching heuristics and show how its performances are improved.

1 Introduction

QBF, the validity problem for $\mathcal{QBF}$s, has a growing importance in AI. This can be explained by the fact that, as the canonical PSPACE-complete problem, many AI problems can be polynomially reduced to QBF; furthermore, there is some empirical evidence from various AI fields (including among others planning, nonmonotonic reasoning, paraconsistent inference) that a translation-based approach can prove more "efficient" than domain-dependent algorithms dedicated to such AI tasks (see [1, 2, 3, 4, 5]). Accordingly, many QBF solvers have been designed and evaluated for the past few years (see mainly [6, 7, 8, 9, 10, 11, 12, 13, 14]).

Among the few approaches to increase the set of instances that are feasible from the practical point of view are the *restriction-based* ones. The key idea is to recognize instances for which specific algorithms can prove much more efficient than general QBF solvers. Several tractable restrictions of $\mathcal{QBF}$ have been identified so far. [15, 16] show that quantified Krom formulas ($\mathcal{QKF}$) are polynomially solvable. [17] proved that quantified Horn formulas ($\mathcal{QHF}$) form a tractable restriction of $\mathcal{QBF}$. As an easy consequence, quantified renamable Horn formulas (ren$\mathcal{QHF}$s for short), are polynomially solvable too.

The main objective of this paper is to present a new branching heuristics for QBF solvers based on the DPLL procedure. This new heuristics aims at promoting the generation of quantified renamable Horn formulas, for which efficient solvers exist. The rest of this paper is organized as follows. Some formal preliminaries are given in Section 2. The new heuristics is presented in Section 3. Experimental results obtained by our QBF solver with this heuristics are reported in section 4. Finally, Section 5 concludes the paper and gives some perspectives.

* Thanks to anonymous reviewers for their remarks. This work has been supported in part by UIT de Lens and Université d'Artois.

F. Bacchus and T. Walsh (Eds.): SAT 2005, LNCS 3569, pp. 393–399, 2005.

2 Quantified Boolean Formulas

Let $Var(\Sigma)$ be the set of variables of the propositional formula Σ. A $\mathcal{QBF}$ formula Σ is said to be *prenex* if and only if $\Sigma = Qx_1(\ldots Qx_n(\phi)\ldots)$ where each occurrence of Q stands for either $\forall$ or $\exists$, and ϕ does not contain any quantified occurrence of a variable. ϕ is said to be the *matrix* of Σ and the sequence $Qx_1 \ldots Qx_n$ of quantifications is the *prefix* of Σ. A formula is said to be *closed* if and only if it has no free variable. In this paper, we always consider a $\mathcal{QBF}$ in prenex form with a $\mathcal{CNF}$ matrix. We focus in this paper on two restrictions of $\mathcal{QBF}$s: $\mathcal{QHF}$s and ren$\mathcal{QHF}$s.

Definition 1 (Quantified Horn Formula). *Clauses of a $\mathcal{QHF}$ contain at most one positive literal.*

Our branching heuristics oriented towards the generation of ren$\mathcal{QHF}$s is based on Hébrard's recognition of renamable Horn formulas [18]. Before presenting it, we first need to recall some notions introduced by Hébrard. Let Σ be a $\mathcal{QBF}$ and let L be a set of literals. L is *consistent* if it does not contain p and $\neg p$ for every propositional variable p. L is *complete* if p belongs to L or $\neg p$ belongs to L, for all p in $Var(\Sigma)$. A *renaming* R is a complete and consistent set of literals. R is a Horn renaming for Σ if a $\mathcal{QHF}$ formula is obtained when replacing in the matrix of Σ every occurrence of a literal by its complementary literal if the negative literal belongs to R.

Definition 2 ($\Rightarrow$, $\Rightarrow^*$ and Clos(l)).
Let l and t be two literals and Σ a $\mathcal{CNF}$ formula being the matrix of a $\mathcal{QBF}$.

- *$l \Rightarrow t$ iff $\exists$ a clause $C \in \Sigma$ s.t. $l \in C$, $\neg t \in C$ and $l \neq \neg t$.*
- *The reflexive transitive closure of $\Rightarrow$ is denoted by $\Rightarrow^*$.*
- *Clos(l) denotes the set $\{t \mid l \Rightarrow^* t\}$.*

A set of literals L is said to be *closed* if Clos(l) $\subseteq L$, $\forall l \in L$.

