Bounded Universal Expansion for Preprocessing QBF

Uwe Bubeck
and
Hans Kleine Büning

International Graduate School
Dynamic Intelligent Systems,
University of Paderborn

30.5.2007
Outline

• Introduction
• Bounded Expansion
• Variable Selection
• Refinements
• Results and Conclusion
Introduction
Universals and Complexity

Universals make QBF more difficult:

• must consider **both values for each universal**

• **existentials** may have different values depending on preceding universals

\[ \forall x_1 \forall x_2 \exists y (y \lor x_1 \lor x_2) \land (\neg y \lor \neg x_1) \land (\neg y \lor \neg x_2) \]

\[ \uparrow \quad \uparrow \]

\[ y \text{ depends on } x_1 \text{ and } x_2 \]

• quantifier ordering must be respected

\[ \forall x \exists y (x \lor \neg y) \land (\neg x \lor y) \neq \exists y \forall x (x \lor \neg y) \land (\neg x \lor y) \]

\[ \rightarrow \text{ So let us focus on the universals!} \]
Semantics of Quantifiers

Quantifiers are actually abbreviations:

\( \forall x \, \phi(x) \) is true if and only if 
\( \phi[x/0] \) is true \textbf{and} \( \phi[x/1] \) is true

\( \exists x \, \phi(x) \) is true if and only if 
\( \phi[x/0] \) is true \textbf{or} \( \phi[x/1] \) is true

\( \rightarrow \) we can eliminate them by expansion.
Bounded Universal Expansion

Bounded Expansion
Expansion Procedure 1/3

\[ \forall x \, \phi(x) \text{ is true if and only if} \]
\[ \phi[x/0] \text{ is true and } \phi[x/1] \text{ is true} \]

\[ \rightarrow \text{ Duplicate the matrix of the formula. } \]
\[ \text{For dominated existentials, we need two separate instances.} \]

Example:

\[ \exists y_1 \, \forall x_1 \, \forall x_2 \, \exists y_2 \, \phi(x_1, x_2, y_1, y_2) \]
\[ \approx \exists y_1 \, \forall x_2 \, \exists y_2^{(0)} \, \exists y_2^{(1)} \, \phi(0, x_2, y_1, y_2^{(0)}) \land \phi(1, x_2, y_1, y_2^{(1)}) \]
Expansion Procedure 2/3

Longer Example:

\[ \Phi(z) = \forall x_1 \exists y_1 \forall x_2 \forall x_3 \exists y_2 \phi(x_1, x_2, x_3, y_1, y_2, z) \]

\[ \approx \exists y_1^{(0)} (\exists y_2^{(0,0,0)} \phi(0, 0, 0, y_1^{(0)}, y_2^{(0,0,0)}, z) \land \exists y_2^{(0,0,1)} \phi(0, 0, 1, y_1^{(0)}, y_2^{(0,0,1)}, z) \land \exists y_2^{(0,1,0)} \phi(0, 1, 0, y_1^{(0)}, y_2^{(0,1,0)}, z) \land \exists y_2^{(0,1,1)} \phi(0, 1, 1, y_1^{(0)}, y_2^{(0,1,1)}, z)) \land \]

\[ \exists y_1^{(1)} (\exists y_2^{(1,0,0)} \phi(1, 0, 0, y_1^{(1)}, y_2^{(1,0,0)}, z) \land \exists y_2^{(1,0,1)} \phi(1, 0, 1, y_1^{(1)}, y_2^{(1,0,1)}, z) \land \exists y_2^{(1,1,0)} \phi(1, 1, 0, y_1^{(1)}, y_2^{(1,1,0)}, z) \land \exists y_2^{(1,1,1)} \phi(1, 1, 1, y_1^{(1)}, y_2^{(1,1,1)}, z)) \]

Exponential Growth!
Expansion Procedure 3/3

How to avoid exponential blowup?

Do not expand all universals!
Try to eliminate as many as possible under bounded expansion costs.
Leave expensive universals to other solvers / techniques.
Hypothesis:
we can make it easier for QBF solvers by first removing carefully selected cheap or rewarding universals.

Crucial for success:
• accurate a priori cost estimates
• suitable selection strategy
Bounded Expansion

Enforce tight bounds:

- **global size limit** $C_{\text{global}}$: $|\Phi_{\text{cur}}| \leq C_{\text{global}} \cdot |\Phi|$
  
  Suitable values: $2 \text{ – } 4$
  
  → prevents flooding solver with huge formulas

- **individual cost limit** $C_{\text{single}}$: $c_x \leq C_{\text{single}} \cdot |\Phi_{\text{cur}}|$
  
  Suitable values: $\approx 0.5$
  
  → leave expensive universals to the solver
Variable Selection
Variable Locality

\[ \forall x_1 \exists y_1 \forall x_2, x_3 \exists y_2, y_3 (\phi(x_1, y_1, x_2, y_2) \land \psi(x_1, y_1, x_3, y_3)) \]

\(x_1\) and \(y_1\) occur in the whole formula,
\(x_2 / y_2\) and \(x_3 / y_3\) are only used locally in \(\phi\) resp. \(\psi\).

