OpenMath and SMT-LIB

James Davenport, Matthew England, Roberto Sebastiani & Patrick Trentin[1]
Universities of Bath/Coventry/Trento/Trento
J.H.Davenport@bath.ac.uk

17 July 2017

[1]Thanks to EU H2020-FETOPEN-2016-2017-CSA project $\mathcal{SC}^2$ (712689) and Pascal Fontaine

Aims

OpenMath and SMT-LIB [BFT15] are languages with very different origins, but both "represent mathematics". We describe first SMT, and then SMT-LIB for the OpenMath community and consider adaptations for both languages to support the SC^2 initiative.

The overall aim of the EU-funded Horizon 2020 Project SC^2 is to create a new research community bridging the gap between **S**atisfiability **C**hecking and **S**ymbolic **C**computation, so that members well informed about both fields can ultimately resolve problems currently beyond the scope of either.

SAT and SMT

SAT(isfiability) Is a Boolean formula (generally in CNF) in Boolean p_i satisfiable? In theory, the output is either SAT or UNSAT: in practice SAT is accompanied by a witness, and UNSAT should be accompanied by a proof.

S(at) M(odulo) T(heories) The p_i now have an interpretation in some underlying theory.

$(p_1 \vee p_2) \wedge (p_1 \vee \neg p_2) \wedge (\neg p_1 \vee \neg p_2)$ is SATisfiable: $p_1 = T$, $p_2 = F$.
But when $p_1 = (x > 1)$ and $p_2 = (x > 0)$, this means $x > 1 \wedge x \leq 0$: UNSATisfiable.

SAT (DIMACS format [Spe15])

- Problems in Conjunctive Normal Form format
- Housekeeping information, then one clause per line
- p_i is represented by i, $\neg p_i$ by $-i$

$(p_1 \vee p_2) \wedge (p_1 \vee \neg p_2) \wedge (\neg p_1 \vee \neg p_2)$

```
p cnf 2 3

1 2 0
1 -2 0
-1 -2 0
```

All SAT-solvers eat this, it's used in contests etc.

SMT-LIB is (sort of) the SMT equivalent

1. a language for writing terms and formulas in a sorted (i.e., typed) version of first-order logic;
2. a language for specifying background theories and fixing a standard vocabulary of sort, function, and predicate symbols for them;
3. a language for specifying logics, suitably restricted classes of formulas to be checked for satisfiability with respect to a specific background theory;
4. a command language for interacting with SMT solvers via a textual interface that allows asserting and retracting formulas, querying about their satisfiability, examining their models or their unsatisfiability proofs, and so on.

Example: levels 1 and 2

$(p_1 \vee p_2) \wedge (p_1 \vee \neg p_2) \wedge (\neg p_1 \vee \neg p_2)$

```
(and (or p1 p2)
     (or p1 (not p2))
     (or (not p1) (not p2)))
```

In the language

```
(theory Core
  :sorts ( (Bool 0) )
  :funs ( (true Bool) (false Bool) (not Bool Bool)
          (=> Bool Bool Bool :right-assoc) (and Bool Bool
          (or Bool Bool Bool :left-assoc) (xor Bool Bool Bo
          (par (A) (= A A Bool :chainable))
          (par (A) (distinct A A Bool :pairwise))
          (par (A) (ite Bool A A A)) )
```

SMT-LIB Commands (Level 4) 1 of 2

(assert ⟨term⟩)
(check-sat)
(check-sat-assuming (⟨prop_literal⟩*))
(declare-const ⟨symbol⟩ ⟨sort⟩)
(declare-fun ⟨symbol⟩ (⟨sort⟩*) ⟨sort⟩)
(declare-sort ⟨symbol⟩ ⟨numeral⟩)
(define-fun ⟨fun_def⟩)
(define-fun-rec ⟨fun_def⟩)
(define-funs-rec (⟨fun_dec⟩$^{n+1}$) (⟨term⟩$^{n+1}$))
(define-sort ⟨symbol⟩ (⟨symbol ⟩*) ⟨sort⟩)
(echo ⟨string⟩)
(exit)
(get-assertions)
(get-assignment)
(get-info ⟨info_flag⟩)
(get-model)

SMT-LIB Commands (Level 4) 2 of 2

(`get-option` ⟨keyword⟩)
(`get-proof`)
(`get-unsat-assumptions`)
(`get-unsat-core`)
(`get-value` (⟨term⟩+))
(pop ⟨numeral⟩)
(push ⟨numeral⟩)
(reset)
(reset-assertions)
(set-info ⟨attribute⟩)
(set-logic ⟨symbol⟩)
(set-option ⟨option⟩)

Initial Comparison

concept	OpenMath	SMTLIB
expression	`<OMA>` terms `</OMA>`	(terms)
binders	`<OMBIND>` terms `</OMBIND>`	`exists/forall/let`
variables	`<OMV name="x">`	nullary function x
bound "	`<OMBVAR>` ...	occurrence in binders
	(untyped)	(sorted)
integers	`<OMI>`[-]nn`</OMI>`	nn or (- nn)
typed	Small Type System	well-sorted in a level2 theory
semantics	FMP/CMP	level 3 (informal?)
commands	SCSCP?	level 4

Exists Uniquely: $\exists!$ [Alu09, p. 3] for a textbook

It is not logically necessary: two alternative definitions are as follows:

$$\exists! P(x) \Leftrightarrow \exists x \left(P(x) \wedge \forall y (P(y) \Rightarrow x = y)\right) \tag{1}$$

$$\exists! P(x) \Leftrightarrow (\exists x P(x)) \wedge (\forall y \forall z P(y) \wedge P(z) \Rightarrow y = z) \tag{2}$$

Considered computationally, (1) introduces an alternation, but fewer distinct quantifiers, and fewer repetitions of P than (2).

