Формальные методы в программной инженерии

Satisfiability Modulo Theories

задача выполнимости формул в теориях

Зубарев Алексей Юрьевич
zub.rev@yandex.ru

2023

SATisfiability

задача выполнимости булевых формул

Пример булевой формулы

$\neg(a \vee b \Rightarrow d) \wedge ((c \Rightarrow d) \Rightarrow (a \wedge b)) \wedge \neg d$

- a, b, c, d – пропозициональные (булевы) переменные;
- $\wedge$, $\vee$, $\neg$, $\Rightarrow$ – логические операторы;
- (,) – вспомогательные символы.

Булева формула

- Пропозициональная переменная – булева формула.

- Если Φ и Ψ – булевы формулы, то $(\Phi \wedge \Psi)$, $(\Phi \vee \Psi)$, $(\Phi \Rightarrow \Psi)$, $\neg \Phi$ – булевы формулы.

- Других булевых формул нет.

Булевы формулы:

$a \wedge b \wedge c$

$a \wedge (b \Rightarrow c)$

$a \Rightarrow \neg c$

$\neg(b \Rightarrow d) \vee a$

Не булевы формулы:

$a \wedge \wedge b$

$\exists a \neg a \wedge a$

$a \geq 5 \wedge c$

$\forall x \exists y\ x \neq y$

Интерпретация

Пусть **s** – функция (интерпретация),

- q – истина в **s**, если **s**(q) – истина;
- q – ложь в **s**, если **s**(q) – ложь;
- $\neg$Φ – истина в **s**, если Φ – ложь в **s**;
- (Φ $\wedge$ Ψ) – истина в **s**, если Φ – истина в **s** и Ψ – истина в **s**;
- (Φ $\vee$ Ψ) – истина в **s**, если Φ – истина в **s** или Ψ – истина в **s**;
- (Φ $\Rightarrow$ Ψ) – истина в **s**, если Φ – ложь в **s** или Ψ – истина в **s**.

Выполнимость булевой формулы

Формула ***выполнима***, если существует интерпретация, в которой она принимает значение «истина».

$a \wedge b \wedge c$

$a \wedge \neg a$

Задача SAT

Для заданной булевой формулы Ф определить, является ли она выполнимой.

Алгоритмы решения задачи SAT:

- Алгоритм с возвратом
- Алгоритм DPLL
- Алгоритм CDCL

Алгоритм с возвратом

```
SAT(Φ) {
    если в Φ нет переменных {
        return (значение Φ);
    }
    иначе {
        пусть x переменная из Φ
        Φ' = [Φ]x=истина;
        Φ'' = [Φ]x=ложь;
        return (SAT(Φ') or SAT(Φ''));
    }
}
```

Satisfiability **M**odulo **T**heories

задача выполнимости формул в теории

Пример формулы логики первого порядка

$a \neq b \wedge \exists\, c\, (\forall\, d\, (f(c + a, d) = b) \Rightarrow (a = b)) \wedge e$

- a, b, c, d, e – переменные;
- +, f – функциональные символы;
- =, ≠ – предикатные символы;
- ∧, ∃, ∀, ⇒ – логические операторы;
- (,), , – вспомогательные символы.

} **сигнатура**

Сигнатура Σ

Упорядоченная тройка $\Sigma = <R, F, \mu>$ называется **сигнатурой**, если выполняются следующие условия:

- множества R (предикатных символов) и F (функциональных символов) не имеют общих элементов;
- μ является отображением множества $R \cup F$ в $\mathbb{N}$ (отображение местности, или арности).

Формула логики первого порядка

Терм – символ переменной либо $f(t_1, ..., t_n)$, где f – функциональный символ, $t_1 ... t_n$ термы.

Атом (атомарная формула) – $p(t_1, ..., t_n)$, где p – предикатный символ, $t_1 ... t_n$ термы.

Формула – атом или $\neg\Phi$, $(\Phi \wedge \Phi_1)$, $(\Phi \vee \Phi_1)$, $(\Phi \Rightarrow \Phi_1)$, $\forall x\ (\Phi)$, $\exists x\ (\Phi)$, где Φ, Φ_1 – формулы, x – переменная.

Свободные переменные формулы

Квантор связывает ту переменную, которая следует за ним ($\forall x\,(\Psi)$, $\exists x\,(\Psi)$).

Вхождение переменной в области действия квантора, связывающего эту переменную, называется **связанным**.

Вхождение переменной в формулу, не являющееся связанным, называется **свободным**.

Переменная называется **свободной** , если она имеет свободное вхождение в формулу.

Теория

Предложение – формула без свободных переменных.

Теория – множество предложений.

Выполнимость формулы

Пусть **M** - модель (множество и набор определенных на нем предикатов и функций) и **s** – функция (подстановка), отображающая переменные формулы в множество **M**.

