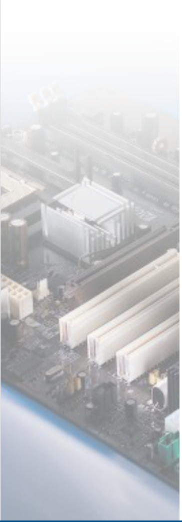


metaSMT: A Unified Interface to SMT-LIB2

Heinz Riener, Mathias Soeken, Clemens Werther,
Görschwin Fey and Rolf Drechsler
University of Bremen
hriener@cs.uni-bremen.de

Forum on specification & Design Languages (FDL) 2014
Munich, Germany

`http://www.informatik.uni-bremen.de/agra/ger/metasmt.php`

October 15, 2014

Solver Integration

① Integration via API

```
z3::context ctx;
z3::solver s(ctx);
const Z3_bool chk = s.check();
if ( chk == z3::unsat ){
  std::cout << "UNSAT" << ’\n’;
}
else if ( chk == z3::sat ){
  std::cout << "SAT" << ’\n’;
}
```

```
Z3_ast Z3_API Z3_mk_add(
  __in Z3_context c,
  __in unsigned num_args,
  __in_ecount(num_args) Z3_ast const args[]);

Z3_ast Z3_API Z3_mk_sub(
  __in Z3_context c,
  __in unsigned num_args,
  __in_ecount(num_args) Z3_ast const args[]);
```

② Integration via File Interface

```
z3::expr x = ctx.bv_const(’x’,8);
// ...
z3::expr formula = x && y || z;
// ...
std::fstream fs("f.smt2",’w’);
// ...
fs <<
  Z3_benchmark_to_smtlib_string(ctx,
    "benchmark", "QF_BV", "unknown",
    0, 0, 0, formula) << ’\n’;
```

```
$ cat f.smt2
(declare-fun x() (_ BitVec 8))
(declare-fun y() (_ BitVec 8))
(declare-fun z() (_ BitVec 8))
;; ...
(assert (= (bvor (bvand x y) z)))
;; ...

$ smt2-solver f.smt2
sat
```

Solver Integration

① Integration via API

```
z3::context ctx;
z3::solver s(ctx);
const Z3_bool chk = s.check();
if ( chk == z3::unsat ){
  std::cout << "UNSAT"
}
else if ( chk == z3::s
  std::cout << "SAT" <
}
```

```
Z3_ast Z3_    Z3_mk_add(
         ext c,
         d num_args,
         um_args) Z3_ast const args[]);

   ast Z3_API Z3_mk_sub(
  __in Z3_context c,
  __in unsigned num_args,
  __in_ecount(num_args) Z3_ast const args[]);
```

Customized, but not easy to change the solver

② Integration via File Interface

```
z3::expr x = ctx.bv_const('x',8);
// ...
z3::expr formula = x && y || z;
// ...
std::fstream fs("f.smt2",'w');
// ...
fs <<
  Z3_benchmark_to_smtlib_string(ctx,
    "benchmark", "QF_BV", "unknown",
    0, 0, 0, formula) << '\n';
```

→

←

```
$ cat f.smt2
(declare-fun x() (_ BitVec 8))
(declare-fun y() (_ BitVec 8))
(declare-fun z() (_ BitVec 8))
;; ...
(assert (= (bvor (bvand x y) z)))
;; ...

$ smt2-solver f.smt2
sat
```

Solver Integration

① Integration via API

```
z3::context ctx;
z3::solver s(ctx);
const Z3_bool chk = s.check();
if ( chk == z3::unsat ){
  std::cout << "UNSAT"
}
else if ( chk == z3::s
  std::cout << "SAT" <
}
```

```
Z3_ast Z3          Z3_mk_add(
             ext c,
             d num_args,
             um_args) Z3_ast const args[]);

     Z3_API Z3_mk_sub(
  __in Z3_context c,
  __in unsigned num_args,
  __in_ecount(num_args) Z3_ast const args[]);
```

