Bit-Vectors: Complexity and Decision Procedures

Andreas Fröhlich
Johannes Kepler University, Linz

…joined work with: Gergely Kovásznai, Armin Biere,
Helmut Veith, Christoph Wintersteiger, Youssef Hamadi

10th Alpine Verification Meeting (AVM’15)

Monday, May 4, 2015
Attersee, Austria

- QF_BV e.g. $(x^{[8]} + y^{[8]} = x^{[8]} \ll 2^{[8]}) \land (y^{[8]} * z^{[8]} = x^{[8]} | z^{[8]})$
  - Common solving approach:
    - Bit-blasting (encoding the bit-vector formula as a circuit)
    - . . . and then using a SAT-solver
  - Often assumed to be NP-complete
  - Complexity actually depends on the encoding of bit-widths
- In practice: **logarithmic encoding**, e.g. SMT-LIB format

```
(set-logic QF_BV)
(declare-fun a () (_ BitVec 1024))
(declare-fun b () (_ BitVec 1024))
(assert (distinct (bvadd a b) (bvadd b a)))
```

- QF_BV with logarithmic encoding (QF_BV$_2$) is **NEXP-complete** [KFB-SMT'12]

Propositional domain $\{0,1\}$:

- SAT $[\exists x_1, x_2, x_3 \,.]\quad (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_3) \wedge (x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2)$

  NP-complete

- QBF $\forall u_1 \exists e_1 \forall u_2 \exists e_2 \,.\quad (u_2 \vee \neg e_1) \wedge (\neg u_1 \vee e_1) \wedge (u_1 \vee \neg e_2) \wedge (\neg u_2 \vee e_2)$

  PSPACE-complete

- DQBF $\forall u_1, u_2 \exists e_1(u_1), e_2(u_2) \,.\quad (u_2 \vee \neg e_1) \wedge (\neg u_1 \vee e_1) \wedge (u_1 \vee \neg e_2) \wedge (\neg u_2 \vee e_2)$

  NEXP-complete

First-order but no functions:

- EPR $\exists a, b \forall x, y \,.\quad (p(a,x,y) \vee \neg q(y,x,b)) \wedge (q(x,b,y) \vee \neg p(y,a,x))$

  NEXP-complete

- QF_BV$_2$ is NEXP-complete [KFB-SMT'12]
  - Bit-blasting replaces logarithmic bit-widths by its unary encoding
  - Hardness by giving a reduction from DQBF to QF_BV$_2$
    - Use the so-called **binary magic numbers** to represent universal variables

$$u_0, u_1, u_2 \quad \rightarrow \quad U_0^{[8]} := \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}, U_1^{[8]} := \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}, U_2^{[8]} := \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

    - **Eliminate dependencies** by introducing constraints on shifted indices

$$e_0(u_0, u_1), e_1(u_1, u_2) \quad \rightarrow \quad E_0^{[8]} = E_0^{[8]} \ll 4^{[8]},\ E_1^{[8]} = E_1^{[8]} \ll 1^{[8]}$$

- Can be extended to **quantification** and **uninterpreted functions** [KFB-SMT'12]
  - QF_BV (quantifier-free bit-vectors)

    QF_BV$_1$ is NP-complete, QF_BV$_2$ is NEXP-complete

  - BV+UF (quantified bit-vectors with uninterpreted functions)

    BV$_1$+UF is NEXP-complete, BV$_2$+UF is 2-NEXP-complete

- Generalizations for arbitrary **complete problems** and **multi-logarithmic succinct encodings** are possible [KVFB-MFCS'14]
  - Implication: Word-Level Model Checking and Reachability for bit-vectors with binary encoded bit-widths are EXPSPACE-complete

Restricted Operator Sets

- Logarithmic case: Restrictions on the set of operators are possible [FKB-CSR'13]
  - QF_BV$_{bw}$ (only bitwise operations and equality)

    QF_BV$_{bw}$ is NP-complete
  - QF_BV$_{\ll 1}$ (only bitwise operations, equality, and left shift by one)

