ON CONFLICTS AND STRATEGIES IN QBF

Nikolaj Bjørner\textsuperscript{1}  \hspace{1em} Mikoláš Janota\textsuperscript{2}  \hspace{1em} William Klieber\textsuperscript{3}

LPAR-20 2015, Suva, Fiji

\textsuperscript{1}Microsoft Research, Redmond, USA
\textsuperscript{2}Microsoft Research, Cambridge, UK
\textsuperscript{3}CERT/SEI, Carnegie Mellon University
QUANTIFIED BOOLEAN FORMULA (QBF)

- an extension of SAT with quantifiers
• an extension of SAT with quantifiers

Example

$$\forall y_1 y_2 \exists x_1 x_2. (\neg y_1 \lor x_1) \land (y_2 \lor \neg x_2)$$
• an extension of SAT with quantifiers

Example

\[ \forall y_1 y_2 \exists x_1 x_2. (\neg y_1 \lor x_1) \land (y_2 \lor \neg x_2) \]

• we consider \textit{prenex} form with \textit{maximal blocks} of variables

\[ \forall u_1 \exists e_2 \ldots \forall u_{2N-1} \exists e_{2N}. \phi \]
RELATION TO TWO-PLAYER GAMES

- A QBF represents a two-player games between $\exists$ and $\forall$.
A QBF represents a two-player games between $\exists$ and $\forall$.

$\exists$ wins a game if the matrix becomes true.

$\forall$ wins a game if the matrix becomes false.
A QBF represents a two-player games between $\exists$ and $\forall$.

$\exists$ wins a game if the matrix becomes true.

$\forall$ wins a game if the matrix becomes false.
RELATION TO TWO-PLAYER GAMES

- A QBF represents a two-player games between \(\exists\) and \(\forall\).
- \(\exists\) wins a game if the matrix becomes true.
- \(\forall\) wins a game if the matrix becomes false.
- A QBF is true iff there exists a winning strategy for \(\exists\).
RELATION TO TWO-PLAYER GAMES

- A QBF represents a two-player games between $\exists$ and $\forall$.
- $\exists$ wins a game if the matrix becomes true.
- $\forall$ wins a game if the matrix becomes false.
- A QBF is true iff there exists a winning strategy for $\exists$.
- A QBF is false iff there exists a winning strategy for $\forall$. 
RELATION TO TWO-PLAYER GAMES

- A QBF represents a two-player game between $\exists$ and $\forall$.
- $\exists$ wins a game if the matrix becomes true.
- $\forall$ wins a game if the matrix becomes false.
- A QBF is true iff there exists a winning strategy for $\exists$.
- A QBF is false iff there exists a winning strategy for $\forall$.

Example

$$\forall u \exists e. (u \lor e) \land (\neg u \lor \neg e)$$
A QBF represents a two-player games between $\exists$ and $\forall$.
$\exists$ wins a game if the matrix becomes true.
$\forall$ wins a game if the matrix becomes false.
A QBF is true iff there exists a winning strategy for $\exists$.
A QBF is false iff there exists a winning strategy for $\forall$.

Example

$$\forall u \exists e. (u \lor e) \land (\neg u \lor \neg e)$$

$\exists$ wins by playing $e \leftarrow \neg u$. 
Two types of solving: expansion [Janota et al., 2012] and conflict-driven (DPLL) [Zhang and Malik, 2002].

Both approaches have their weaknesses [Beyersdorff et al., 2015].

What is a good way of combining them?
Consider $\forall x. \phi$

- traditional expansion [Janota et al., 2012]
  abstract as: $\bigwedge_{c \in \omega} \phi|_{x=c}$ where $\omega \subseteq \{0, 1\}$
- strategy expansion
  abstract as: $\bigwedge_{f \in \omega} \phi|_{x=f}$ where $\omega \subseteq \text{dom}(X) \rightarrow \mathbb{B}$

E.g. $\exists e \forall u. (u \leftrightarrow e)$, expand with $u \triangleq \neg e$. Simplifies to false.
WHERE TO GET STRATEGIES?