→ when expanding \(x_3\), we only need to duplicate \(y_3\) and clauses in \(\psi\).

→ take into account variable connectivity:
  start with locally connected variables and compute
  the transitive closure (see Quantor [Biere ‘04])
Expansion Order 1/2

Quantor always expands from the innermost block.

\[ \forall \forall \forall \exists \exists \forall \exists \forall \forall \forall \exists \exists \phi \]

Our Hypothesis:
For a preprocessor, it might be rewarding to expand from further outside.

\[ \forall \forall \forall \exists \exists \forall \exists \forall \forall \forall \exists \exists \phi \]

Reasons:

• subsequent solver might do inner universals more efficiently
• preprocessor has more time for cost computations
• universals further outside might actually be cheaper
Expansion Order 2/2

\[ \Phi = \forall x_1 \exists y_1 \forall x_2,x_3 \exists y_2,y_3 \forall x_4,x_5 \exists y_4,y_5 (\phi \land \psi) \]

In which order to expand the universals?

Further left, we have more dependent existentials
→ right to left \( x_5, x_4, x_3, \ldots \) appears best...

... but what if \( \Phi \) can be rewritten as
\[ \Phi = \forall x_1 \exists y_1 (\forall x_2 \exists y_2 \forall x_4 \exists y_4 \phi) \land (\forall x_3 \exists y_3 \forall x_5 \exists y_5 \psi) \]?

If |\( \phi \)| < |\( \psi \)|, it is cheaper to expand \( x_4, x_2 \) before \( x_5, x_3 \)

... or what if \( x_2 \) or \( x_3 \) are particularly rewarding?

(→ goal orientation)
Scheduling of eliminations requires tight a priori estimates of expansion costs:

• duplicate clauses with dependent existentials $D_x$

• when $x = 0$:
  remove clauses with $\neg x$ and occurrences of $+x$

• when $x = 1$:
  remove clauses with $+x$ and occurrences of $\neg x$

Estimate: $c_x \leq s(D_x) - s(\neg x) - o(x) - s(x) - o(\neg x)$

In Quantor, $D_x = Y_r$ (innermost existentials).
We expand far less universals and can therefore afford to compute $D_x$. 
Cost Estimates 2/2

Only subsequent existentials propagate connectivity, universals never do.

Example:

\[ \exists y_1 \forall x_1 \exists y_2 \forall x_2 \exists y_3, y_4 \]

\[
(x_1 \lor \neg y_2) \land (y_2 \lor y_3) \land (\neg y_2 \lor \neg x_2) \land (y_4 \lor x_2) \land (\neg y_3 \lor y_1) \land (\neg y_1 \lor y_4)
\]

\[
(x_1 \lor \neg y_2) \land (y_2 \lor y_3) \land (\neg y_2 \lor \neg x_2) \land (y_4 \lor x_2) \land (\neg y_3 \lor y_1) \land (\neg y_1 \lor y_4)
\]

\[
(x_1 \lor \neg y_2) \land (y_2 \lor y_3) \land (\neg y_2 \lor \neg x_2) \land (y_4 \lor x_2) \land (\neg y_3 \lor y_1) \land (\neg y_1 \lor y_4)
\]

Only \( y_2 \) and \( y_3 \) depend on \( x_1 \), only underlined clauses are affected by expansion.
Refinements
Expanding just because it is cheap lacks foresight.

→ we need more goal orientation

Suitable goal: obtaining unit literals

\[ \Phi = \forall x \exists y_1, y_2 \ (x \lor y_1) \land (\neg y_1 \lor \neg y_2) \land y_2 \]

must be true

propagate

\[ \Phi = \forall x \exists y_1, y_2 \ (x \lor y_1) \land \neg y_1 \]
Goal Orientation 2/2

Discover new unit literals through expansion:

$$\forall x_1, x_2 \exists y_1, y_2 (x_1 \lor y_1) \land (\neg y_1 \lor y_2) \land (x_2 \lor \neg y_1 \lor \neg y_2)$$

When $x_1 = 0$, we obtain a new unit literal $y_1$

$\rightarrow$ formula collapses

But $x_2$ has a lower cost estimate!

$\rightarrow$ take into account units obtained for $x_i = 0$ or $x_i = 1$

No need to balance costs and goals:
unit propagations have direct impact on costs
We can also eliminate existential variables $y_i$ through resolution:

resolve all clauses containing positive $y_i$ with all clauses containing negative $\neg y_i$
Resolution is often problematic:
• generates large clauses
• produces excessive numbers of new clauses

Solution:
estimate resolution costs $c_y$ and only resolve if
• resolution is cheap: $c_y < C_{\text{res}} \cdot |\Phi_{\text{cur}}|$, $C_{\text{res}} \approx 0$
• resolution reduces expansion costs
Before expanding a universal $x$, resolve only on dependent existentials $y \in D_x$.

→ eliminates $y$ and its soon-to-be-created copy $y'$.

→ may save duplicating some clauses in the expansion.