Customized, but not easy to change the solver

② Integration via File Interface

```
z3::expr x = ctx.bv_const('x',8);
// ...
z3::expr formula = x && y || z;
// ...
std::fstream fs("f.smt2"
// ...
fs <<
  Z3_benchmark_to_smtl
    "benchmark", "QF_B                   ",
    0, 0, 0, formula) <<  '\n';
```

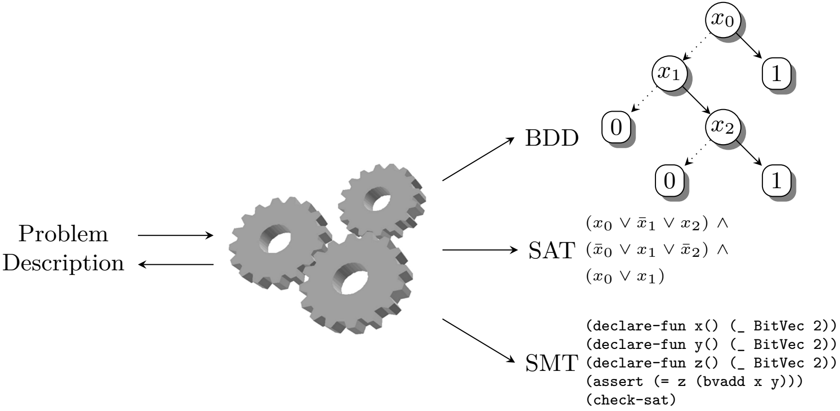
```
$ cat f.smt2
(           x() (_ BitVec 8))
             () (_ BitVec 8))
             () (_ BitVec 8))

               (bvor (bvand x y) z)))
;; ...

$ smt2-solver f.smt2
sat
```

Flexible, but overhead for parsing SMT-LIB2 strings

Formal Reasoning



Problem Description

BDD

SAT

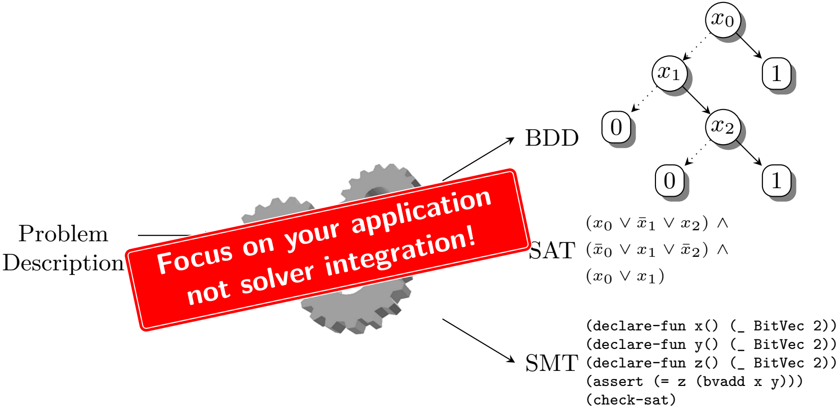
$$(x_0 \vee \bar{x}_1 \vee x_2) \wedge$$
$$(\bar{x}_0 \vee x_1 \vee \bar{x}_2) \wedge$$
$$(x_0 \vee x_1)$$

SMT

```
(declare-fun x() (_ BitVec 2))
(declare-fun y() (_ BitVec 2))
(declare-fun z() (_ BitVec 2))
(assert (= z (bvadd x y)))
(check-sat)
```

Finn Haedicke, Stefan Frehse, Görschwin Fey, Daniel Große, Rolf Drechsler, **metaSMT: Focus on Your Application not on Solver Integration**, DIFTS@FMCAD 2011.

