Complexity of Symbolic Verification and Decision Problems in Bit-Vector Logic

Gergely Kovásznai\textsuperscript{1}, Helmut Veith\textsuperscript{1}, Andreas Fröhlich\textsuperscript{2}, Armin Biere\textsuperscript{2}

\textsuperscript{1} Vienna University of Technology, Vienna
\url{http://forsyte.at}

\textsuperscript{2} Johannes Kepler University, Linz
\url{http://fmv.jku.at}

MFCS 2014
August 26, 2014
Budapest, Hungary
Bit-precise verification

Hardware/software verification:
- competitions: HWMCC, SV-COMP, SMT-COMP
- industry: e.g., Intel, Microsoft

Bit-level verification = \textit{bit-blasting}:
- states as tuples of Boolean variables
- problem as a Boolean circuit

Word-level verification:
- registers/variables as words = \textit{bit-vectors}
- problem as a bit-vector formula

\textit{Similar to word-level instructions in processor/source code.}
Example: QF_BV formula

$$x^{[16]} \neq y^{[16]} \land (u^{[32]} + v^{[32]} = (x^{[16]} \circ y^{[16]}) \ll 1^{[32]})$$

Standard format: SMT-LIB

*fixed set of common operations, such as bitwise, shift, arithmetic, concatenating/slicing ops*

Example in SMT-LIB v2

```
(set-logic QF_BV)
(declare-fun x () (_ BitVec 16))
(declare-fun y () (_ BitVec 16))
(declare-fun u () (_ BitVec 32))
(declare-fun v () (_ BitVec 32))
(assert (distinct x y))
(assert (= (bvadd u v) (bvshl (concat x y) (_ bv1 32))))
```
Tools

*Bit-vector solvers*: Boolector (by Armin’s group), STP, Z3 (by Microsoft), CVC, MathSAT, etc.

*Word-level model checkers*: NuSMV, Boolector

They *all* rely on *bit-blasting* = “unwrap” to bit-level circuit.

Huge industrial interest and research on *directly* working on *word-level*.

Computational complexity matters in practice? *YES*.

- What techniques to employ/adapt to attack the particular verification problem?
- How to restrict the verification problem to improve solving speed?
Complexity of $\text{QF}_\text{BV}$


Decision problem: satisfiability checking

Main factor: how *bit-widths* are encoded?

- unary form $\Rightarrow$ NP-complete
  - same as Boolean formalism (SAT also NP-complete)
- binary form $\Rightarrow$ NExpTime-complete

Exponential jump also in *hardness*!
Decision problem: safety checking

- explicit encoding $\Rightarrow$ NL-complete
  - same as graph reachability
- bit-level encoding $\Rightarrow$ PSPACE-complete
  - using Boolean formalism
- word-level encoding $\Rightarrow$ ??? (EXPSPACE-complete)
  - using bit-vector formalism with binary encoded bit-width

Exponential jumps also in *hardness*!
Using the encoding

- explicit $\Rightarrow$ $C$-complete

- Boolean/QF$_{BV}$ with *unary* encoded bit-width $\Rightarrow$ $\text{Exp}_1(C)$-complete

- QF$_{BV}$ with *binary* encoded bit-width $\Rightarrow$ $\text{Exp}_2(C)$-complete

- QF$_{BV}$ with *$\nu$-logarithmic* encoded bit-width $\Rightarrow$ $\text{Exp}_{\nu}(C)$-complete

- practical example: 3-logarithmic encoding in SMT-LIB array declarations
Examples and new results

