Software Verification with Satisfiability Modulo Theories
- Introduction -

Ming-Hsien Tsai

Institute of Information Science
Academia Sinica

FLOLAC 2017

Reference book: Aaron R. Bradley and Zohar Manna. The Calculus of Computation. Springer 2007

Software with Bugs

- Have you ever seen this?

```
MacBook-Pro ~ $ ./a.out
Segmentation fault: 11
MacBook-Pro ~ $ █
```

- How to avoid it?

- Programmers usually write assertions for debugging and testing.

C Assertions

- When an assertion is violated, the program aborts immediately (if the program is compiled with NDEBUG undefined).

```c
#include<assert.h>

int div(int x, int y) {
  assert(y != 0);
  return x / y;
}

int main(void) {
  int x = 10;
  int y = 0;
  int z = div(x, y);
  return 0;
}
```

```
MacBook-Pro ~ $ ./a.out
Assertion failed: (y != 0), function div, file a.c, line 4.
Abort trap: 6
```

Example 1

- Will the assertion be violated?

```
#include <assert.h>
#include <stdio.h>

int main(void) {
  int x;
  scanf("%d", &x);
  while (x < 10) {
    x++;
  }
  assert(x > 0);
}
```

Example taken from Yu-Fang's slides

Example 1

- Will the assertion be violated?

No

```c
#include <assert.h>
#include <stdio.h>

int main(void) {
  int x;
  scanf("%d", &x);
  while (x < 10) {
    x++;
  }
  assert(x > 0);
}
```

Example taken from Yu-Fang's slides

Example 2

- Will the assertion be violated?

```c
#include <assert.h>
#include <stdio.h>

int main(void) {
  int x;
  scanf("%d", &x);
  while (x < 10) {
    x--;
  }
  assert(x > 0);
}
```

Example taken from Yu-Fang's slides

Example 2

- Will the assertion be violated?

No

```c
#include <assert.h>
#include <stdio.h>

int main(void) {
  int x;
  scanf("%d", &x);
  while (x < 10) {
    x--;
  }
  assert(x > 0);
}
```

Example taken from Yu-Fang's slides

Example 3

- Will the assertion be violated?

```c
#include <assert.h>
#include <stdio.h>

int main(void) {
  int x;
  scanf("%d", &x);
  while (x < 4324358) {
    x--;
  }
  assert(x > 4324358);
}
```

Example taken from Yu-Fang's slides

Example 3

- Will the assertion be violated?

Yes

```
#include <assert.h>
#include <stdio.h>

int main(void) {
  int x;
  scanf("%d", &x);
  while (x < 4324358) {
    x--;
  }
  assert(x > 4324358);
}
```

Example taken from Yu-Fang's slides

Example 4

- Will the assertion be violated?

```
void A(bool h, bool g) {
  h = !g;
  g = B(g, h);
  g = B(g, h);
  assert(g);
}

bool B(bool a1, bool a2) {
  if (a1)
    return B(a2,a1);
  else
    return a2;
}
```

Example taken from Yu-Fang's slides

Example 4

- Will the assertion be violated?

No

```
void A(bool h, bool g) {
  h = !g;
  g = B(g, h);
  g = B(g, h);
  assert(g);
}

bool B(bool a1, bool a2) {
  if (a1)
    return B(a2,a1);
  else
    return a2;
}
```

Example taken from Yu-Fang's slides

Software Verification

- Given a program with assertions, automatically verify if any assertion could be violated.
- There are various techniques:
  - Model checking
  - Craig interpolation
  - Satisfiability modulo theories (SMT)
  - ...

Verification With SMT

- Convert a program with assertions into SMT formulas such that an assertion is violated if an SMT formula is satisfiable.
- Solve satisfiability of the SMT formulas by SMT solvers.