    QF_BV$_{\ll 1}$ is PSPACE-complete
  - QF_BV$_{\ll c}$ (only bitwise operations, equality, and left shift by constant)

    QF_BV$_{\ll c}$ is NEXP-complete
- Certain operators can be added or shown to be equally expressive [KFB-TOCS'15]

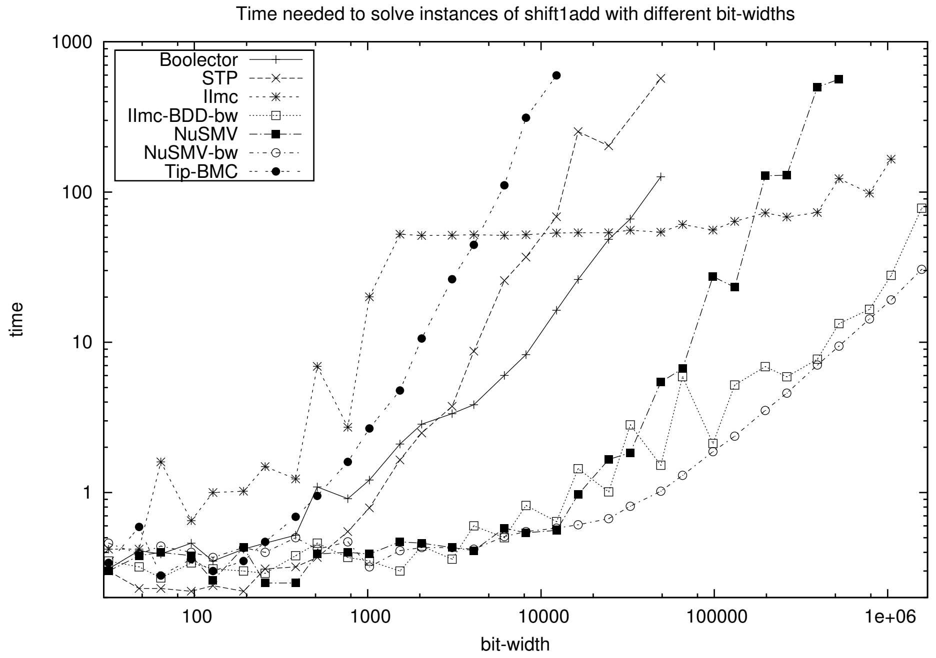
Decision Procedures for QF_BV

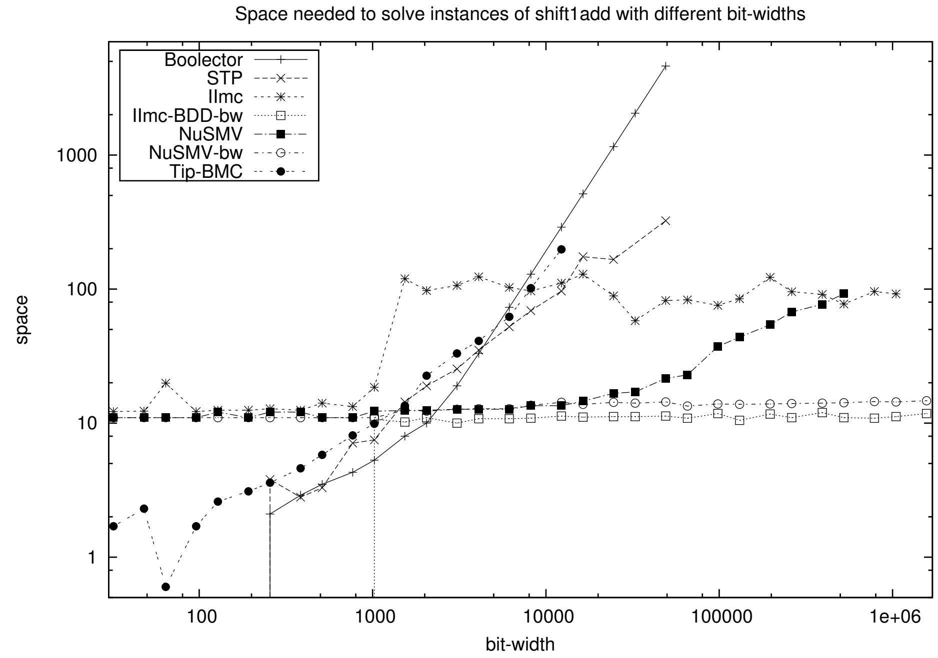
- State-of-the-art solvers for QF_BV rely on **bit-blasting** and SAT solvers
  - Bit-blasting can be exponential
  - Is it possible to solve QF_BV without bit-blasting?
  - Can we profit from knowing the complexity of certain bit-vector classes?