Example:

let $\{(y \vee y_2), (y \vee \neg y_3), (\neg y \vee y_4)\} \subseteq \phi$

and $y, y_2 \in D_x$, but $y_3, y_4 \not\in D_x$.

resolvents: $\{(y_2 \vee y_4), (\neg y_3 \vee y_4)\}$

Cost estimate ($\delta$: assumed rate of duplication):

$$c_{y|x} \approx (1 + \delta) \cdot c_y - (1 - \delta) \cdot (s(y) + s(\neg y))$$
Resolution can also cut dependencies:

\[ x \quad y \quad y \text{ connects } x \text{ to the existentials in } D_x^{"} \quad y \quad D_x^{"} \]

\[ \neg y \quad \text{no dependent existentials} \]
→ Expansion of $x$ does not need to duplicate $y$ when one phase of $y$ only occurs in clauses with variables $v \notin D_x$ and $v \neq x$. 
Quantifier Shifting 1/2

After expanding a universal $x_i$, we can reorder the prefix by moving existentials $y_j \in D_{x_i}$:

\[
\forall x_1 \exists y_1 \forall x_2 \exists y_2 \exists y_3 \forall x_3 \exists y_4 \forall x_4 \exists y_5
\]

\[
\forall x_1 \exists y_1 \forall x_2 \exists y_2 \exists y_3 \exists y_4^{(0)} \exists y_4^{(1)} \forall x_4 \exists y_5^{(0)} \exists y_5^{(1)}
\]

Duplicate dependent existentials in situ

Assume $y_4, y_4' \notin D_{x_2}$

\[
\forall x_1 \exists y_1 \exists y_4^{(0)} \exists y_4^{(1)} \forall x_2 \exists y_2 \exists y_3 \forall x_4 \exists y_5^{(0)} \exists y_5^{(1)}
\]
Quantifier Shifting Strategy:
Given an existential $y_j$, if the next universal $x_i$ further outside does not dominate $y_j \not\in D_{x_i}$, move $y_j$ in front of $x_i$.

Information about variable dependencies already available for universal expansion $\rightarrow$ solvers can benefit from these calculations.
Results and Conclusion
Implementation

- written in **Java** on top of our logic framework ProverBox
  
  [http://www.ub-net.de/cms/proverbox.html]

- can perform **standard simplifications**:
  - unit propagation
  - pure literal detection
  - universal reduction
  - dual binary clause detection
  - subsumption (forward only initially)

- **fully automatic** with reasonable default parameters
Experiments

Chosen Parameters:

• Global size limit $C_{\text{global}} = 2$
• Individual cost limit $C_{\text{single}} = 0.5$
• Resolution cost limit $C_{\text{res}} = 0.05$

Experiment Setup:

• state-of-the-art solvers $sKizzo$ and $SQBF$
• 12 benchmark families from the QBFLIB, in total 688 instances
• time limit 300 sec. for each instance, give-up mode
Results - sKizzo 1/2

Increase in number of solved problems:
Speedup (including preprocessing time):

- Adder: 1.0
- ASP: 1.0
- Blocks: 6.3
- Connect3: 0.4
- Counter: 1.5
- CounterFactul: 1.4
- Evader-Pursuer: 1.0
- k_branch_n: 2.0
- k_path_n: 5.9
- RobotsD2: 52.2
- Sortnet: 1.0
- Szymanski: 0.8
- Total: 2.1
Results - SQBF 1/2

Increase in number of solved problems:
Results - SQBF 2/2

Speedup (including preprocessing time):

- Adder: 1.7
- ASP: 14.6
- Blocks: 1.0
- Connect3: 3.6
- Counter: 1.1
- CounterFactual: 4.8
- Evader-Pursuer: 0.7
- k_branch_n: 1.4
- k_path_n: 3.8
- RobotsD2: 21.7
- Sortnet: 2.0
- Szymanski: 1.0
- Total: 3.8
Preprocessing Costs

% of time spent on preprocessing (SQBF):

- Adder
- ASP
- Blocks
- Connect3
- Counter
- CounterFactual
- Evader-Pursuer
- k_branch_n
- k_path_n
- RobotsD2
- Sortnet
- Szymanski
- Total
Summary of Results:

<table>
<thead>
<tr>
<th></th>
<th>sKizzo</th>
<th>sKizzo+pre</th>
<th>SQBF</th>
<th>SQBF+pre</th>
</tr>
</thead>
<tbody>
<tr>
<td>solved</td>
<td>327</td>
<td>360</td>
<td>205</td>
<td>294</td>
</tr>
<tr>
<td>time</td>
<td>23,730</td>
<td>11,390</td>
<td>37,675</td>
<td>10,014</td>
</tr>
</tbody>
</table>

sKizzo+pre: 10% more problems in 48% of time
SQBF+pre: 43% more problems in 27% of time

Only 2 benchmark families where preprocessing decreased number of solved instances

→ boundedness lowers risk of negative effects
Conclusion

QBF Solvers indeed benefit from expanding carefully selected universal variables in a preprocessing step.

Future Work:

- other strategies for variable selection
- more sophisticated quantifier shifting