# Towards a Mostly-Automated Prover for Bit-Vector Arithmetic

Iago Abal

High-Assurance Software Laboratory
INESC TEC & Universidade do Minho
Braga, Portugal
iagoabal@di.uminho.pt

July 11, 2013

# Motivation

- The SATisfiability problem.
- Fixed-size bit-vector arithmetic.
- Bit-blasting & clause explosion.

# Motivation

- The SATisfiability problem.
  - Does a model for this formula exist?
- Fixed-size bit-vector arithmetic.
- Bit-blasting & clause explosion.

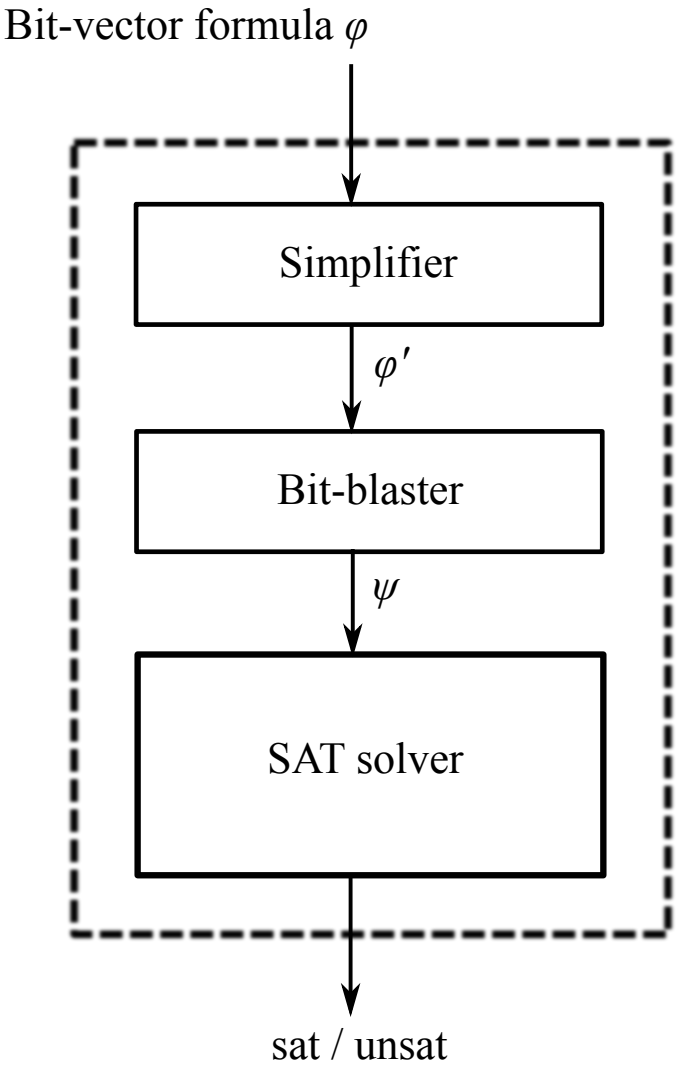
# Motivation

- The SATisfiability problem.
  - Does a model for this formula exist?
- Fixed-size bit-vector arithmetic.
  - Encoding of hardware circuits, crypto algorithms, etc.
- Bit-blasting & clause explosion.

# Motivation

- The SATisfiability problem.
  - Does a model for this formula exist?
- Fixed-size bit-vector arithmetic.
  - Encoding of hardware circuits, crypto algorithms, etc.
- Bit-blasting & clause explosion.
  - Multiplication on large bit-vectors.