Proposition 1. *A renaming R is a Horn renaming if and only if it is closed, i.e. $\forall l \in R$, Clos(l) $\subseteq R$(Proposition 1.1 from [18]).*

A sketch of Hébrard's variables renaming algorithm is presented in algorithm 1.1.

3 A New Branching Heuristics

The idea of our heuristics is to branch first on variables whose propagation leads to a renamable Horn formula, or a formula that is "almost" renamable Horn. We use Hébrard's algorithm to detect renamable Horn formulas. This algorithm computes a set of literals to be renamed in order to obtain a Horn formula. If such a set exists, then we can use Kleïne-Büning polynomial time algorithm [17] in order to solve the instance. Otherwise, we use the algorithm to measure how far we are from a ren$\mathcal{QHF}$: the greater the measure is, the closer we are from a ren$\mathcal{QHF}$. We call that measure the *Contradiction's distance*.

Algorithme 1.1: Hébrard's Horn renaming algorithm (on the left) and measure of the contradiction's distance δ (on the right)

```
function RHClos
Input : a QBF Q
Output : ∅ if Q is not Horn renamable,
         else a Horn renaming R for Q.
begin
    R ← ∅ ;
    foreach variable p in Var(Q) do
        if p ∉ R and ¬p ∉ R then
            if ∀q ∈ Clos(p)\R, ¬q ∉ Clos(p)\R then
                # Clos(p)\R is consistent
                R ← R ∪ (Clos(p)\R) ;
            else if ∀q ∈ Clos(¬p)\R, ¬q ∉
            Clos(¬p)\R then
                # Clos(¬p)\R is consistent
                R ← R ∪ (Clos(¬p)\R) ;
            else return ∅ ;
end
```

```
function δ
Input : a QBF Q, a litral l
Output : the distance δ of l to a Horn renaming.
begin
    Closl ← {l} ;
    Distance ← 0 ;
    Put(lQueue, l) ;
    while not EmptyQueue(lQueue) do
        m ← Get(lQueue) ;
        LeadsTo ← {¬t|m ⇒ t} ;
        foreach e ∈ LeadsTo do
            if ¬e ∈ Closl then
                return Distance + 1 ;
            else if e ∈ Closl then
                Closl ← Closl ∪ {e} ;
                Put(lQueue, e) ;
        Distance ← Distance + 1 ;
    return Distance + 1 ;
end
```

Definition 3 (Contradiction's distance). *The distance from a contradiction of Horn renamability δ_l for a literal l in a QBF Q is defined as follows:*

$$\delta_l = \begin{cases} 0 \ if \ \nexists v | l \Rightarrow v \\ 1 \ if \ l \Rightarrow t \ and \ l \Rightarrow \neg t \\ 1 + min(\delta_v | l \Rightarrow v) \ otherwise. \end{cases}$$

The idea behind that distance is that the closer we are to a renamable Horn formula, the greater the contradiction distance should be.

In order to compute that distance, we need to slightly modify Hébrard's algorithm: Hébrard's algorithm performs DFS (Depth-First Search) while ours use BFS (Breadth-First Search). This is mandatory to ensure that the value returned by the algorithm is minimal. The right part of the algorithm 1.1 describes such a computation. It's in linear complexity ($O(n+m)$, with n the number of literals and m the number of clauses, considering that size of the clauses is bounded by a constant).

This distance is often not sufficient to elect a single variable. As a consequence, we refine it using the well-known two-sided Jeroslow-Wang heuristics [19] with a setting widely used in complete solvers for random k-SAT inherited from POSIT [20]. We restrict the selection of the variables to the outermost quantifier scope and we branch first on the literal having the greatest contradiction distance.

Definition 4 (Heuristics Δ of Horn Renamability). *Let X_1 be the outermost quantifier group. Our heuristic function Δ is as follows:*

- $\Delta_x = 1024 \times \delta_x \times \delta_{\neg x} + \delta_x + \delta_{\neg x}$;
- *the variable x is chosen if $x \in X_1$ and, $\forall y \in X_1$, $(x \neq y \Rightarrow \Delta_x \leq \Delta_y)$;*
- *the literal x is finally chosen if $\delta_x > \delta_{\neg x}$, $\neg x$ otherwise.*

4 Experimental Results

4.1 Methodology

The empirical results presented in this section have been obtained on PIV 3GHz computers with 512MB of RAM. In order to compare Qbfl (version 1.7) with other solvers, we chose to use QuBE-Rel[1] (version 1.3) and Semprop[2] (version 010604) for those experiments since those two solvers are two state-of-the-art QBF solvers according to the recent $\mathcal{QBF}$ evaluations[3] [21] that are freely available. In the following tables, only the CPU time of instances solved is taken into account. As a result, it is necessary to also check the number of benchmarks solved in order to compare the behavior of the solvers.