- $p(t_1, ..., t_n)$ – истина в **(M, s)**, если $[p_{\mathbf{M}}(t_1, ..., t_n)]_{\mathbf{s}}$ – истина в **(M, s)**;
- $p(t_1, ..., t_n)$ – ложь в **(M, s)**, если $[p_{\mathbf{M}}(t_1, ..., t_n)]_{\mathbf{s}}$ – ложь в **(M, s)**;
- $\neg\Phi$ – истина в **(M, s)**, если Φ – ложь в **(M, s)**;
- $(\Phi \wedge \Psi)$ – истина в **(M, s)**, если Φ – истина и Ψ – истина в **(M, s)**
- $(\Phi \vee \Psi)$ – истина в **(M, s)**, если Φ – истина или Ψ – истина в **(M, s)**;
- $(\Phi \Rightarrow \Psi)$ – истина в **(M, s)**, если Φ – ложь или Ψ – истина в **(M, s)**;

Выполнимость формулы

- $\forall x$ (Ф) – истина в **(M, s)**, если Ф – истина во всех s′, которые отличаются от **s** только значением x;
- $\exists x$ (Ф) – истина в **(M, s)**, если Ф – истина в некоторой s′, которая отличается от **s** только значением x.

Формула ***выполнима***, если существует модель **M** с подстановкой **s**, относительно которых она принимает значение «истина».

Задача SMT

Для заданной формулы Φ сигнатуры Σ определить является ли она выполнимой в теории T сигнатуры Σ.

Примеры SMT решателей:

- Z3
- CVC4
- Yices 2

Схема алгоритма решения задачи SMT

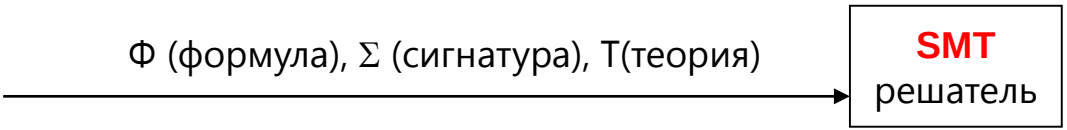
SMT(Ф, Σ, T) :

1. Получаем **Ф'**, заменой всех атомарных формул на пропозициональные переменные.
2. Если **SAT(Ф') = UNSAT**, то **SMT(Ф, Σ, T) = UNSAT**.
3. Иначе существует интерпретация **s** для **Ф'**.
4. Проверяем интерпретацию **s** на совместимость с формулой **Ф** относительно теории **T**.
5. Если формула совместна, то **SMT(Ф, Σ, T) = SAT**.
6. Иначе повторно решаем задачу **SAT(Ф')** исключая рассмотренную интерпретацию **s** (шаг 2).

SMT-LIB

стандарт для SMT решателей

SMT-LIB



Задача (формула), постоянно меняется, а теория и сигнатура практически всегда неизменна. **SMT-LIB** предлагает единый формат ввода вывода для решателей с набором наиболее используемых теорий и сигнатур.

Язык SMT-LIB

Lisp-подобный язык.

Каждая конструкция является **S-выражением** и записывается в виде списка.

- (**+** a b) ~ a **+** b
- (**f** x y) ~ **f**(x, y)
- (**=** (**f** x) (**f** (**+** y 1))) ~ **f**(x) **=** **f**(y **+** 1)

Теории SMT-LIB

Core theory – булевы числа;

Ints theory – целые числа;

Reals theory – вещественные числа;

Reals_Ints theory – целые с вещественными числами;

ArraysEx theory – массивы;

Fixed_Size_BitVectors theory – битовые вектора.

Операции сравнения

<, **>** , **>=** , **<=** , **=**

(**>** x y)

Логические операции

not, **or** , **xor** , **and** , **=>**

(**and** (**>** x y) (**>** y z))

Тернарная операция

ite

(**ite** (**>** y z) 1 0)

Арифметические операции

+, **-**, *****, **/**, **div**, **mod**, **abs** (**mod** x y)

Чтение и запись для массива

select, **store** (**select** a i)

Битовые операции

bvand, **bvor**, **bvadd**, **bvmul**, (**bvor** v1 v2)

bvudiv, **bvurem**, **bvshl**, **bvlshr**...

Типы данных (sort)

Bool, **Int**, **Real**, **Array**, **BitVec**

Неинтерпретируемая функция

declare-fun (**declare-fun** matrix (Int Int) Int)

Функция с выражением

define-fun (**define-fun** sqr ((i Int)) Int (* i i))

Кванторы

forall, **exists** (**exists** ((x Int) (y Int)) (>x y))

