

Embedding SMT-LIB into B for Interactive Proof and Constraint Solving

Sebastian Krings, Michael Leuschel

Niederrhein University of Applied Sciences, University of Düsseldorf



Introductory Examples

Listing 1: Boolean Example in SMT-LIB

```
(set-logic QF_UF)
(declare-fun p () Bool)
(assert (and p (not p)))
(check-sat)
```

Listing 2: Boolean Example in B

```
MACHINE BooleanExample
CONSTANTS p
PROPERTIES
  p:BOOL & p=TRUE & not (p=TRUE)
END
```



Introductory Examples

Listing 3: Integer Example in SMT-LIB

```
(set-logic QF_LIA)
(declare-fun x () Int)
(declare-fun y () Int)
(declare-fun z () Int)
(assert (= (+ (* 6 x) (* 2 y) (* 12 z)) 30))
(assert (= (+ (* 3 x) (* 6 y) (* 3 z)) 12))
(check-sat)
```

Listing 4: Integer Example in B

MACHINE IntegerExample

CONSTANTS x, y, z

PROPERTIES

x:INTEGER & y:INTEGER & z:INTEGER &

$6 * x + 12 * y + 3 * z = 30$ &

$3 * x + 6 * y + 3 * z = 12$

END



Tricky Parts

- Predicates and booleans distinct in B
 - ▶ $\Rightarrow$ Just use `x = TRUE` instead of `x`
- Definition of division diverges
 - ▶ Rounding behavior
 - ▶ Division by zero
 - ▶ $\Rightarrow$ Use definition of division rather than native B division



Even Trickier Parts

- If-Then-Else
- Let
- Well-definedness



If-Then-Else

Translate (ite P E_1 E_2):

- Both E_1 and E_2 predicates: $(P \Rightarrow E_1) \wedge (\neg P \Rightarrow E_2)$
- Otherwise:
 - ▶ Relation mapping 1 to E_1 and E_2 :
 $(\lambda t \cdot (t = 1 | E_1)) \cup (\lambda t \cdot (t = 1 | E_2))$
 - ▶ Assure that one is empty depending on P :
 $(\lambda t \cdot (t = 1 \wedge P | E_1)) \cup (\lambda t \cdot (t = 1 \wedge \neg P | E_2))$
 - ▶ apply to 1 to extract the result:
 $(\lambda t \cdot (t = 1 \wedge P | E_1)) \cup (\lambda t \cdot (t = 1 \wedge \neg P | E_2))(1)$



Well-Definedness

$$(ite\ (not\ (= x\ 0))\ (div\ 10\ x)\ (div\ 10\ (-\ x\ 1)))$$

translated to

$$(\lambda t \cdot (t = 1 \wedge x \neq 0 | 10/x)) \cup (\lambda t \cdot (t = 1 \wedge \neg(x \neq 0) | 10/(x-1)))(1)$$

Why not use

$$\{TRUE \mapsto 10/x, FALSE \mapsto 10/(x-1)\}(bool(x \neq 0))?$$

- Needs to compute set comprehension before application
- i.e., needs to compute $10/x$ and $10/(x-1)$
- Even for $x = 0$ and $x = 1$
- $\Rightarrow$ Not well-defined in B!



But why translate to B, when we have all these nice SMT solvers?



Advantages on the B Side of Live

- Interactive Proof
- Meta-Level Reasoning, i.e., reason about SMT-solving algorithms using B-method
- Use PROB for Constraint Solving, Animation, Visualization
- We gain Benchmarks :-)



Proof and Meta-Level Reasoning

- Use B-method tools to proof properties in SMT-LIB
- Use B-method to reason about SMT solving algorithms



