GhostQ-PG QBF Solver Description (2017)

William Klieber

Carnegie Mellon University

1 Overview

GhostQ [3] has not changed much since 2012, when CEGAR learning [2] was added. A new feature for 2017 is an enhancement to the preprocessor to have some support for the Plaisted-Greenbaum encoding. Three configurations of GhostQ have been submitted to QBFEVAL'17:

1. `ghostq-pg-cegar`: Plaisted-Greenbaum and CEGAR learning.
2. `ghostq-pg-plain`: Plaisted-Greenbaum, but without CEGAR learning.
3. `ghostq-cegar`: Unchanged from the 2016 version (no Plaisted-Greenbaum).

2 Reverse Engineering for Plaisted-Greenbaum (cf. [1])

In the Tseitin encoding, the gate definition $g = x_1 \vee ... \vee x_n$ is encoded by the following clauses: $(\neg g \vee x_1 \vee ... \vee x_n)$, $(g \vee \neg x_1)$, $\ldots$, $(g \vee \neg x_n)$. During unit propagation, the binary clauses (i.e., clauses with exactly two literals) will force $g = \texttt{true}$ if any x_i becomes true. The single non-binary clause will force $g = \texttt{false}$ if all the x_i become false. The Plaisted-Greenbaum encoding for g may omit either all the binary clauses or the single non-binary clause, depending on how g occurs in the formula. (If g occurs both positively and negatively, then no clauses can be omitted.)

Conventions: A double negation of a variable is considered equivalent to the variable itself. The order of literals in a clause is immaterial; the clause $(x \vee y)$ is considered equivalent to $(y \vee x)$. Given a variable v, $\text{var}(\neg v) = v = \text{var}(v)$.

Consider a QBF formula Φ of the form $P.\phi$ where P is the quantifier prefix and ϕ is a conjunction of clauses and equivalences[1]. Consider a literal g, where $\text{var}(g)$ is existentially quantified in the innermost quantification block. We define *binary half-def* and *non-binary half-def* as follows:

- If (1) g does not occur (as a disjunct[2]) in any non-binary clause and (2) the set of clauses in which g occurs (as a disjunct) is $\{(g \vee \neg x_1), ..., (g \vee \neg x_n)\}$, then $(g \vee x_1 \vee ... \vee x_n)$ is a *binary half-def* of g in Φ.
- If $\neg g$ occurs (as a disjunct) in exactly one clause, and that clause is a non-binary clause $(\neg g \vee x_1 \vee ... \vee x_n)$, then $g \wedge (x_1 \vee ... \vee x_n)$ is a *non-binary half-def* of g in Φ.

[1] Motivation: We start with a formula in CNF. When we discover clauses that constitute a gate definition, we delete the clauses and insert an equivalence for the gate definition. E.g., $(x \vee y) \wedge (\neg x \vee \neg y) \wedge C_3 \wedge ... \wedge C_n$ might become $(x \Leftrightarrow \neg y) \wedge C_3 \wedge ... \wedge C_n$.

[2] E.g., x doesn't occur as a disjunct in the clause $(\neg x \vee y)$, but $\neg x$ does.

Notation: Given two formulas ϕ and f and a variable v, let "$\phi[v \to f]$" denote the result of taking ϕ and substituting all occurrences of v with f. Given a negative literal $\ell = \neg v$, let "$\phi[\ell \to f]$" denote $\phi[v \to \neg f]$.

Notation: Given a formula ϕ and an assignment $\pi = \{x_1{:}c_1, ..., x_n{:}c_n\}$, let "$\phi|_\pi$" denote the substitution of π in ϕ, i.e., $\phi|_\pi = \phi[x_1 \to c_1][x_2 \to c_2] \cdots [x_n \to c_n]$.

Lemma 1. If a formula f of the form $(g \vee x_1 \vee ... \vee x_n)$ is a binary half-def of a literal g in $P.\,\phi$, then $P.\,\phi[g \to f]$ has the same truth value as $P.\,\phi$.
Proof. Let us write "$\exists g$" as an abbreviation of "$\exists\, \mathrm{var}(g)$". Since $\mathrm{var}(g)$ is quantifed innermost and existentially, it suffices to prove the following: For every assignment π to all variables in ϕ except g, $\exists g.\, \phi[g \to f]|_\pi = \exists g.\, \phi|_\pi$. Consider such an assignment π. There are two cases:

1. If $(x_1 \vee ... \vee x_n)|_\pi = \texttt{false}$, then $f|_\pi = g|_\pi$, and thus $\phi[g \to f]|_\pi = \phi|_\pi$.
2. If $(x_1 \vee ... \vee x_n)|_\pi = \texttt{true}$, then:
   (a) $\exists g.\, \phi|_\pi \quad = \phi|_{\pi \cup \{g\,:\,\texttt{false}\}} \vee \phi|_{\pi \cup \{g\,:\,\texttt{true}\}}$ (by def of "$\exists$")
   (b) $\phi|_{\pi \cup \{g:\texttt{false}\}} = \texttt{false}$ because at least one of the binary clauses with g is false under $\pi \cup \{g : \texttt{false}\}$.
   (c) $\exists g.\, \phi|_\pi \quad = \phi[g \to \texttt{true}]|_\pi$ (follows from the above two steps)
   (d) $\exists g.\, \phi[g \to f]|_\pi \quad = \phi[g \to \texttt{true}]|_\pi$, because $f|_\pi = \texttt{true}$

Lemma 2. If f is a non-binary half-def of a literal g in $P.\,\phi$, then $P.\,\phi[g \to f]$ has the same truth value as $P.\,\phi$. **Proof.** Similar to proof of Lemma 1 above.

Internally, the GhostQ preprocessor maintains a list of clauses C and a hashtable `GateDef` that maps gate variables to their definitions.

Definition. A half-def f of g is a *definite* half-def iff g is not already defined in `GateDef` and either (1) f is the only half-def of g and there are no half-defs of $\neg g$ or (2) the only innermost-quantified variable in f without a gate def is g.

The GhostQ preprocessor handles Plaisted-Greenbaum roughly as follows:

- Repeat until fixed point:
  - For each literal g that has a definite half-def f:
    - If the below actions won't cause a cycle in the gate defs, then:
      - Delete the binary clauses or single non-binary clause associated with the half-def.
      - Let h be a fresh variable. Replace all occurrences of g with h in both the clauses C and the gate definitions.
      - Add a gate definition: `GateDef[`h`]` $= f$

References

1. A. Goultiaeva and F. Bacchus. Recovering and utilizing partial duality in QBF. In *SAT 2013*.
2. M. Janota, W. Klieber, J. Marques-Silva, and E. Clarke. Solving QBF with Counterexample Guided Refinement. In *SAT 2012*.
3. W. Klieber, S. Sapra, S. Gao, and E. M. Clarke. A Non-prenex, Non-clausal QBF Solver with Game-State Learning. In *SAT 2010*.