A Simple Example

Input Program

```
int main(void) {
  int x;
  if (x < 10)
    x = x - 1;
  assert(x != 9);
  return 0;
}
```

Static Single Assignment (SSA)

```
int main(void) {
  int x0;
  if (x0 < 10)
    x1 = x0 - 1;
  x2 = Φ(x0, x1);
  assert(x2 != 9);
  return 0;
}
```

SMT Formula

$$(x0 < 10 \land x1 = x0 - 1 \land x2 = x1 \land x2 = 9) \lor$$

$$(x0 \geq 10 \land x2 = x0 \land x2 = 9)$$

Example taken from Yu-Fang's slides

Recall: First-Order Logic

- Terms
  - Variables: x, y, ...
  - Function symbols: f, g, ...
- Formulas
  - Predicate symbols: p, q, ...
  - Logical operators: $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$
  - Quantifications: $\forall$, $\exists$

Recall: First-Order Logic (cont'd)

- A FOL formula is interpreted under a model and an environment.
  - Model: gives the meanings of function symbols and predicate symbols
  - Environment: gives the values of variables

Signature

- A collection of non-logical symbols excluding variables
- Examples:
  - $(0, S, +, =)$
  - $(\varnothing, \subseteq)$

First-Order Theories

- A *first-order theory* T is defined by the following two components.
  - Signature Σ
  - Axioms A: set of closed Σ-formula
- Σ*-formula*: a FOL formula constructed from the signature Σ plus variables, logical connectives, and quantifiers

Validity and Satisfiability

- A *T-model* is a model that satisfies the axioms of a first-order theory T.
- A Σ-formula φ is valid in the theory T, or *T-valid*, if every T-model M satisfies φ.
- We write $T \vDash \varphi$ if φ is T-valid.
- A Σ-formula φ is satisfiable in T, or *T-satisfiable*, if there is a T-model M that satisfies φ.

Complete, Consistent, and Equivalent

- A theory T is *complete* if for every closed Σ-formula φ, $T \models \varphi$ or $T \models \neg\, \varphi$.

- A theory is *consistent* if there is at least one T-model.

- Two formulas φ and ψ are equivalent in T, or T-equivalent, if $T \models \varphi \leftrightarrow \psi$.

Fragment and Decidable

- A *fragment* of a theory is a syntactically-restricted subset of formulae of the theory.
- Example:
  - quantifier-free fragment
- A theory T is *decidable* if $T \vDash \varphi$ is decidable for every Σ-formula φ.

Union of Theories

- The union $T_1 \cup T_2$ of two theories T_1 and T_2 has signature $\Sigma_1 \cup \Sigma_2$ and axioms $A_1 \cup A_2$.

- $(T_1 \cup T_2)$-interpretation is both a T_1-interpretation and a T_2-interpretation.

- A formula that is T_1-valid or T_2-valid is $(T_1 \cup T_2)$-valid.

- A formula that is $(T_1 \cup T_2)$-satisfiable is both T_1-satisfiable and T_2-satisfiable.

Decidability

- FOL is undecidable in general.
- There are some important theories or fragment of theories that are decidable.
  - Equality
  - Peano arithmetic
  - Presburger arithmetic
  - Linear integers
  - Recursive data structures
  - Arrays

Binary Relation

- Let's talk about binary relations before introducing the equality theory.
- Consider a set S and a binary relation R over S
- For two elements $s_1,\ s_2 \in S$, either $s_1 R s_2$ or $\neg(s_1 R s_2)$

S: Integers	S: Humans
R: <	R: IsChildOf

Equivalence Relation

- The relation R is an *equivalence relation* if it is
  - *reflexive*: $\forall s \in S.\ sRs$;
  - *symmetric*: $\forall s_1, s_2 \in S.\ s_1Rs_2 \rightarrow s_2Rs_1$;
  - *transitive*: $\forall s_1, s_2, s_3 \in S.\ s_1Rs_2 \wedge s_2Rs_3 \rightarrow s_1Rs_3$

$$=,\quad \cdot \equiv \cdot \pmod{c}$$

Congruence Relation

- The relation R is a *congruence relation* if it additionally obeys congruence: for every n-ary function f,

$$\forall S, T.\ (\wedge_{i=1 \text{ to } n}\ s_i R t_i) \rightarrow f(S) R f(T)$$

Capital S and T are vectors of variables

Equality T_E

- $\Sigma_E : \{=,\ a,\ b,\ c,\ ...,\ f,\ g,\ h,\ ...,\ p,\ q,\ r,\ ...\}$ contains
  - $=$, a binary predicate; and
  - all constants, function and predicate symbols.
- Also called equality with uninterpreted functions (EUF)