Interactive Proof - CVC4 Snippet

```
if(den.isConst()) {
  if(den != 0) {
    if(den > 0) {
      d_containing.d_out->lemma(nm->mkNode(kind::AND,
        nm->mkNode(kind::LEQ,
          nm->mkNode(kind::MULT, den, intVar), num),
        nm->mkNode(kind::LT, num,
          nm->mkNode(kind::MULT, den,
            nm->mkNode(kind::PLUS, intVar,
              nm->mkConst(Rational(1))))))),));
    } else {
      d_containing.d_out->lemma(nm->mkNode(kind::AND,
        nm->mkNode(kind::LEQ,
          nm->mkNode(kind::MULT, den, intVar), num),
        nm->mkNode(kind::LT, num,
          nm->mkNode(kind::MULT, den,
            nm->mkNode(kind::PLUS, intVar,
              nm->mkConst(Rational(-1))))))),));
    }
  }
  return intVar;
}
```



Interactive Proof - Corresponding B Machine

DEFINITIONS

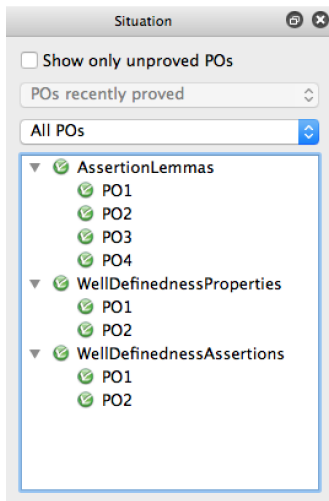
```
smt_abs(x) == max({-x,x});  
smt_div_mod(m,n,q,r) ==  
  (n /= 0 => m = n*q + r & 0 <= r & r <= smt_abs(n) - 1);  
smt_div(x,y,divres) ==  
  #(modres).(modres:INTEGER  
    & smt_div_mod(x,y,divres,modres));  
smt_mod(x,y,modres) ==  
  #(divres).(divres:INTEGER  
    & smt_div_mod(x,y,divres,modres))
```

ASSERTIONS

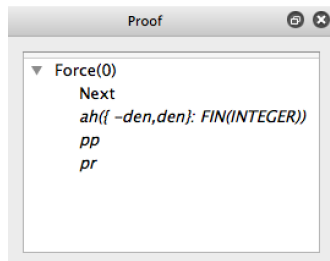
```
!(num,den,intVar).(num:INTEGER & den:INTEGER  
  & intVar:INTEGER &  
  smt_div(num,den,intVar) =>  
    (den > 0 =>  
      den*intVar <= num & num < den*(intVar + 1)) &  
    (den <= 0 =>  
      den*intVar <= num & num < den*(intVar - 1)))
```



Interactive Proof - AtelierB



(a) List Of Proof Obligations



(b) Successful Proof Tree

Constraint / SMT-Solving

SMT solvers and B-method tools use different techniques:

- SMT: DPLL(τ), underlying CDCL-based SAT solving
- PROB: Prolog-style constraint logic programming over finite domains
- AtelierB: different theorem provers



Solving - Benchmarks - Per Solver

solver	configuration	# successful tests	∅ runtime sat	∅ runtime unsat
PROB	Vanilla	9081	0.84 (0.04)	1.38 (0.85)
PROB	Random	9081	0.84 (0.04)	1.39 (0.85)
PROB	CSE	9074	0.84 (0.04)	1.38 (0.84)
PROB	CHR	9065	0.87 (0.04)	1.32 (0.73)
Z3	-	16322	0.01 (0.01)	0.56 (0.56)
CVC4	-	7757	3.5 (5.16)	1.29 (1.29)

