ArgoSMTe: SMT-LIB 2.0 compliant expression library

Milan Banković
milan@matf.bg.ac.rs

Department of Computer Science
Faculty of Mathematics
University of Belgrade, Serbia

Pragmatics of SAT 2012, June 16, Trento, Italy
Outline

1. Introduction
2. ArgoSMTe library
3. Conclusions
SMT solving

**SMT problem:**
- is a problem of deciding whether a first order formula is **satisfiable** with respect to some given **background theory**
- is solved using the procedures called **SMT solvers**
- a great number of different SMT solvers exist

**Issues:**
- **external incompatibility:** solvers use different input languages, produce different **outputs**
- **internal incompatibility:** solvers have incompatible implementations of **expressions** (terms and formulae)
- **interface incompatibility:** solvers have incompatible **Application Programming Interfaces (APIs)**
- thus, **cooperation** within the community is made **difficult**
SMT-LIB 2.0

SMT-LIB is an international effort in:

- providing a **standard language** for rigorous descriptions of first-order theories used in SMT
- providing a **standard language** for SMT solvers’ input and output
- providing a large **library of benchmarks** for testing SMT solvers

**History and Credits**

- Developed since **2003**.
- **Version 1.1** (2005)
- **Version 1.2** (2006) supported by most SMT solvers
- **Version 2.0** (2010) still not fully supported by some solvers
- Joint work of three work groups led by Cesare Tinelli, Clark Barrett, and Aaron Stump
Why expression library is important?

- Implementation of expressions is usually the first step in development of SMT solver
- The component that is most dependant on the input language
- Must be flexible enough to support all the standard features
- Must be extensible, to allow further development of the solver

Unresolved issues:

- Expression implementations in SMT solvers do not fully support the standard
- Intermixing codebases between SMT solvers in not possible
- Tools that use SMT solvers must implement support for different SMT solvers API’s
The solution is:

- Standardization and specification of SMT expression library that fully supports all the features of SMT-LIB 2.0
- Providing a reference implementation of such library specification
- It should include all parts that are common to all SMT solvers (sorts, expressions, signatures, syntax checking, definitions of standard theories and logics, API, parser)
- It should provide an easy and uniform way to communicate with the decision procedures that are developed on top of the library
ArgoSMTe features:

- developed in **standard C++** (GNU/Linux, g++)
- developed using well-known **design patterns** of object oriented programming
- easy to use, develop, understand, extend
- is a **free software** (GNU/GPL licensed)
- strong adherence to the **SMT-LIB 2.0** standard
Overall structure of ArgoSMTe

- **ExpressionNode**
  - getFactory()
  - getSymbol()
  - getSort()
  - getAttributes()
  - expandExpression()
  - inferSort()
  - ...

  typedef shared_ptr<ExpressionNode> Expression;

- **ExpressionFactory**
  - createExpression()

- **SortNode**
  - getFactory()
  - getSymbol()
  - expandSort()
  - ...

  typedef shared_ptr<SortNode> Sort;

- **Signature**
  - addSortSymbol()
  - addFunctionSymbol()
  - addSpecialConstant()
  - checkSortSymbol()
  - checkFunctionSymbol()
  - checkSpecialConstant()

- **SortFactory**
  - createSort()

- **SMTLibAPI**
  - setLogic()
  - assertExpression()
  - checkSat()
  - ...

- **SolverInterface**
  - checkSat()
  - getProof()
  - getValue()
  - ...

- **LogicDeclaration**
  - createSignature()
  - checkExpression()

- **TheoryDeclaration**
  - createSignature()
At the low level:

- **Basic types**: special constants, symbols, variables
- **Expressions**: common subexpression sharing
- **Sorts**: also have expression-like structure
- **Shared pointers** are used as handles to sorts and expressions
- **Signatures**: store declarations of sort and function symbols
- **Signature combination and expansion**
- **Well-sortedness checking and sort inference**
Intermediate level

At the intermediate level:

- **Theory declarations**: simple way to define signatures for standard theories
- **Logic declarations**: define signatures for standard logics
- Makes the library **easy to use**, in case of standard logics and theories
- Support for **future extensions** of the standard
**At the high level:**

- **Standard API:** the operation of the solver can be completely driven by **invoking API methods**
- **Parser:** can read commands from **SMT-LIB 2.0 scripts** instead of calling API methods programmatically
- **API class** fully implements commands that **do not need solver** (set-option, set-logic, declare-fun, declare-sort, etc.)
- **Solver interface:** defines standard interface to **connect** with the solver’s code (i.e. to **invoke** the decision procedure, when needed)
Instead of conclusions:

A simple demonstration: EUF theory solver

The library is available at:

http://www.matf.bg.ac.rs/~milan/software/argosmte/
THANK YOU