Formal Reasoning



Problem Description

BDD

SAT

$(x_0 \vee \bar{x}_1 \vee x_2) \wedge$
$(\bar{x}_0 \vee x_1 \vee \bar{x}_2) \wedge$
$(x_0 \vee x_1)$

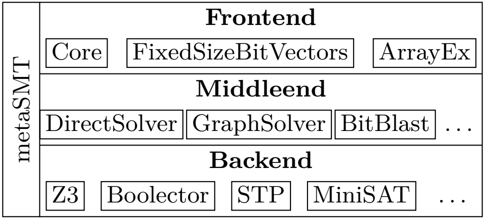
SMT

```
(declare-fun x() (_ BitVec 2))
(declare-fun y() (_ BitVec 2))
(declare-fun z() (_ BitVec 2))
(assert (= z (bvadd x y)))
(check-sat)
```

Focus on your application not solver integration!

Finn Haedicke, Stefan Frehse, Görschwin Fey, Daniel Große, Rolf Drechsler, **metaSMT: Focus on Your Application not on Solver Integration**, DIFTS@FMCAD 2011.

metaSMT: Layer Architecture



metaSMT

Frontend

Core FixedSizeBitVectors ArrayEx

Middleend

DirectSolver GraphSolver BitBlast . . .

Backend

Z3 Boolector STP MiniSAT . . .

metaSMT: one API to rule them all

```
#include <metaSMT/frontend/QF_BV.hpp>
#include <metaSMT/backend/Z3_Backend.hpp>
#include <metaSMT/DirectSolver_Context.hpp>
typedef metaSMT::DirectSolver< Z3_Backend > Context;

Context ctx;
const unsigned width = <parameter>;

bitvector a = new_bitvector(width);
bitvector b = new_bitvector(width);
bitvector c = new_bitvector(width);
assertion( ctx, nequal(a, bvuint(1,width)) );
assertion( ctx, nequal(b, bvuint(1,width)) );
assertion( ctx, equal( zero_extend(width, c),
                  bvmul( zero_extend(width, a), zero_extend(width, b))
  ));

for (unsigned i=0; i < 10000; ++i) {
  unsigned r = random_number ( 2, 2^width-1  );
  assumption( ctx, equal(c, bvuint(r, 2*width)) );

  if( solve( ctx ) ) {
    unsigned a_value = read_value( ctx, a );
    unsigned b_value = read_value( ctx, b );
    printf("factorized %d into %d * %d\n", r, a_value, b_value);
  } else {
    printf("%d is prime.", r);
  }
}
```

Contribution

1. SMT-LIB2 parser and generic evaluator: any metaSMT backend (SAT, BDD, AIG, etc.) can be turned into an SMT solver.
2. TCP server and client architecture to decide SMT instances with multiple decision procedures in parallel. Easy to build (portfolio solver).
3. Experimental results for a selected subset of SMT library benchmarks.

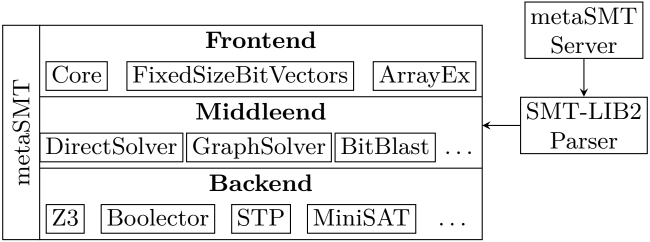
SMT-LIB Format v2.0

```
(declare-fun a () (_ BitVec 2))
(declare-fun b () (_ BitVec 2))
(declare-fun c () (_ BitVec 2))
(declare-fun d () (_ BitVec 2))
(assert
(let (($x38 (ite (= ((_ extract 1 1) d) (_ bv1 1)) true false)))
(let (($x32 (ite (= ((_ extract 1 1) c) (_ bv1 1)) true false)))
(let (($x153 (not $x32)))
(let (($x29 (ite (= ((_ extract 0 0) c) (_ bv1 1)) true false)))
(let (($x26 (ite (= ((_ extract 1 1) b) (_ bv1 1)) true false)))
(let (($x23 (ite (= ((_ extract 0 0) b) (_ bv1 1)) true false)))
(let (($x20 (ite (= ((_ extract 1 1) a) (_ bv1 1)) true false)))
(let (($x17 (ite (= ((_ extract 0 0) a) (_ bv1 1)) true false)))
(let (($x173 (not $x17)))
(let (($x35 (ite (= ((_ extract 0 0) d) (_ bv1 1)) true false)))
(let (($x147 (and
  (or $x17 $x20 $x23 $x26 $x29 $x32 (not $x35)) (or $x17 $x20 $x23 $x26 $x29 $x32 (not $x38))
  (or $x173 $x20 $x23 $x26 $x29 $x32 $x35) (c
  (or $x17 (not $x20) $x23 $x26 $x29 $x32 (no
  (or $x17 $x20 (not $x23) $x26 $x29 $x32 $x3
  (or $x17 $x20 $x23 (not $x26) $x29 $x32 (no
  (or $x17 $x20 $x23 $x26 (not $x29) $x32 (no                                     x38))
  (or $x17 $x20 $x23 $x26 $x29 $x153 (not $x3
  (or $x173 $x20 $x23 $x26 $x29 $x153 $x35) (
(not $x147))
)))))))))))
(check-sat)
```