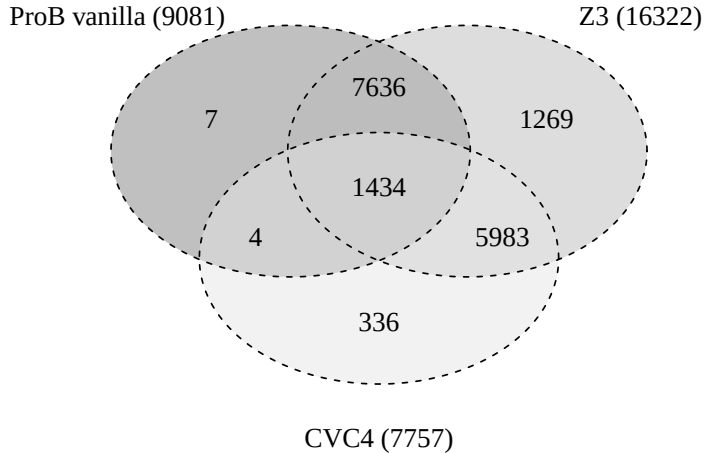


Solving - Benchmarks - Common Successful

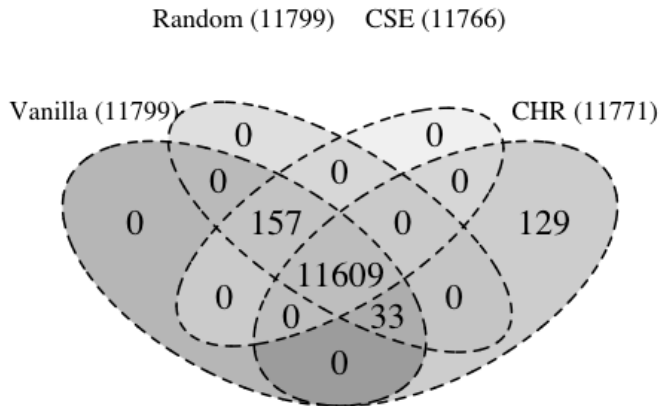
solver	configuration	# successful tests	∅ runtime sat	∅ runtime unsat
PROB	Vanilla	1328	1.65 (0.83)	1.23 (0.61)
PROB	Random	1328	1.64 (0.82)	1.24 (0.63)
PROB	CSE	1328	1.64 (0.8)	1.22 (0.6)
PROB	CHR	1328	2.61 (2.42)	1.29 (0.68)
Z3	-	1328	0.12 (0.16)	0.02 (0.02)
CVC4	-	1328	0.57 (0.81)	0.05 (0.05)



Solving - Benchmarks - Overlap



Solving - Benchmarks - ProB Options



Solving - Benchmarks - Overall Results

solver	configuration	sat	unsat	unknown	timeout	memory out
PROB	Vanilla	8625	456	1444	7228	0
PROB	Random	8625	456	1444	7228	0
PROB	CSE	8625	449	1444	7235	0
PROB	CHR	8627	438	1138	7550	0
Z3	-	12472	3850	412	1019	0
CVC4	-	4152	3605	9324	672	0

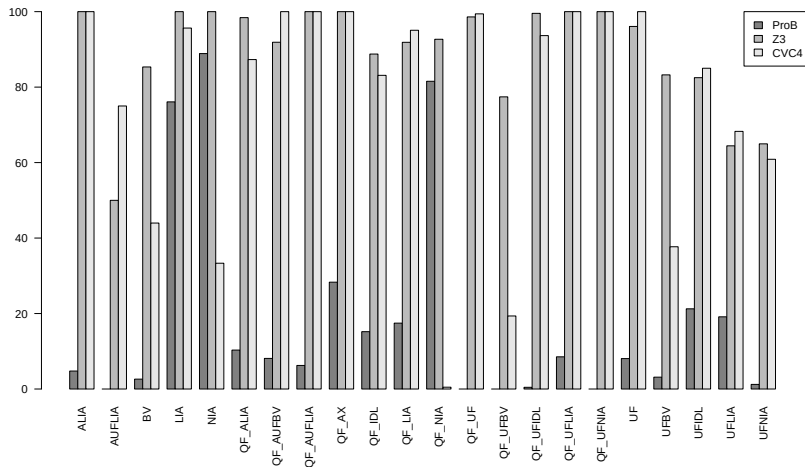


Future Work

- Support more of SMT-LIB, e.g., uninterpreted functions
- Tried some simple encodings
- In theory, full coverage of SMT-LIB
- In practice, encodings often unhelpful



Simple Encoding not Helpful



Conclusions

- Groundwork for analysis or proof of SMT solving algorithms using the B method
- Cross-checking of results using a different approach to solving
- Using SMT-LIB's benchmarks to validate and improve B tools
- Using Atelier B, provided interactive (assisted) proof
- PROB already provides reasoning over sets and strings, features starting to find their way into SMT-LIB
- Competitive for special kinds of benchmarks



Thank you!

Any questions?