- Some **alternative approaches** (and optimizations) exist
  - Translation to EPR [KFB-CADE'13]
  - Translation to SMV [FKB-SMT'13]
  - Bit-width reduction [Johannsen]
  - SLS for SMT [FBWH-AAAI'15]

- BV2EPR: Polynomial translation from QF_BV to EPR [KFB-CADE'13]
- EPR formulas can be solved with iProver (by [Korovin])
  - CEGAR approach
- Performance worse than bit-blasting for most instances
  - Beneficial on some instances (0.1s instead of T/O)
  - Less memory used (can be several orders of magnitude)

- BV2SMV: Polynomial translation from QF_BV$_{\ll 1}$ to SMV [FKB-SMT'13]
- SMV formulas can be solved with model checkers
  - BDD based model checkers are most efficient





- Practical benchmarks actually do exist

- For QF_BV$_{bw}$, bit-width reduction can be applied
  - There is a solution iff there is a solution with smaller bit-width, e.g.

$$\left(X^{[32]} \neq Y^{[32]} | 3^{[32]}\right) \wedge \left(Y^{[32]} \neq Z^{[32]} \& X^{[32]}\right)$$

$$\rightarrow \qquad \left(X^{[2]} \neq Y^{[2]} | 3^{[2]}\right) \wedge \left(Y^{[2]} \neq Z^{[2]} \& X^{[2]}\right)$$

  - Can be extended to allow certain cases of other operators

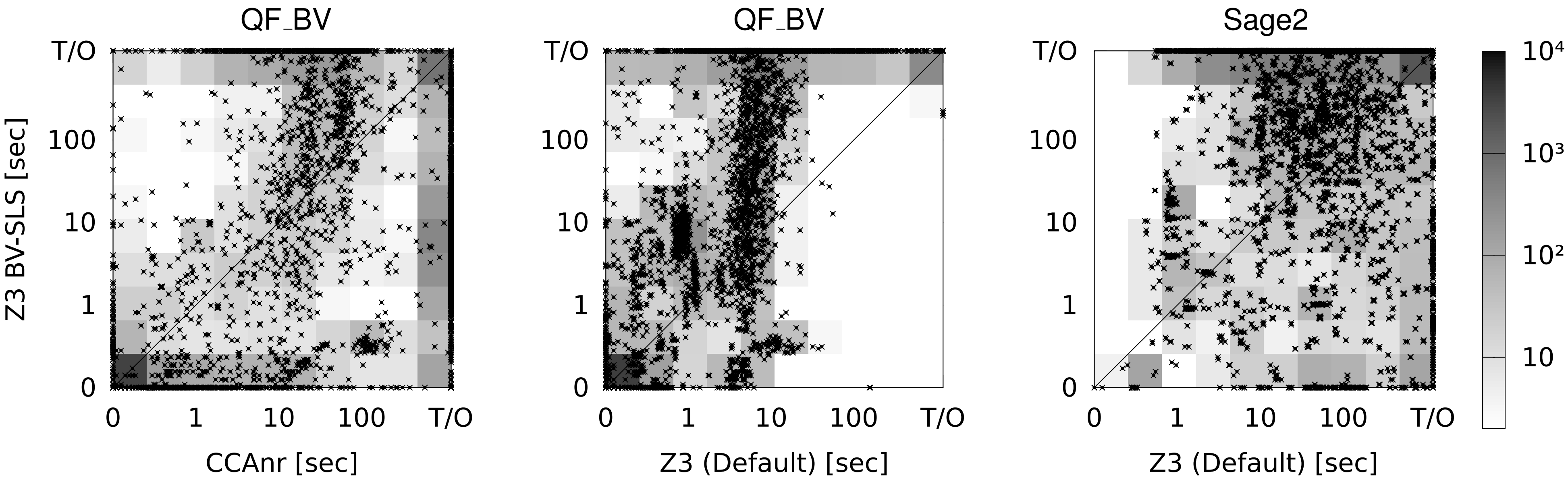
- Existing work for RTL Property Checking [Johannsen]
  - Reduces size of design model to up to 30%
  - Reduces runtimes to up to 5%