Abstraction → give decisions → Propagation

Abstraction → refine if conflict → Propagation

Propagation → cont. if ok → Abstraction
(In Q-Resolution)

\[
\begin{align*}
C \lor u \in \phi & \quad \Rightarrow \quad \frac{C \lor u \lor (u \leftarrow 0)}{C \lor u : (u \leftarrow 0)} \\
C \lor \bar{u} \in \phi & \quad \Rightarrow \quad \frac{C \lor \bar{u} \lor (u \leftarrow 1)}{C \lor \bar{u} : (u \leftarrow 1)} \\
C \in \phi & \quad \Rightarrow \quad \frac{C \lor (u \leftarrow \star)}{C : (u \leftarrow \star)} \quad u, \bar{u} \notin C \\
C_1 \lor x : (u \leftarrow f_1) & \quad C_2 \lor \bar{x} : (u \leftarrow f_2) \\
\hline
C_1 \lor C_2 : (u \leftarrow x \oplus f_2 : f_1)
\end{align*}
\]
COMBINED QBF SOLVING

1 \( \delta \leftarrow \emptyset \) // initialization of decisions
2 \( \alpha \leftarrow [\alpha_1 = \text{true}, \ldots, \alpha_n = \text{true}] \) // initialization
3 while true do
4 (\( \tau \), loser) \( \leftarrow \) Propagate(\( \delta \), \( \Phi \)) // propagation
5 if \( \tau = \bot \) then // conflict resolution
6 (\( k \), \( \alpha \), \( \Phi \)) \( \leftarrow \) LearnAndRefine(\( \delta \), \( \Phi \), \( \alpha \), loser)
7 if \( k = \bot \) then return (loser = \( \forall \))?true : false
8 \( \delta \leftarrow \{ l \in \delta \mid \text{qlevel}(l) < k \} \) // backtrack decisions
9 else // decision-making
10 \( k \leftarrow \) minimal quantification level not fully assigned in \( \tau \)
11 \( \tau_k \leftarrow \{ l \in \tau \mid \text{qlevel}(l) \leq k \} \) // filtering
12 (\( \mu \), \( \tau' \)) \( \leftarrow \) \( \text{SAT}(\alpha_k \land \tau_k) \) // consult \( \alpha_k \)
13 if \( \mu = \bot \) then // abstraction unsatisfiable
14 \( \Phi \leftarrow \text{ResolveUnsat}(\tau', Q_k, \Phi) \) // update \( \Phi \)
15 if \( \Phi = \bot \) then return (\( Q_k = \forall \))?true : false
16 else
17 \( v \leftarrow \) a variable unassigned by \( \tau \) at quantification level \( k \)
18 \( \delta \leftarrow \delta \cup \{ \mu(v) ? v : \neg v \} \) // make a decision on \( v \)
EXPERIMENTS 2QBF

- qosta
- qosta-noref
- areqs
- depqbf
- ghostq

CPU Time (s)

number of solved instances
• key ingredient: in solving *explicit strategies* in solving
CONCLUSIONS AND FUTURE WORK

- key ingredient: in solving *explicit strategies* in solving
- construction of strategies from DPLL conflicts
CONCLUSIONS AND FUTURE WORK

- key ingredient: in solving explicit strategies in solving
- construction of strategies from DPLL conflicts
- enables combining DPLL and expansion
CONCLUSIONS AND FUTURE WORK

- key ingredient: in solving *explicit strategies* in solving
- construction of strategies from DPLL conflicts
- enables combining DPLL and expansion
- different applications of strategies? (prediction of the opponent)
CONCLUSIONS AND FUTURE WORK

• key ingredient: in solving explicit strategies in solving
• construction of strategies from DPLL conflicts
• enables combining DPLL and expansion
• different applications of strategies? (prediction of the opponent)
• better ways how to come up with strategies?
CONCLUSIONS AND FUTURE WORK

- key ingredient: in solving explicit strategies in solving
- construction of strategies from DPLL conflicts
- enables combining DPLL and expansion
- different applications of strategies? (prediction of the opponent)
- better ways how to come up with strategies?
- combining strategies?
Thank You for Your Attention!

Questions?


Zhang, L. and Malik, S. (2002). **Conflict driven learning in a quantified Boolean satisfiability solver.** In *ICCAD.*