Axioms of T_E

1. Reflexivity: $\forall x.\ x = x$

2. Symmetry: $\forall x, y.\ x = y \rightarrow y = x$

3. Transitivity: $\forall x, y, z.\ x = y \rightarrow y = z \rightarrow x = z$

4. Function congruence: for n-ary ($n > 0$) function symbol f,

   - $\forall \underline{x}, \underline{y}.\ (\wedge_{i=1}{}^{n}\ x_i = y_i) \rightarrow f(\underline{x}) = f(\underline{y})$

5. Predicate congruence: for n-ary ($n > 0$) predicate symbol f,

   - $\forall \underline{x}, \underline{y}.\ (\wedge_{i=1}{}^{n}\ x_i = y_i) \rightarrow (p(\underline{x}) \leftrightarrow p(\underline{y}))$

$\underline{x}$: list of variables x_1, ..., x_n

Properties of T_E

- Axioms 1, 2, and 3 state that $=$ is a equivalence relation.
- All the axioms assert that $=$ is a congruence relation.
- T_E is undecidable.
  - Every FOL formula can be encoded as a Σ_E formula by replacing $=$ with a fresh symbol.
- Quantifier-free fragment of T_E is both efficiently decidable.

An Example of T_E

- $\varphi : a = b \wedge b = c \rightarrow g(f(a), b) = g(f(c), a)$ is T_E-valid
- Assume there is a T_E-model M such that $M \not\models \varphi$

1. $M \not\models \varphi$
2. $M \models a = b \wedge b = c$
3. $M \not\models g(f(a), b) = g(f(c), a)$
4. $M \models a = b$
5. $M \models b = c$
6. $M \models a = c$
7. $M \models f(a) = f(c)$
8. $M \models b = a$
9. $M \models g(f(a), b) = g(f(c), a)$
10. $M \models \bot$

Exercise

- Use the semantic method to prove the validity of the following Σ_E-formulae or find a counterexample.
  - $f(x,\ y) = f(y,\ x) \rightarrow f(a,\ y) = f(y,\ a)$
  - $f(g(x)) = g(f(x)) \wedge f(g(f(y))) = x \wedge f(y) = x \rightarrow g(f(x)) = x$

Peano Arithmetic T_{PA}

- $\Sigma_{PA} : \{0,\ 1,\ +,\ \cdot,\ =\}$ where
  - 0 and 1 are constants;
  - $+$ (addition) and $\cdot$ (multiplication) are binary functions ($x \cdot y$ may be written as xy); and
  - $=$ (equality) is a binary predicate.

Axioms of T_{PA}

- Zero: $\forall x.\ \neg(x{+}1 = 0)$
- Successor: $\forall x{,}y.\ x{+}1 = y{+}1 \rightarrow x = y$
- Induction: $P[0] \wedge (\forall x.\ P[x] \rightarrow P[x + 1]) \rightarrow \forall x.\ P[x]$ (an axiom schema)
- Plus Zero: $\forall x.\ x{+}0 = x$
- Plus Successor: $\forall x{,}y.\ x{+}(y{+}1) = (x{+}y) + 1$
- Times Zero: $\forall x.\ x \cdot 0 = 0$
- Times Successor: $\forall x{,}y.\ x{\cdot}(y{+}1) = x{\cdot}y{+}x$

Intended Models of T_{PA}

- The intended models of T_{PA} have domain $\mathbb{N}$ and assignments α_M defining 0, 1, $+$, $\cdot$, and $=$ as we understand them in everyday arithmetic.

- $\alpha_M[0]$ is $0_{\mathbb{N}}$: α_M maps the symbols "0" to $0_{\mathbb{N}} \in \mathbb{N}$;

- $\alpha_M[1]$ is $1_{\mathbb{N}}$: α_M maps the symbols "1" to $1_{\mathbb{N}} \in \mathbb{N}$;

- $\alpha_M[+]$ is $+_{\mathbb{N}}$, addition over $\mathbb{N}$;

- $\alpha_M[\cdot]$ is $\cdot_{\mathbb{N}}$, multiplication over $\mathbb{N}$;

- $\alpha_M[=]$ is $=_{\mathbb{N}}$, equality over $\mathbb{N}$.

Example 1 of T_{PA}

- $3x+5 = 2y$ can be written using the signature Σ_{PA} as:
  - $x+x+x+1+1+1+1+1 = y+y$, or as
  - $(1+1+1)\cdot x+1+1+1+1+1 = (1+1)\cdot y$
- In practice, we write $3x+5 = 2y$ for short.