# Deciding Bit-Vector Arithmetic




$$a_{[32]}[31:16] \cdot a_{[32]}[15:0] > 0_{[32]}$$

$$\equiv \{a[j:t+1] \cdot a[t:i] \rightarrow a[j:i]\}$$

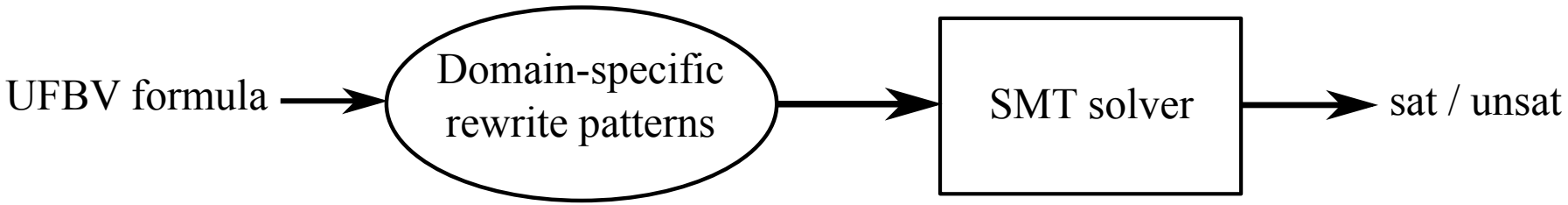
$$a_{[32]}[31:0] > 0_{[32]}$$

$$\equiv \{a_{[n]}[n-1:0] \rightarrow a_{[n]}\}$$

$$a_{[32]} > 0_{[32]}$$

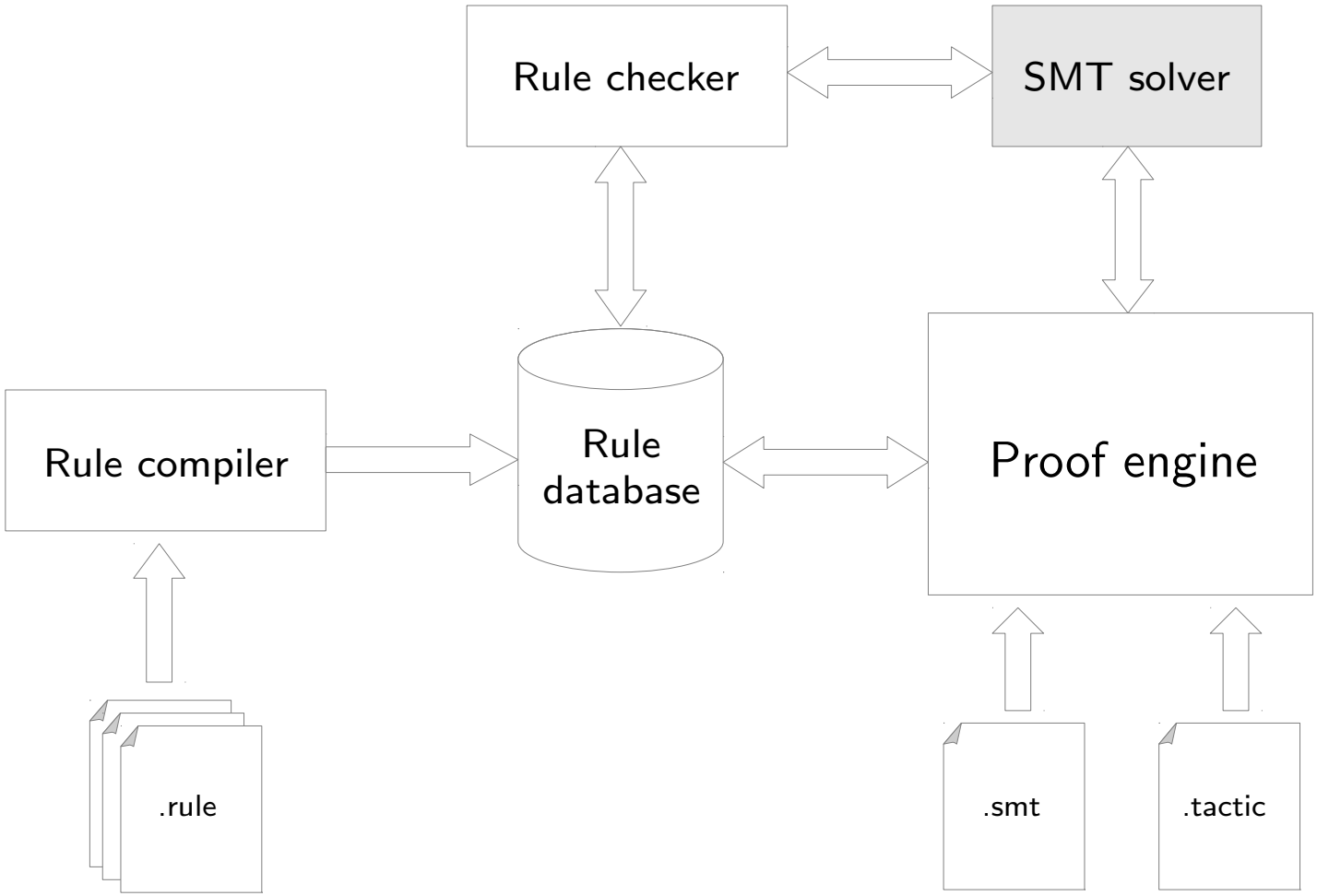
Is $\bigvee_{0 \leq i < 32} a_i$ satisfiable?

# Previous Work




| Benchmark | TR(+Z3) | Z3 | Yices | CVC3 |
|---|---|---|---|---|
| DES encrypt | 0.68 | **0.31** | 1.62 | 25.53 |
| SHA-1 | **4.93** | OOM | T/O | T/O |
| SHA-2 | T/O | OOM | T/O | OOM |
| Peasant multiplication 32-bit | **0.01** | T/O | T/O | OOM |
| Interleaving multiplication 32-bit | OOM | T/O | T/O | OOM |
| Binary exponentiation 128-bit | **0.25** | OOM | OOM | OOM |

# EasyBV


Rule checker
SMT solver
Rule compiler
Rule database
Proof engine
.rule
.smt
.tactic

# Testing Powered by SMT

$$(a_{[32]} \bmod b_{[32]}) \bmod b_{[32]} = a_{[32]} \bmod b_{[32]}$$

# Testing Powered by SMT

$$(k \bmod b_{[32]}) \bmod b_{[32]} = k \bmod b_{[32]},\ k \in [0, 2^{32})$$

# Proof Tactics

## (auto-)rewrite

Apply (installs) rewriting rules.

- Distributivity.

## orelse $t_1 \ldots t_n$

Backtracks on failure.

## par $t_1 \ldots t_n$

Picks the best alternative.

## lift-if

Performs (conservative) if-lifting.

## simpl

Global simplifications and contextual rewriting.

- Constant propagation.
- Unconstrained terms.
- ...

## smt

Off-the-shelf SMT solver.

## auto

Heuristic search.

# Towards a Mostly-Automated Prover for Bit-Vector Arithmetic

Iago Abal

High-Assurance Software Laboratory
INESC TEC & Universidade do Minho
Braga, Portugal
iagoabal@di.uminho.pt

July 11, 2013