A Theory of Finite Sets, Lists, and Maps for the SMT-LIB Standard

Daniel Kroening  Philipp Rümmer  Georg Weissenbacher

Oxford University Computing Laboratory
philr@comlab.ox.ac.uk

SMT Workshop 2009 on CADE 22
2 August 2009
Motivation of new theories for SMT-LIB 2
Proposal of theories
Some examples (VDM, Event-B)
SMT-LIB 2 converter
Practical and theoretical issues

More information + implementation:
http://www.cprover.org/SMT-LIB-LSM/
## Motivation: SMT Applications

<table>
<thead>
<tr>
<th>Bounded Model Checking for C, C++ (CBMC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Until recently: based on SAT solving</td>
</tr>
<tr>
<td>SMT backend using bit-vectors + arrays (QF_AUFBV)</td>
</tr>
<tr>
<td>Planned: other theories to handle library models (e.g., STL), memory models, etc.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model-based test-case generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modelling languages like UML/OCL (e.g., state charts), Simulink/Stateflow, Lustre</td>
</tr>
<tr>
<td>E.g., by bounded model checking, constraint solving</td>
</tr>
<tr>
<td>EU projects Mogentes, CESAR</td>
</tr>
</tbody>
</table>
Motivation: SMT Applications (2)

Analysis of requirements + architecture specifications
- Consistency, coverage, animation, etc.
- Languages developed in EU project CESAR

System development in Event-B, VDM
- Set-theoretic specification languages
- Proof obligations for invariant preservation, refinement, etc.
- Event-B tool with built-in proof assistant (Rodin)
  ⇒ Currently no usage of SMT
SMT-LIB is Great!

- Simple format
- Many available solvers, backend easily changeable
- Experiences in CBMC:
  Significantly reduced implementation effort
  (Compared to native SMT solver formats)
SMT-LIB is Great! but there are some issues . . .

Issues with SMT-LIB 1.2, from our point of view

- Fixed types of arrays indices/values (e.g., no boolean arrays)
- Separation between formulae and terms
- No constant arrays
- . . . missing datatypes

⇒ Many problems are fixed by upcoming SMT-LIB 2
Much can be encoded in arrays + uninterpreted functions + axioms
⇒ Difficult to build decision procedures (for decidable fragments)

Many solvers offer extensions + further theories (e.g., algebraic datatypes)
⇒ Not standardised, against the SMT-LIB idea

*Introduce further SMT-LIB theories*
## Proposal for Additional SMT-LIB 2 Theories

### Datatypes inspired by VDM-SL

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

- Defined as parametric SMT-LIB 2 theories
- Semantics in terms of classical set theory
### Signature of the Datatypes

<table>
<thead>
<tr>
<th>Tuples</th>
<th>Sets</th>
<th>Lists</th>
<th>Maps</th>
</tr>
</thead>
<tbody>
<tr>
<td>((\text{Tuple}_n T_1 \ldots T_n))</td>
<td>((\text{Set} \ T))</td>
<td>((\text{List} \ T))</td>
<td>((\text{Map} \ S \ T))</td>
</tr>
<tr>
<td>(\text{tuple} \ (x_1, \ldots, x_n))</td>
<td>(\text{emptySet} \ \emptyset)</td>
<td>(\text{nil} \ [])</td>
<td>(\text{emptyMap} \ \emptyset)</td>
</tr>
<tr>
<td>(\text{project} \ x_k)</td>
<td>(\text{insert} \ M \cup {x})</td>
<td>(\text{cons} \ x :: L)</td>
<td>(\text{apply} \ f(x))</td>
</tr>
<tr>
<td>(\text{product} \ M_1 \times \ldots \times M_n)</td>
<td>(\text{in} \ \in)</td>
<td>(\text{head})</td>
<td>(\text{overwrite})</td>
</tr>
<tr>
<td></td>
<td>(\text{subset} \ \subseteq)</td>
<td>(\text{tail})</td>
<td>(\leftrightarrow)</td>
</tr>
<tr>
<td></td>
<td>(\text{union} \ \cup)</td>
<td>(\text{append} \ \lhd)</td>
<td>(\text{domain})</td>
</tr>
<tr>
<td></td>
<td>(\text{inter} \ \cap)</td>
<td>(\text{length} \</td>
<td>\cdot</td>
</tr>
<tr>
<td></td>
<td>(\text{setminus} \ \setminus)</td>
<td>(\text{nths} \ l_k)</td>
<td>(\text{restrict} \ \triangle)</td>
</tr>
<tr>
<td></td>
<td>(\text{card} \</td>
<td>\cdot</td>
<td>)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(\text{elems} \ {l_1, \ldots,</td>
<td>\cdot</td>
</tr>
</tbody>
</table>
Defined Theories

- Sets with cardinality
- Sets + Tuples
- Lists with length
- Finite Maps
- Combined theories
Defined Theories (preliminary decidability results)

- Sets with cardinality:
  - non-nested: decidable
  - nested + quantifiers: undecidable
  - nested, quantifier-free: ???

- Sets + Tuples: undecidable

- Lists with length:
  - word equations with equal-length predicate,
    known open problem

- Finite Maps: ???

- Combined theories: undecidable
In VDM-SL notation:

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

In SMT-LIB notation:

\[
\text{(forall } ((l (List Int)) (i Int)) \\
\text{ (implies} \\
\text{ (implies} \\
\text{ (and } (\geq i 0) (\text{in } i (\text{inds } l)))) \\
\text{ (forall } (j Int) \\
\text{ (implies} \\
\text{ (implies} \\
\text{ (in } j (\text{setminus} (\text{inds } l) (\text{set } i))) \\
\text{ (in } j (\text{inds } l)))\text{)))}
\]
parent ∈ objects \ {root} → objects,
obj ∈ objects \ {root}, des ⊆ objects,
des = (tcl(parent)) ∼ [{obj}], objs = des ∪ {obj}
⇒ objs ⊆ parent ∈ (objects \ objs) \ {root} → objects \ objs

:extrafuns((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)
            ))
Status of the Proposal

- Syntax + Semantics of theories is defined
  ⇒ In collaboration with Cesare Tinelli

- Parser + type checker available

- Meaningful sublogics still to be identified

- Decidability is being investigated
SMT-LIB 2 Front-end

⇒ Finished just in time for workshop

- Motivation: ease the adoption of SMT-LIB 2 + theories
- Parser + type checker for (almost) complete SMT-LIB 2
- All proposed theories are supported

- Current backend: converter to SMT-LIB 1
  ⇒ We consider further backends (like: native Z3)

- Of course: we would prefer direct support by SMT solvers

http://www.cprover.org/SMT-LIB-LSM/
Current (Incomplete) Axiomatisation of Theories

- Sets with cardinality: arrays + uninterpreted functions + axioms
- Tuples, lists, maps: uninterpreted functions + axioms
CBMC already has an SMT-LIB 2 backend
Significantly reduced implementation effort
Code size $\approx 50\%$ compared to SMT-LIB 1
Currently: Tuples are used, other theories in near future

Most positive changes compared to SMT-LIB 1.2
- No term/formula distinction
- Possibility of polymorphic arrays
- Let-expressions as terms
- CBMC verification conditions
- Received $\approx 5500$ verification conditions from Event-B community
- Still to be converted to SMT-LIB 2 format
Trade-off when defining theories:

- Generality  → good for users
- Implementation complexity  → good for tool writers
- Decidability

⇒ We hope that we have found a compromise
⇒ Comments are welcome!

- We received very positive feedback from Event-B community
Thanks for your attention!

http://www.cprover.org/SMT-LIB-LSM/