4.2 "Polynomial" Benchmarks

Our first experiments are on sets of randomly generated $\mathcal{QHF}$ and ren$\mathcal{QHF}$ benchmarks[4] The instances are generated as follows:

$\mathcal{QHF}$ for each clause, we randomly choose the size of the clause. Then, a literal is chosen to be the positive one in the clause. We complete it by randomly choosing negative literals;

ren$\mathcal{QHF}$ we first chose the number of variables to be renamed then those variables are chosen before generating clauses using the same method as the one used for generating $\mathcal{QHF}$s but renaming the chosen literals.

We chose to use only two alternations of quantifiers, $\forall X \exists Y$, such that $|X| > \alpha$ with $2 < \alpha < 2/3 \times \#V$ with $\#V$ the number of variables of the formula.

Those benchmarks are sorted by groups of 1000 (resp. 100) instances of $\mathcal{QHF}$s (resp. ren$\mathcal{QHF}$s). Each instance has 400 variables and we increase the usual clauses over variables ratio from 3 to 6. The timeout is fixed at 60 seconds. Note that this timeout is 3 orders of magnitude greater than the CPU time needed by our dedicated solver for those benchmarks and that similar results were obtained using a timeout of 300s.

Figure 1 reports empirical comparisons on ren$\mathcal{QHF}$s. The behavior of the solvers is quite similar to the $\mathcal{QHF}$case. However, those benchmarks look more difficult for **Semprop** which fails on 10% of them.

Note that **Qbfl** does not appear on the graph on the right hand side because our solver solved all those benchmarks.

We can observe that the runtime used by **Qbfl** to solve those polynomial formulas increases slowly and regularly. Those first experiments do not show that our approach is useful: They only show that our solver can detect and solve $\mathcal{QHF}$ and ren$\mathcal{QHF}$ formulas quite efficiently and that those formulas are not trivial for current state-of-the-art QBF solvers. We submitted to the 2005 $\mathcal{QBF}$

[1] http://www.star.dist.unige.it/~qube.
[2] http://www4.in.tum.de/~letz/semprop.
[3] Some results of the evaluations are available on http://www.qbflib.org/qbfeval.
[4] Generators are available at http://www.cril.uni-artois.fr/~letombe/qbfg.

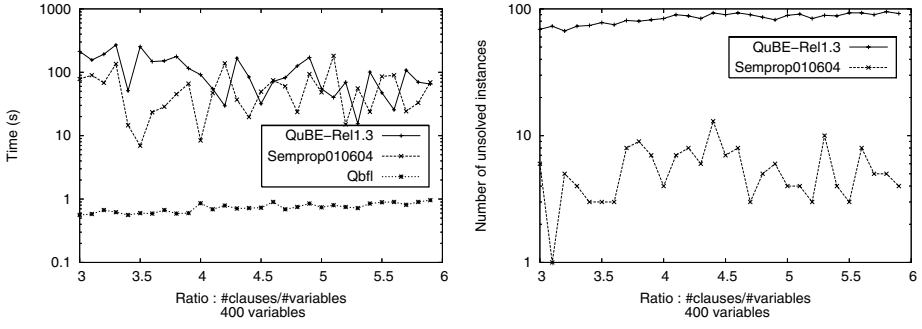

Fig. 1. Comparison between Semprop, QuBE and Qbfl on sets of 100 renamable Horn random instances

Evaluation the benchmarks on which **Semprop** failed. It is more interesting to see how our solver behave on last year $\mathcal{QBF}$ evaluation set of benchmarks.

4.3 QBF 2004 Evaluation Benchmarks

We focus first on formulas proposed by G. Pan, translated from modal logic K from TANCS'98 [22]. Those 378 instances - 9 types of 21 benchmarks valid and 9 not valid - have been proposed for 2003 $\mathcal{QBF}$ solvers evaluation and most of them were classified as *hard*[5]. Since 2003, solvers have improved and in the 2004 $\mathcal{QBF}$ evaluation classification, those benchmarks appear a bit easier.