<table>
<thead>
<tr>
<th>Encoding $\rightarrow$</th>
<th>explicit</th>
<th>Boolean circ./formula, BDD</th>
<th>unary QF_BV</th>
<th>binary QF_BV</th>
<th>$\nu$-logarithmic QF_BV</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>↓ Problem</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Word-Level MC, Reachability</td>
<td>NL</td>
<td>PSPACE</td>
<td>PSpace</td>
<td>ExpSpace</td>
<td>$(\nu - 1)$-ExpSpace</td>
</tr>
<tr>
<td>Circuit Value, Alternating Reachability</td>
<td>P</td>
<td>EXP_TIME</td>
<td>ExpTime</td>
<td>2-ExpTime</td>
<td>$\nu$-ExpTime</td>
</tr>
<tr>
<td>Clique, 3-SAT, SAT, Knapsack</td>
<td>NP</td>
<td>NEXP_TIME</td>
<td>NExpTime</td>
<td>2-NExpTime</td>
<td>$\nu$-NExpTime</td>
</tr>
<tr>
<td>$k$-QBF</td>
<td>$\Sigma_k^P$</td>
<td>$\text{NE}^\Sigma_k^P$</td>
<td>$\text{NE}^\Sigma_k^P$</td>
<td>2-$\text{NE}^\Sigma_k^P$</td>
<td>$\nu$-$\text{NE}^\Sigma_k^P$</td>
</tr>
</tbody>
</table>

- **Hardness** requires only a restricted bit-vector operator set: \( \land, \lor, \neg, =, +_1(\text{increment}) \)

- **Membership** holds for all common operators
Insight 1

- all bit-vector operators used in practice can be translated into Boolean logic easily
- closely related to circuit uniformity: operators are parametrized by bit-width \( n \)

**Definition (Log-space bit-blasting)**

A bit-vector operator allows log-space bit-blasting, if an expression over bit-vectors of bit-width \( n \) can be translated into a Boolean formula in space \( \mathcal{O}(\log n) \).
Lemma

If a problem $A$ is in complexity class $C$, and the class $\Omega$ of operators allows log-space bit-blasting, then the $\nu$-logarithmic encoded problem $\text{bv}_\nu^\Omega(A)$ is in $\text{Exp}_\nu(C)$.

We assume that $C$ is a standard complexity class such as $L$, $P$, $NP$, $PSpace$ etc.
Insight 2

Bit-vector logics are amenable to a systematic technique for hardness:

Automatically derive the hardness of a symbolic problem encoding $s(A)$ from the hardness of the explicit problem $A$.

Circuits: Galperin/Wigderson 83, Papadimitriou/Yannakakis 86, Balcazar/Lozano/Toran 96, Veith 96

Formulas: Veith 97

OBBDs: Veith 98 Verification by BDDs is PSPACE-complete.
If $A \leq B$, then $s(A) \leq s(B)$  

“Conversion Lemma”

“Lifting a reduction to symbolic representation.”

Choose an operator $\text{long}(A)$ such that

$A \leq s(\text{long}(A))$  

“Compensation Lemma”

Classical choice of $\text{long}(A)$: \{w such that $|w| \in A$\}
Assume $A$ if hard for $C$, and $\text{long}(D) \subseteq C$.

Let $B \in D$.

1. By assumption, $\text{long}(B) \in C \Rightarrow \text{long}(B) \leq A$

2. Conversion Lemma $\Rightarrow s(\text{long}(B)) \leq s(A)$

3. Compensation Lemma $\Rightarrow B \leq s(\text{long}(B))$

4. Transitivity $\Rightarrow B \leq s(A)$

5. Thus, $s(A)$ is hard for $D$. QED
Lemma (Conversion Lemma)

If $A \leq_{\text{quantifier-free}} B$, then $bv^\Omega_\nu(A) \leq_{\text{logspace}} bv^\Omega_\nu(B)$ for all $\nu$ and for all $\Omega$ containing $\wedge, \vee, \sim, =, +_1$.

Proof idea: Simulate the effect of the quantifier-free interpretation in bit-vector logic. Operators $=$ and $+_1$ make it possible to simulate equality and successor.
Lemma (Compensation Lemma)

\[ A \leq_{\logspace} \text{bv}_\nu^\Omega (\text{long}_\nu (A)) \]

for all \( \nu \) and for all \( \Omega \) containing \( =, <_u \).

**Proof idea:** Instance of \( A \) is read as a binary number \( n \), and reduced to a formula with bit-width \( \text{EXP}_\nu (n) \).
High complexity is often good news for symbolic verification.

A formalism with strong compression and good expressivity may formally increase complexity...

...but has the potential to decrease complexity \textit{in practice}. 