OpenMath Since it is both useful and economical (saving the repetition of P, and the human/computer needing to recognise that it is the same P), there seems no reason not to introduce it.

SMT-LIB Here the argument is more finely balanced. The arguments for are the same as for OpenMath (except that an SMT solver is expected to have clever heuristics, and idiom recognition might well be one of those). The converse argument is that adding syntactic sugar is adding noise too.

Challenges for SMT-LIB (JHD's view)

1. `get-model` in nonlinear contexts, e.g. $x^2 = 2$

   * One could try adding algebraic numbers. However, the current SMT-LIB requires [BFT15, Page 60] that two terms with different expressions in the description of the model have a different value in the model. So it would be necessary to use some form of canonicalization of algebraic numbers that guarantees this.

2. Extension to max: see OptiMathSAT [ST15] which adds `maximize`, `minimize` "commands". In fact, these are statements as to the nature of the goal(s), and the goal is achieved by `check-sat`. OpenMath, of course, has max.

3. Extension to argmax: see OptiMathSAT [ST15] which can do `(get-value argument1)` etc. to find the values at which the maximum discovered by `check-sat` was achieved. OpenMath doesn't have argmax.

Is OpenMath's max what we want?

`minmax1.max` returns the maximum of a set, so $\max_{x \in [0,1]} x(1-x)$ becomes $\max\{x(1-x) : x \in [0,1]\}$, a maximum of an uncountable set. Is this what we want? A solution could be either (or both) of these.

1. An operator that took both a set and a function: essentially making explicit the idiom `minmax1.max(set1.map(...))` referred to above.
2. A binder that took both a function body and a predicate, using the same bound variable for both.

The second one probably has the advantage of being closer to common usage, but would require some of the extensions to binding suggested in [DK09].

OpenMath and argmax: is this what we want?

A relatively recent piece of mathematical notation is argmax, which does not have an extremely formal definition. The Wikipedia definition is that these "are the points of the domain of some function at which the function values are maximized." Hence naïvely,

$$\operatorname*{argmax}_{x \in \mathbf{R}} \sin(x) = \left\{ \frac{\pi}{2} + 2n\pi | n \in \mathbf{Z} \right\}.$$

Though it can be defined in terms of other objects, it might be helpful to have a `argmax` constructor in OpenMath, capable of encoding $\operatorname{argmax}_{x \in [0,1]} x(1-x)$ (whose value is $\left\{\frac{1}{2}\right\}$) .
This is a perfectly sound mathematical definition, but does not really meet the requirements of SC^2, or computation in general. What SC^2 really needs is a *witness* point, i.e. a single value x_0 such that $f(x_0) = \max_{f \in S} f(x)$. For the sake of mathematical notation, we term this $\operatorname{argmax}^{(1)}$ — one important point would be that it is not necessarily deterministic. This constructor could be called `argmaxone`.

Observations

1. SMT-LIB has a richer meta-syntactic vocabulary: distinguishes `:left-assoc` from `:right-assoc`, has `:chainable` and `:pairwise`
2. SMT-LIB has a command layer (part generic, part SMT-specific)

OpenMath Recommendations

1. Introduce `existsuniquely`
2. Introduce `max` (function, set) somehow
3. Introduce `argmaxone` (function, set) analogously
4. Consider incorporating or referring to SCSCP

Questions?

Bibliography


P. Aluffi.
Algebra: Chapter 0.
AMS, 2009.


C. Barrett, P. Fontaine, and C. Tinelli.
The SMT-LIB Standard: Version 2.5.
http://smtlib.cs.uiowa.edu/papers/smt-lib-reference-v2.5-r2015-06-28.pdf, 2015.


J.H. Davenport and M. Kohlhase.
Quantifiers and Big Operators in OpenMath.
https://www.researchgate.net/profile/Dan_Roozemond/publication/253932330_OpenMath_in_SCIEnce_

Bibliography

Evolving_of_Symbolic_Computation_Interaction/links/00b7d5375cc5cea0ce000000.pdf#page=119, 2009.

I. Spence.
Weakening Cardinality Constraints Creates Harder Satisfiability Benchmarks.
J. Exp. Algorithmics Article 1.4, 20, 2015.

R. Sebastiani and P. Trentin.
OptiMathSAT: A Tool for Optimization Modulo Theories.
In Daniel Kroening and Corina S. Păsăreanu, editors, *Computer Aided Verification: 27th International Conference, CAV 2015, San Francisco, CA, USA, July 18-24, 2015, Proceedings, Part I*, pages 447–454, Cham, 2015. Springer International Publishing.