Results obtained by **Qbfl** on those instances are sketched in Table 1. For each type of instance, the ratio of instances solved over all instances are shown in column "%solved" and the ratio of renamable Horn formulas reached over all checks performed are shown in the column "%RH". Those data are presented for both **Qbfl** with Jeroslow-Wang's and Δ heuristics.

The main observation is that really much more (about twice) benchmarks are solved by our solver using the heuristic Δ compared with Jeroslow-Wang's. The result is impressive, especially for *k_grz_n* instances: 30 benchmarks are solved with Jeroslow-Wang's heuristic while up to 59 are solved with Δ. Unfortunately, it is difficult to know if that good behavior is related to the renamable Horn detection or if the change of heuristics is alone responsible of it. Indeed, we have no way to compare the number of renamable Horn formulas found during the search by the two heuristics: reducing the search space also reduces the number of renamable Horn formulas found.

We tried Δ on some other types of formulas. In the crowd of existing benchmarks, we tried all instances proposed by A. Ayari (72), C. Castellini (169), M. Mneimneh and K. A. Sakallah (202), J. Rintanen (67) and C. Scholl and B. Becker (64)[6]. The main observation is that Δ does not alter significantly the

[5] Hard instances of this evaluation are those solved by one solver or even none.
[6] All those benchmarks are available on http://www.qbflib.org.

Table 1. Percents of benchmarks solved and renamable Horn instances reached

Instance type	Jeroslow-Wang %solved	Jeroslow-Wang %RH	Renamable Horn Δ %solved	Renamable Horn Δ %RH
k_branch_n	4.76	**25.26**	**9.52**	9.29
k_d4_n	4.76	**13.25**	4.76	5.63
k_dum_n	4.76	**11.69**	**23.80**	11.51
k_grz_n	0	-	**61.90**	**11.94**
k_lin_n	9.52	20.00	9.52	**24.26**
k_path_n	9.52	5.47	**14.28**	**12.12**
k_ph_n	23.80	2.66	23.80	**11.73**
k_poly_n	9.52	0	**14.28**	**6.66**
k_t4p_n	4.76	11.37	4.76	**26.62**
Total valid	7.93	11.21	**18.51**	**13.30**

Instance type	Jeroslow-Wang %solved	Jeroslow-Wang %RH	Renamable Horn Δ %solved	Renamable Horn Δ %RH
k_branch_p	4.76	**24.56**	4.76	9.33
k_d4_p	9.52	**26.33**	**14.28**	25.34
k_dum_p	4.76	**11.71**	**14.28**	11.40
k_grz_p	0	-	0	-
k_lin_p	9.52	**11.88**	**19.04**	8.65
k_path_p	14.28	7.43	**19.04**	**7.64**
k_ph_p	19.04	**10.06**	19.04	6.89
k_poly_p	4.76	**12.91**	**9.52**	11.04
k_t4p_p	0	-	**4.76**	**26.02**
Total false	7.40	**14.98**	**11.64**	13.28

Instance type	Jeroslow-Wang %solved	Jeroslow-Wang %RH	Renamable Horn Δ %solved	Renamable Horn Δ %RH
Total valid	7.93	11.21	**18.51**	**13.30**
Total false	7.40	**14.98**	**11.64**	13.28
Total	7.67	13.09	**15.07**	**13.29**

behavior of **Qbfl** without it: Among the 283 instances solved using the Jeroslow-Wang heuristics, only 14 instances were not solved using Δ. Taking into account the Pan benchmarks, **Qbfl** equipped with Δ perform better than without it.

5 Conclusion

Our work focuses on the practical use of tractable restrictions of $\mathcal{QBF}$ in solvers. We proposed a heuristics based on Hébrard's algorithm to detect Horn renamability. We noticed that in practice current QBF solvers are not able to solve in reasonable time some instances of $\mathcal{QHF}$ or ren$\mathcal{QHF}$. Quantor[23], which is a solver somehow different from the other QBF solvers, performs also badly on those benchmarks.

In contrast, we tried some classical SAT solvers on the very same benchmarks (without the prefix): They can solve them easily. An explanation is that the extra complexity in the QBF case is due to the constraint on the prefix that most solvers try to satisfy first. We are currently investigating the behavior of Audemard and Saïs solver [14] on those benchmarks: Their solver considers last that constraint. Unfortunately, first experimental results show that even this solver can't easily solve those instances.