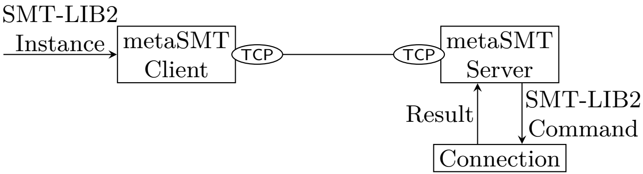
SMT-LIB2

- Standardized format to express problems using (first-order) logic modulo background theories
- Machine-readable, but not necessarily human-readable

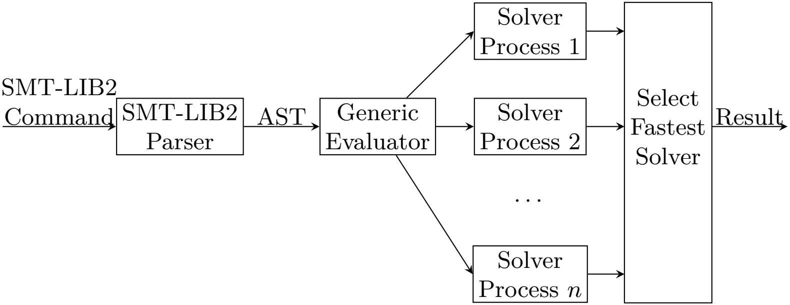
metaSMT: Layer Architecture



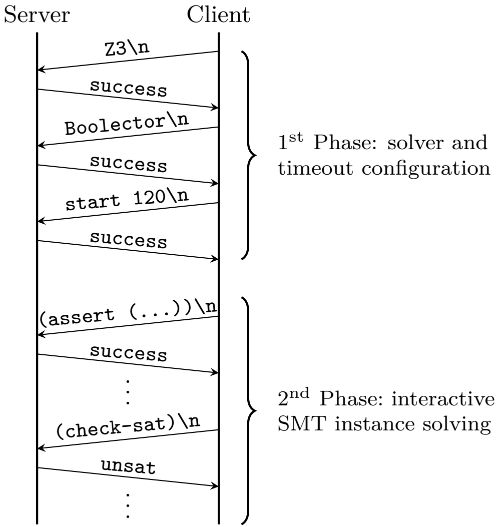
metaSMT: Server/Client Architecture



metaSMT: Solving



metaSMT: Protocol



metaSMT: Performance Evaluation

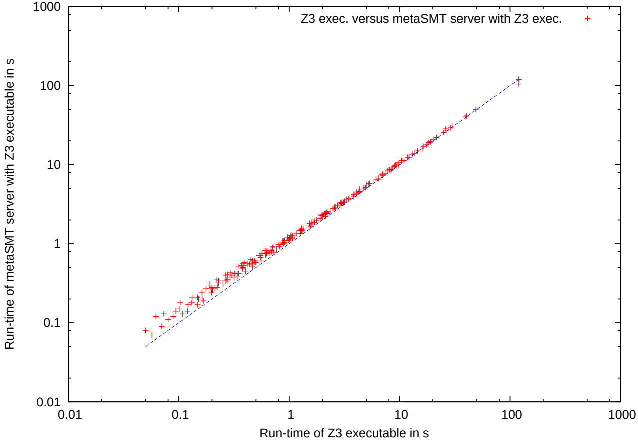
Evaluation of the performance overhead in two experiments:

1. Experiment: Z3 4.1 executable versus metaSMT TCP server configured with I/O streaming interface using Z3 4.1 as backend
2. Experiment: metaSMT portfolio solver using API backends Boolector, Z3, and STP versus fastest and slowest solver

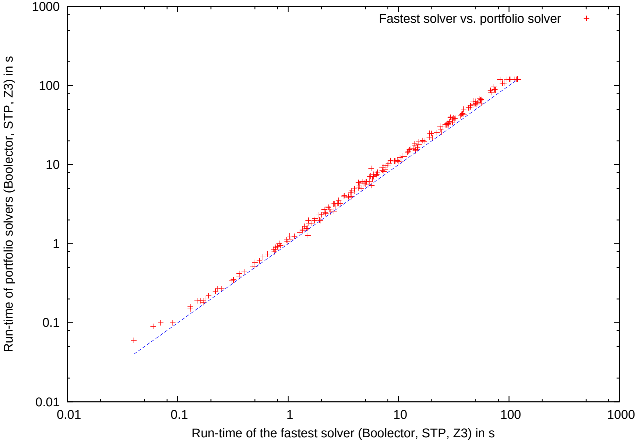
Benchmark set “bruttomesso/lfsr” of crafted benchmarks are used for evaluation.[1]

[1] http://www.informatik.uni-bremen.de/agra/projects/smtlib.html

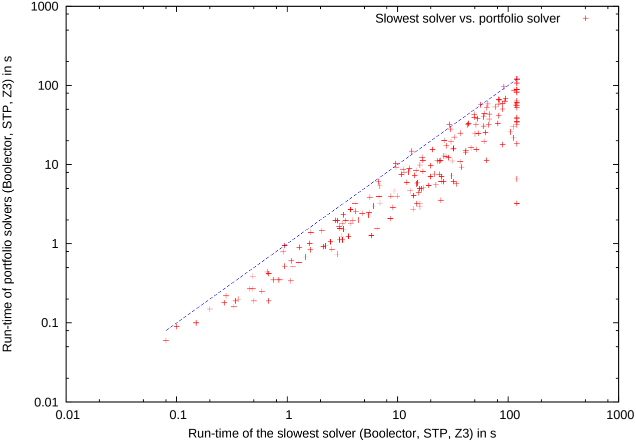
metaSMT: Performance Evaluation



metaSMT: Performance Evaluation

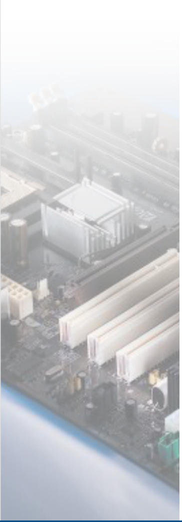


metaSMT: Performance Evaluation



Threats to Validity

- metaSMT API backends use a fixed API mapping which is not necessarily optimal.
- Single benchmark set ("bruttomesso/lfsr") for all experiments.
- These benchmarks were particularly designed to stress the SMT solver (benchmark category "crafted").
- Future work: evaluation with a larger benchmark set and incremental SMT instances.



metaSMT: A Unified Interface to SMT-LIB2

Heinz Riener, Mathias Soeken, Clemens Werther,
Görschwin Fey and Rolf Drechsler
University of Bremen
hriener@cs.uni-bremen.de

Forum on specification & Design Languages (FDL) 2014
Munich, Germany

`http://www.informatik.uni-bremen.de/agra/ger/metasmt.php`

October 15, 2014