SLS for SMT (1)

- BV-SLS: Stochastic Local Search for bit-vectors [FBWH-AAAI'15]
  - No bit-blasting
  - Works on the theory representation of the formula
- Idea: Combine techniques from SAT-SLS with theory information
  - Many techniques from SAT can successfully be lifted
  - Theory information allows to deal with structure efficiently
- Promising results (see next slide)
  - Shows that SLS solvers can actually profit from structure

SLS for SMT (2)

	QF_BV	Sage2
CCAnr	**5409**	64
CCASat	4461	8
probSAT	3816	10
Sparrow	3806	12
VW2	2954	4
PAWS	3331	**143**
YalSAT	3756	142
Z3 (Default)	7173	5821
BV-SLS	**6172**	**3719**



Conclusion

- Complexity of bit-vector formulas depends ...
  - ... on the encoding of the bit-widths
  - ... on the operators we use
- Bit-blasting ...
  - ... is not polynomial in general
  - ... can profit from bit-width reduction
- Alternative approaches
  - CEGAR approach using iProver
  - Model checkers for PSPACE fragments
  - Stochastic local search on the theory level

- Gergely Kovásznai, Andreas Fröhlich, Armin Biere. On the Complexity of Fixed-Size Bit-Vector Logics with Binary Encoded Bit-Width. [KFB-SMT'12]
- Gergely Kovásznai, Andreas Fröhlich, Armin Biere. BV2EPR: A Tool for Polynomially Translating Quantifier-free Bit-Vector Formulas into EPR. [KFB-CADE'13]
- Andreas Fröhlich, Gergely Kovásznai, Armin Biere. More on the Complexity of Quantifier-Free Fixed-Size Bit-Vector Logics with Binary Encoding. [FKB-CSR'13]
- Andreas Fröhlich, Gergely Kovásznai, Armin Biere. Efficiently Solving Bit-Vector Problems Using Model Checkers. [FKB-SMT'13]
- Gergely Kovásznai, Helmut Veith, Andreas Fröhlich, Armin Biere. On the Complexity of Symbolic Verification and Decision Problems in Bit-Vector Logic. [KVFB-MFCS'14]
- Gergely Kovásznai, Andreas Fröhlich, Armin Biere. Complexity of Fixed-Size Bit-Vector Logics. [KFB-TOCS'15]
- Andreas Fröhlich, Armin Biere, Christoph M. Wintersteiger, Youssef Hamadi. Stochastic Local Search for Satisfiability Modulo Theories. [FBWH-AAAI'15]

ν-Logarithmic Succinct Encodings

- **Upgrading Theorem**: If a problem is complete for a complexity class C, it is complete for a ν-exponentially harder complexity class than C when represented by bit-vectors with ν-logarithmic encoded scalars. [KVFB-MFCS'14]
  - Implication: Word-Level Model Checking and Reachability for bit-vectors with binary encoded bit-widths are EXPSPACE-complete.

- **ν-Succinct SAT**: Satisfiability for quantifier-free bit-vector formulas with ν-logarithmic encoded scalars is $(\nu - 1)$-NEXP-complete (with 0-NEXP := NP). (not published yet)
  - Proof: Reduction from Turing machines or domino tiling problems.