SMT-LIB for HOL

Daniel Kroening    Philipp Rümmer    Georg Weissenbacher
Oxford University Computing Laboratory

ITP Workshop
MSR Cambridge
25 August 2009
The SMT-LIB Standard

SMT → Satisfiability Modulo Theories

SMT-LIB is . . .

- a standardised input format for SMT-solvers (since 2003)
- a standardised format for exchanging SMT problems
- a library of more than 60,000 SMT benchmarks
- the basis for the annual SMT competition (this year: at CADE)

Theories in SMT-LIB:

- integer and rational arithmetic (linear)
- uninterpreted functions
- arrays
- finite-width bit-vectors
Some state-of-the-art SMT-solvers:

- Alt-Ergo, Argo-lib, Barcelogic, CVC3, DTP, Fx7, haRVey, MathSAT, Spear, STP, Yices, Z3
- All are completely automatic
- Standard architecture: DPLL + small theory engines + quantifier heuristics
- “Good for shallow reasoning”
- Used as back-ends in many verification systems: Krakatoa, Caduceus, ESC/Java2, Spec#, VCC, Havoc, CBMC, ...
Example in SMT-LIB Format

```
(benchmark Ensures_Q_noinfer_2
 :source { Boogie/Spec# benchmarks. }
 :logic AUFLIA

[...]
:extrapreds ((InRange Int Int ))
:extrafuns (( this Int ))
:extrafuns (( intAtLeast Int Int Int ))

[...]
:assumption
   (forall (?t Int) (?u Int) (?v Int)

[...]
:formula
   (not (implies (implies (implies (implies
      (and
         (forall (?o Int) (?F Int)
            (implies (and (= ?o this) (= ?F X)) (= (select2 H ?o ?F) 5)))
         (implies
            (forall (?o Int) (?F Int)
               (implies (and (= ?o this) (= ?F X)) (= (select2 H ?o ?F) 5)))
            (implies true true))
         (= ReallyLastGeneratedExit_correct Smt.true))
         (= ReallyLastGeneratedExit_correct Smt.true))
         (= start_correct Smt.true))
         (= start_correct Smt.true)))
```
The SMT-LIB Format

SMT-LIB is currently quite low-level:

- No high-level types like sets, lists, maps, etc.

Solutions practically used:

- Much can be encoded in arrays + axioms (+ prover-specific extensions)
- Some solvers offer algebraic datatypes (not standardised)

⇒ Against the idea of SMT-LIB
The SMT-LIB Format (2)

- Current version of the standard: 1.2
- Version 2 to be finished sometime in 2009

New Features in Version 2

- Type constructors, parametric theories
- Various simplifications
- ...

- New theories! (hopefully)
Our Proposal for New SMT-LIB Theories

Datatypes inspired by VDM-SL

- Tuples
- (Finite) Lists
- (Finite) Sets
- (Finite) Partial Maps

Our main applications

- Reasoning + test-case generation for UML/OCL
- (Bounded) Model checking with abstract library models
- VDM-SL
## Signature of the SMT-LIB Theories

<table>
<thead>
<tr>
<th>Tuples</th>
<th>Sets</th>
<th>Lists</th>
<th>Maps</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Tuple $T_1 \ldots T_n$)</td>
<td>(Set $T$)</td>
<td>(List $T$)</td>
<td>(Map $S \rightarrow T$)</td>
</tr>
<tr>
<td>tuple $\langle x_1, \ldots, x_n \rangle$</td>
<td>emptySet $\emptyset$</td>
<td>nil $\mathbf{[]}$</td>
<td>emptyMap $\emptyset$</td>
</tr>
<tr>
<td>project $x_k$</td>
<td>insert $M \cup {x}$</td>
<td>cons $x :: L$</td>
<td>apply $f(x)$</td>
</tr>
<tr>
<td>product $M_1 \times \ldots \times M_n$</td>
<td>in $\in$</td>
<td>head</td>
<td>overwrite</td>
</tr>
<tr>
<td></td>
<td>subset $\subseteq$</td>
<td>tail</td>
<td></td>
</tr>
<tr>
<td></td>
<td>union $\cup$</td>
<td>append $\bowtie$</td>
<td>domain</td>
</tr>
<tr>
<td></td>
<td>inter $\cap$</td>
<td>length $</td>
<td>l</td>
</tr>
<tr>
<td></td>
<td>setminus $\setminus$</td>
<td>nth $l_k$</td>
<td>restrict $\triangleleft$</td>
</tr>
<tr>
<td></td>
<td>card $</td>
<td>M</td>
<td>$</td>
</tr>
<tr>
<td></td>
<td>$\mathbf{</td>
<td>M</td>
<td>}$</td>
</tr>
</tbody>
</table>
Example: Verification Cond. Generated by VDMTools

In VDM-SL notation:

\[
\forall l : L(\mathbb{Z}), i : \mathbb{N}. \ (i \in \text{inds}(l) \Rightarrow \forall j \in \text{inds}(l) \setminus \{i\}. \ j \in \text{inds}(l))
\]

In SMT-LIB notation:

\[
(\forall l ((l (\text{List} \ \text{Int})) \ (i \text{ Int}))
\ 
(\text{implies}
\ 
(\text{and} \ (\geq i 0) \ (\in i (\text{inds} \ l))))
\ 
(\forall j (\text{Int})
\ 
(\text{implies}
\ 
(\in j (\text{setminus} (\text{inds} \ l) \ (\text{set} \ i)))
\ 
(\in j (\text{inds} \ l))))))
\]
Event-B File System Case Study (delete/inv8)

\[
\begin{align*}
parent \in objects \setminus \{root\} \rightarrow objects, \\
obj \in objects \setminus \{root\}, \quad des \subseteq objects, \\
des = (\text{tcl}(parent)) \sim \{\{obj\}\}, \quad objs = des \cup \{obj\} \\
\Rightarrow \quad objs \nsubseteq parent \in (objects \setminus objs) \setminus \{root\} \rightarrow objects \setminus objs
\end{align*}
\]

objects, des, objs : (Set OBJECT)  
parent : (Map OBJECT OBJECT)  
obj : OBJECT

(implies ... (and
 (= (domain (subtract parent objs)))
 (setminus objects objs (insert emptySet root)))
 (subset (range (subtract parent objs))
 (setminus objects objs)))
Translation of Event-B proof obligations

- Carrier sets → SMT-LIB types
- Sets → finite sets
- Functions → finite partial maps or arrays

- SMT-LIB is strongly typed → type inference necessary
- Potential issue: finiteness of SMT-LIB datatypes
Status of the Proposal

- Syntax + Semantics of theories is formally defined
  ⇒ In collaboration with Cesare Tinelli
  ⇒ Was presented at SMT workshop 2009

- Pre-processor is under development
  ⇒ Converter SMT-LIB 2 → SMT-LIB 1

- Decidability is being investigated
Proofs vs. Refutations

Refutations: SMT solvers produce satisfying assignments.

What about proofs?

- All SMT solvers use DPLL communicating with theory solvers
- Theory solvers can be made to produce deduction steps

⇒ Proof can be exported, checked by trusted kernel in ITP