We are waiting for the results of the 2005 $\mathcal{QBF}$ evaluation to gather additional information that could help us to explain the behavior of the tested solvers.

References

1. Egly, U., Eiter, T., Tompits, H., Woltran, S.: Solving advanced reasoning tasks using quantified boolean formulas. In: AAAI'00, Austin (USA) (2000) 417–422
2. Fargier, H., Lang, J., Marquis, P.: Propositional Logic and One-stage Decision Making. In: KR'00, Breckenridge (CO) (2000) 445–456

3. Besnard, P., Schaub, T., Tompits, H., Woltran, S.: Paraconsistent Reasoning via Quantified Boolean Formulas, I: Axiomatising Signed Systems. In: JELIA'02, Cosenza (Italy) (2002) 320–331
4. Rintanen, J.: Constructing Conditional Plans by a Theorem-Prover. Journal of Artificial Intelligence Research **10** (1999) 323–352
5. Pan, G., Sattler, U., Vardi, M.Y.: BDD-Based Decision Procedures for K. In: CADE'02, Copenhagen (Denmark) (2002) 16–30
6. Cadoli, M., Giovanardi, A., Schaerf, M.: An algorithm to Evaluate Quantified Boolean Formulae. In: AAAI'98, Madison (USA) (1998) 262–267
7. Rintanen, J.: Improvements to the Evaluation of Quantified Boolean Formulae. In: IJCAI'99, Stockholm (Sweden) (1999) 1192–1197
8. Feldmann, R., Monien, B., Schamberger, S.: A distributed algorithm to evaluate quantified boolean formula. In: AAAI'00, Austin (USA) (2000) 285–290
9. Rintanen, J.: Partial implicit unfolding in the Davis-Putnam procedure for Quantified Boolean Formulae. In: QBF'01, Siena (Italy) (2001) 84–93
10. Giunchiglia, E., Narizzano, M., Tacchella, A.: Backjumping for Quantified Boolean Logic Satisfiability. In: IJCAI'01, Seattle (USA) (2001) 275–281
11. Letz, R.: Lemma and Model Caching in Decision Procedures for Quantified Boolean Formulas. In: TABLEAUX'02, Copenhagen (Denmark) (2002)
12. Zhang, L., Malik, S.: Towards a symetric treatment of satisfaction and conflicts in quantified boolean formula evaluation. In: CP'02, Ithaca (USA) (2002) 200–215
13. Pan, G., Vardi, M.: Optimizing a BDD-Based Modal Solver. In: CADE'03, Miami Beach (USA) (2003) 75–89
14. Audemard, G., Saïs, L.: SAT based BDD solver for Quantified Boolean Formulas. In: ICTAI'04, Boca Raton (USA) (2004) 82–89
15. Aspvall, B., Plass, M., Tarjan, R.: A linear-time algorithm for testing the truth of certain quantified boolean formulas. Information Processing Letters **8** (1979) 121–123 Erratum: Information Processing Letters 14(4): 195 (1982).
16. Gent, I.P., Rowley, A.: Solving 2-CNF Quantified Boolean Formulae using Variable Assignment and Propagation. In: QBF'02, Cincinnati (USA) (2002) 17–25
17. Kleine-Büning, H., Karpinski, M., Flögel, A.: Resolution for quantified boolean formulas. Information and Computation **117** (1995) 12–18
18. Hébrard, J.J.: A Linear Algorithm for Renaming a Set of Clauses as a Horn Set. In: Theoretical Computer Science. Volume 124. (1994) 343–350
19. Jeroslow, R.J., Wang, J.: Solving propositional satisfiability problems. Annals of Mathematics and Artificial Intelligence **1** (1990) 167–188
20. Freemann, J.W.: Improvement to Propositional Satisfiability Search Algorithms. PhD thesis, University of Pennsylvania (1995)
21. Le Berre, D., Simon, L., Tacchella, A.: Challenges in the QBF Arena: the SAT'03 evaluation of QBF solvers. In: QBF'03, S. Margherita (Italy) (2003) 468–485
22. Balsiger, P., Heuerding, A., Schwendimann, S.: A benchmark method for the propositional modal logics K, KT, S4. Automated Reasoning **24** (2000) 297–317
23. Biere, A.: Resolve and Expand. In: SAT'04, Vancouver (Canada) (2004) 238–246