SMT-LIB 2.0 Theories and Logics

Hanwen Wu, Wenxin Feng

April 21, 2013

1 Theory

In the following, we are going to present some abstract informal definition of different theories in SMT-LIB 2.0. Note that the Core Theory is included in all other theories by default.

In all the figures, function symbols will only be applied to well-sorted terms according to their own function ranks/signatures/definitions.

1.1 Core

Core Theory is all about boolean sort and boolean functions/constants. It is the very base for all other theories.

sort	α	::=	`bool`
function	f	::=	**true** : `bool` | **false** : `bool`
		|	(**not** `bool`) : `bool` | (**and** `bool bool`) : `bool`
		|	(**or** `bool bool`) : `bool`
		|	(**xor** `bool bool`) : `bool`
		|	($\Rightarrow$ `bool bool`) : `bool` | (= α α) : `bool`
		|	(**distinct** α α) : `bool` | (**ite** `bool` α α) : α
term	t	::=	**true** | **false**
		|	(**not** t) | (**and** t t) | (**or** t t) | (**xor** t t)
		|	($\Rightarrow$ t t) | (= t t) | (**distinct** t t) | (**ite** t t t)

Table 1: Core Theory

1.2 Integer Theory

Integer Theory defines the integer domain, and operations over integers.

1.3 Fixed-Size Bit Vectors Theory

This theory declaration defines a core theory for fixed-size bitvectors where the operations of concatenation and extraction of bitvectors as well as the usual logical and arithmetic operations are overloaded[1].

2 Logic

2.1 Quantifier-Free Uninterpreted Functions

Closed quantifier-free formulas built over an arbitrary expansion of the Core signature with free sort and function symbols [1]. Users can define there own sorts and function symbols, but all of them are abstract. Functions can

sort	α	::=	`bool` | `int`
function	f	::=	...
		|	$\mathbb{Z}$: `int`
		|	(− `int`) : `int` | (− `int` `int`) : `int`
		|	(+ `int` `int`) : `int` | (× `int` `int`) : `int`
		|	(**div** `int` `int`) : `int` | (**mod** `int` `int`) : `int`
		|	(**abs** `int`) : `int`
		|	(⩽ `int` `int`) : `bool` | (< `int` `int`) : `bool`
		|	(⩾ `int` `int`) : `bool` | (> `int` `int`) : `bool`
		|	((_ **divisible** n) `int`) : `bool` (n is a positive integer)
term	t	::=	...
		|	... $-1, 0, 1$ ...
		|	$(-\ t) \mid (-\ t\ t) \mid (+\ t\ t) \mid (\times\ t\ t)$
		|	(**div** $t\ t$) | (**mod** $t\ t$) | (**abs** t)
		|	$(\leqslant t\ t) \mid (<\ t\ t) \mid (\geqslant t\ t) \mid (>\ t\ t)$
		|	((_ **divisible** n) t)

Table 2: Integer Theory

contain variables, but they must be bounded by **let** binder, so that the formulas are closed.

2.2 Quantifier-Free Linear Integer Arithmetic

Closed quantifier-free formulas built over an arbitrary expansion of the Integer Theory with free *constant* symbols, but whose terms of sort `int` are all linear [1]. Note that user can only define constants, not arbitrary functions who take one or more arguments. User can't define sort either. Also, non-linear functions like **div**, **mod**, **abs** and non-linear $\times$ are not allowed.

References

[1] Clark Barrett, Aaron Stump, and Cesare Tinelli. The satisfiability modulo theories library (SMT-LIB). *www.smtlib.org*, 2010.

sort	α	::=	`bool`
		|	(_ `BitVec` m) (m is a positive integer, we use `bv` for short)
function	f	::=	...
		|	`#b`X : `bv` (all binary constants)
		|	`#x`X : `bv` (all hexadeximal constants)
		|	(**concat** `bv bv`) : `bv`
		|	((_ **extract** i j) `bv`) : `bv` (i, j specify the range)
		|	(**bvnot** `bv`) : `bv` | (**bvneg** `bv`) : `bv`
		|	(**bvand** `bv bv`) : `bv` | (**bvor** `bv bv`) : `bv`
		|	(**bvadd** `bv bv`) : `bv` | (**bvmul** `bv bv`) : `bv`
		|	(**bvudiv** `bv bv`) : `bv` | (**bvurem** `bv bv`) : `bv`
		|	(**bvshl** `bv bv`) : `bv` | (**bvlshr** `bv bv`) : `bv`
		|	(**bvult** `bv bv`) : `bool`
term	t	::=	...
		|	`#b`X (all binary constants)
		|	`#x`X (all hexadeximal constants)
		|	(**concat** t t) | ((_ **extract** i j) t)
		|	(**bvnot** t) | (**bvneg** t) | (**bvand** t t) | (**bvor** t t)
		|	(**bvadd** t t) | (**bvmul** t t) | (**bvudiv** t t) | (**bvurem** t t)
		|	(**bvshl** t t) | (**bvlshr** t t) | (**bvult** t t)

Table 3: Fixed-Size Bit Vectors Theory

sort	α	::=	... | α' (α^*) (user defined, abstract)
function	f	::=	... | (f' α^*) : α (user defined, abstract)
term	t	::=	...
		|	(**let** (bindings$^+$) t)
		|	(f t^*)

Table 4: QF-UF Logic

sort	α	::=	`bool` | `int`
function	f	::=	$\ldots \mid f' : \alpha$ (user defined constant)
term	t	::=	$\ldots$
		|	$\ldots \quad -1, 0, 1 \quad \ldots$
		|	$(-\ t)$ | $(-\ t\ t)$ | $(+\ t\ t)$
		|	$(\times\ c\ t)$ | $(\times\ t\ c)$ (c is an integer literal)
		|	$(\leqslant t\ t)$ | $(<\ t\ t)$ | $(\geqslant t\ t)$ | $(>\ t\ t)$
		|	((_ **divisible** n) t)
		|	(**let** (bindings$^+$) t)

Table 5: QF-LIA Logic