Example 2 of T_{PA}

- We can encode $>$ and $\geq$ in T_{PA}.
  - $3x+5 > 2y$ is encoded as $\exists z.\ z \neq 0 \wedge 3x+5 = 2y+z$
  - $3x+5 \geq 2y$ is encoded as $\exists z.\ 3x+5 = 2y+z$

$x \neq y$ *abbreviates* $\neg\ (x = y)$

Example 3 of T_{PA}

- $\varphi : \exists x,y,z.\ x \neq 0 \wedge y \neq 0 \wedge z \neq 0 \wedge xx+yy = zz$ is T_{PA}-valid.
- Every $\varphi \in \{\forall x,y,z.\ x \neq 0 \wedge y \neq 0 \wedge z \neq 0 \wedge x^n+y^n \neq z^n : n > 2 \wedge n \in \mathbb{Z}\}$ is T_{PA}-valid. (x^n: n multiplications of x)

Decidability of T_{PA}

- Satisfiability and validity in T_{PA} is undecidable (Gödel's first incompleteness theorem).
- Try a more restricted theory of arithmetic that does not allow multiplication.

Presburger Arithmetic $T_{\mathbb{N}}$

- $\Sigma_{\mathbb{N}}$: $\{0, 1, +, =\}$, where
  - 0 and 1 are constants;
  - $+$ (addition) is a binary function; and
  - $=$ (equality) is a binary predicate.

Axioms of $T_{\mathbb{N}}$

- Zero: $\forall x.\ \neg(x{+}1 = 0)$
- Successor: $\forall x{,}y.\ x{+}1 = y{+}1 \rightarrow x = y$
- Induction: $P[0] \wedge (\forall x.\ P[x] \rightarrow P[x{+}1]) \rightarrow \forall x.\ P[x]$
- Plus Zero: $\forall x.\ x{+}0 = x$
- Plus Successor: $\forall x{,}y.\ x{+}(y{+}1){=}(x{+}y){+}1$

Intended Models of $T_{\mathbb{N}}$

- The intended models of $T_{\mathbb{N}}$ have domain $\mathbb{N}$ and are such that:

    - $\alpha_M[0]$ is $0_{\mathbb{N}} \in \mathbb{N}$;

    - $\alpha_M[1]$ is $1_{\mathbb{N}} \in \mathbb{N}$;

    - $\alpha_M[+]$ is $+_{\mathbb{N}}$, addition over $\mathbb{N}$;

    - $\alpha_M[=]$ is $=_{\mathbb{N}}$, equality over $\mathbb{N}$.

Decidability of $T_{\mathbb{N}}$

- Presburger showed in 1929 that $T_{\mathbb{N}}$ is decidable.
- Validity of $\Sigma_{\mathbb{N}}$ formulas can be decided by procedures for the validity of $\Sigma_{\mathbb{Z}}$ formulas.

Integer Theory $T_{\mathbb{Z}}$

- $\Sigma_{\mathbb{Z}}$: $\{...,-2,-1,0,\ 1,\ 2,\ ...,-3\cdot,-2\cdot,\ 2\cdot,\ 3\cdot,\ ...,\ +,\ -,\ =,\ >\}$, where

  - ... , -2, -1, 0, 1, 2, ... are constants;

  - ... , $-3\cdot$, $-2\cdot$, $2\cdot$, $3\cdot$, ... are unary functions (representing constant coefficients);

  - $+$ and $-$ are binary functions;

  - $=$ and $>$ are binary predicates.

Encoding of $\Sigma_{\mathbb{Z}}$-Formulas

- $\varphi_0 : \forall w,x. \exists y,z.\ x+2y-z-13 > -3w+5$

- $\varphi_1 : \forall w_p,w_n,x_p,x_n.\ \exists y_p,y_n,z_p,z_n.$
  $(x_p-x_n)+2(y_p-y_n)-(z_p-z_n)-13 > -3(w_p-w_n)+5$

- $\varphi_2 : \forall w_p,w_n,x_p,x_n.\ \exists y_p,y_n,z_p,z_n.$
  $x_p+2y_p+z_n+3w_p > x_n+2y_n+z_p+13+3w_n+5$

- $\varphi_3 : \forall w_p,w_n,x_p,x_n.\ \exists y_p,y_n,z_p,z_n.\ \exists u.\ \neg(u=0) \wedge$
  $x_p+y_p+y_p+z_n+w_p+w_p+w_p =$
  $x_n+y_n+y_n+z_p+w_n+w_n+w_n+u+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1$

- φ_0 is $T_{\mathbb{Z}}$-valid if φ_3 is $T_{\mathbb{N}}$-valid

Encoding of $\Sigma_{\mathbb{N}}$-Formulas

- $\varphi_1 : \forall x.\ \exists y.\ x = y{+}1$
- $\varphi_2 : \forall x.\ x \geq 0 \rightarrow \exists y.\ y \geq 0 \wedge x = y{+}1$
- φ_1 is $T_{\mathbb{N}}$-valid if φ_2 is $T_{\mathbb{Z}}$-valid.

Example of $T_{\mathbb{Z}}$

- $\varphi : \forall x, y, z.\ x > z \wedge y \geq 0 \rightarrow x+y > z$ is $T_{\mathbb{Z}}$-valid.
- Assume there is a $T_{\mathbb{Z}}$-model M such that $M \nvDash \varphi$

1. $M \nvDash \varphi$
2. $M_1 : M[x \rightarrow v_x, y \rightarrow v_y, z \rightarrow v_z] \nvDash x > z \wedge y \geq 0 \rightarrow x+y > z$
3. $M_1 \vDash x > z \wedge y \geq 0$
4. $M_1 \nvDash x+y > z$
5. $M_1 \vDash \neg(x+y > z)$
6. No v_x, v_y, and v_z can satisfy $v_x > v_z \wedge v_y \geq 0 \wedge \neg(v_x + v_y > v_z)$ by querying the theory $T_{\mathbb{Z}}$
7. $M_1 \vDash \perp$

List Theory $Tcons$

- $\Sigma cons$: $\{cons,\ car,\ cdr,\ atom,\ =\}$, where
  - $cons$ is a binary function (constructor): $cons(a,\ b)$ represents the list constructed by concatenating a to b;
  - car is a unary function (left projector): $car(cons(a,\ b)) = a$;
  - cdr is a unary function (right projector): $cdr(cons(a,\ b)) = b$;
  - $atom$ is a unary predicate: $atom(x)$ is true iff x is a single-element list; and
  - $=$ (equality) is a binary predicate.

Axioms of $Tcons$

- The axioms of reflexivity, symmetry, and transitivity of T_E
- Instantiations of the function congruence axiom schema for $cons$, car, and cdr:
  - $\forall x_1, x_2, y_1, y_2.\ x_1 = x_2 \land y_1 = y_2 \rightarrow cons(x_1,\ y_1) = cons(x_2,\ y_2)$
  - $\forall x, y.\ x = y \rightarrow car(x) = car(y)$
  - $\forall x, y.\ x = y \rightarrow cdr(x) = cdr(y)$
- An instantiation of the predicate congruence axiom schema for $atom$:
  - $\forall x, y.\ x = y \rightarrow (atom(x) \leftrightarrow atom(y))$

Axioms of $Tcons$ (cont'd)

- $\forall x, y.\ car(cons(x,\ y)) = x$
- $\forall x, y.\ cdr(cons(x,\ y)) = y$
- $\forall x.\ \neg atom(x) \rightarrow cons(car(x),\ cdr(x)) = x$
- $\forall x, y.\ \neg atom(cons(x,\ y))$

Decidability of $Tcons$

- $Tcons$ is undecidable.
- The following fragment of $Tcons$ is decidable.
  - Quantifier-free fragment of $Tcons$.
  - $Tcons^+$: lists are acyclic

Array Theory T_A

- $\Sigma_A : \{\cdot[\cdot], \cdot\langle\cdot \lhd \cdot\rangle, =\}$, where
  - $a[i]$ (read) is a binary function: $a[i]$ represents the value of array a at position i;
  - $a\langle i \lhd v\rangle$ (write) is a ternary function: $a\langle i \rhd v\rangle$ represents the modified array a in which position i has value v; and
  - $=$ (equality) is a binary predicate defined only for array elements

Axioms of T_A

- The axioms of reflexivity, symmetry, and transitivity of T_E;
- Array Congruence: $\forall a,i,j.\ i = j \rightarrow a[i] = a[j]$
- Read-Over-Write 1: $\forall a,v,i,j.\ i = j \rightarrow a\langle i \triangleleft v\rangle[j] = v$
- Read-Over-Write 2: $\forall a,v,i,j.\ i \neq j \rightarrow a\langle i \triangleleft v\rangle[j] = \mathrm{a}[j]$

Example of T_A

- $\varphi : a[i] = e \rightarrow \forall j.\ a\langle i \triangleleft e\rangle[j] = a[j]$ is T_A-valid.
- Assume there is a T_A-model M such that $\mathrm{M} \not\models \varphi$.

1. $M \not\models \varphi$
2. $M \models a[i] = e$
3. $M \not\models \forall j.\ a\langle i \triangleleft e\rangle[j] = a[j]$
4. $M_1 : M[j \rightarrow v] \not\models a\langle i \triangleleft e\rangle[j] = a[j]$
5. $M_1 \models a\langle i \triangleleft e\rangle[j] \neq a[j]$
6. $M_1 \models i = j$
7. $M_1 \models \mathrm{a}[i] = a[j]$
8. $M_1 \models a\langle i \triangleleft e\rangle[j] = e$
9. $M_1 \models a\langle i \triangleleft e\rangle[j] = a[j]$
10. $M_1 \models \perp$

Equality in T_A

- $\varphi : a[i] = e \rightarrow a\langle i \triangleleft e\rangle = a$ is not T_A-valid

- $\varphi' : a[i] = e \rightarrow \forall j.\ a\langle i \triangleleft e\rangle[j] = a[j]$ is T_A-valid

Decidability of T_A

- T_A-validity is undecidable.
- Decidable fragments of T_A:
  - Quantifier-free fragment of T_A
  - $T_A{}^=$: T_A plus the extensionality axiom
    - $\forall a,b.\ (\forall i.\ a[i] = b[i]) \leftrightarrow a = b$

Decidability of Theories

Theory	Description	Full	QFF
T_E	equality	no	yes
T_{PA}	Peano arithmetic	no	no
$T_{\mathbb{N}}$	Presburger arithmetic	yes	yes
$T_{\mathbb{Z}}$	linear integers	yes	yes
$T_{\mathbb{R}}$	reals (with $\cdot$)	yes	yes
$T_{\mathbb{Q}}$	rationals (without $\cdot$)	yes	yes
T_{RDS}	recursive data structures	no	yes
T_{RDS^+}	acyclic recursive data structures	yes	yes
T_A	arrays	no	yes
$T_{A^=}$	arrays with extensionality	no	yes

Combination Theories

- In practice, formulas may span multiple theories.
  - $\forall a,i,j,k,v.\ a[i] = v \land j = i{+}k \rightarrow a[j] = v$
- Given some decidable theories, is a formula spanning these theories still decidable?

Combination Theories

- In practice, formulas may span multiple theories.
  - $\forall a, i, j, k, v.\ a[i] = v \wedge j = i + k \rightarrow a[j] = v$
- Given some decidable theories, is a formula spanning these theories still decidable? **Yes under some constraints**

Combination Theories

- In practice, formulas may span multiple theories.
  - $\forall a, i, j, k, v.\ a[i] = v \wedge j = i + k \rightarrow a[j] = v$
- Given some decidable theories, is a formula spanning these theories still decidable? **Yes under some constraints**
- Nelson-Oppen approach

Nelson-Oppen Approach

- Given two theories T_1 and T_2 such that $\Sigma_1 \cap \Sigma_2 = \{=\}$, the combined theory $T_1 \cup T_2$ has signature $\Sigma_1 \cup \Sigma_2$ and axioms $A_1 \cup A_2$.

- The quantifier-free fragment of $T_1 \cup T_2$ is decidable if

  - satisfiability in the quantifier-free fragment of T_1 is decidable;

  - satisfiability in the quantifier-free fragment of T_2 is decidable; and

  - certain technical requirements are met.

SMT Solvers

- Solvers (partially listed)
  - Z3 (https://github.com/Z3Prover/z3)
  - CVC4 (http://cvc4.cs.stanford.edu/web/)
  - Yices (http://yices.csl.sri.com)
  - STP (http://stp.github.io)
- Most SMT solvers support SMT-LIB format (http://smtlib.cs.uiowa.edu).
- There are SMT competitions (www.smtcomp.org).