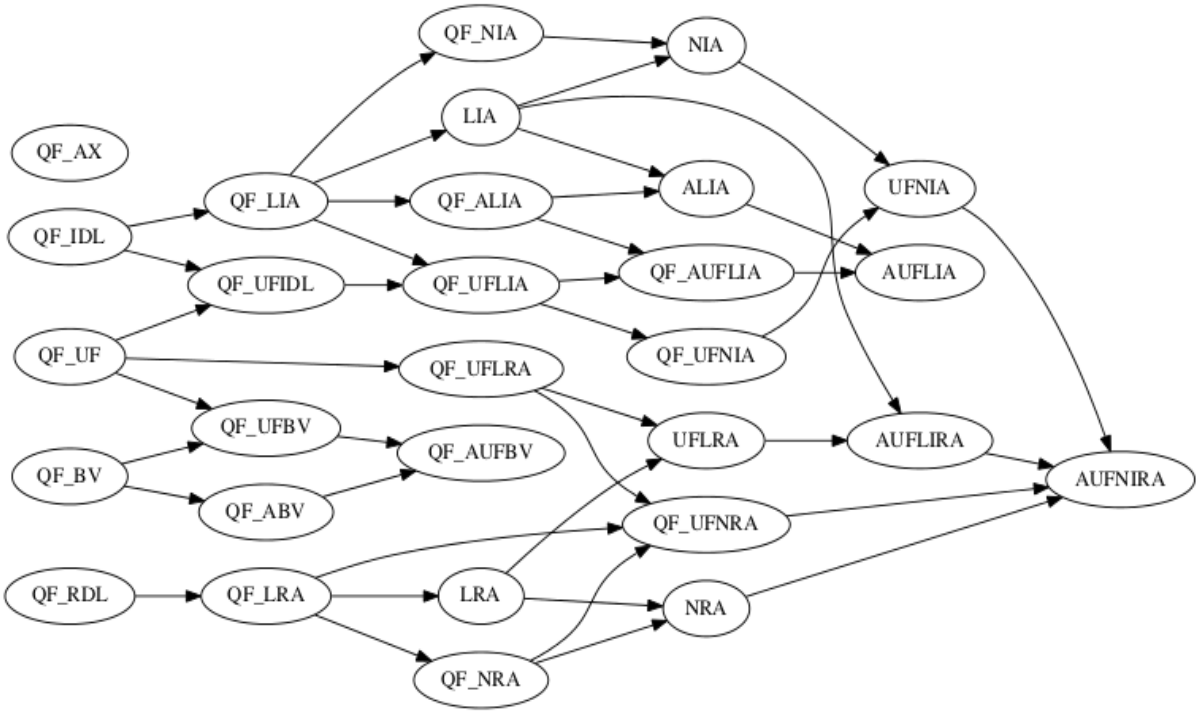
Утверждения

assert (**assert** (> a 0))

Управление решателем

check-sat, **get-value** (**get-value** (* x y))

Логики SMT-LIB



Логики SMT-LIB

QF для формул без кванторов;

A или **AX** для теории ArraysEx;

BV для теории Fixed_Size_BitVectors;

IA для теории Ints;

RA для теории Reals;

IRA для теории Reals_Ints;

L перед **IA**, **RA** или **IRA** для линейной арифметики;

N перед **IA**, **RA** или **IRA** для нелинейной арифметики;

UF для функциональных символов.

Задача

Для целого положительного числа $a \neq 0$ найти его наибольший делитель, неравный ему самому.

Решение задачи

Выбор логики решателя

```
(set-logic UFNIA)
```

Для целого положительного числа $a \neq 0$ найти его наибольший делитель, неравный ему самому.

Решение задачи

Объявление переменных

```
(set-logic UFNIA)

(declare-fun a () Int)
(declare-fun b () Int)
```

Для целого положительного числа a ≠ 0 найти его наибольший делитель, неравный ему самому.

Решение задачи

Начальные условия

```
(set-logic UFNIA)

(declare-fun a () Int)
(declare-fun b () Int)

(assert (> a 0))
(assert (> b 0))
(assert (> a b))
```

Для целого положительного числа $a \neq 0$ найти его наибольший делитель, неравный ему самому.

Решение задачи

Условие на делитель

```
(set-logic UFNIA)

(declare-fun a () Int)
(declare-fun b () Int)

(assert (> a 0))
(assert (> b 0))
(assert (> a b))
```

```
(define-fun isDivisor
    ((x Int) (d Int))
    Bool
    (= (mod x d) 0))

(assert (isDivisor a b))
```

Для целого положительного числа $a \neq 0$ найти его наибольший делитель, неравный ему самому.

Решение задачи

Условие на максимальность

```
(set-logic UFNIA)

(declare-fun a () Int)
(declare-fun b () Int)

(assert (> a 0))
(assert (> b 0))
(assert (> a b))
```

```
(define-fun isDivisor
    ((x Int) (d Int))
    Bool
    (= (mod x d) 0))

(assert (isDivisor a b))

(assert(forall((c Int))
  (=>(and
       (< c a)
       (isDivisor a c))
     (<= c b))))
```

Для целого положительного числа $a \neq 0$ найти его наибольший делитель, неравный ему самому.

Решение задачи

Проверка на выполнимость

```
(set-logic UFNIA)

(declare-fun a () Int)
(declare-fun b () Int)

(assert (> a 0))
(assert (> b 0))
(assert (> a b))
```

```
(define-fun isDivisor
    ((x Int) (d Int))
    Bool
    (= (mod x d) 0))

(assert (isDivisor a b))

(assert(forall((c Int))
  (=>(and
       (< c a)
       (isDivisor a c))
     (<= c b))))

(get-value b)
```

Ссылки

https://microsoft.github.io/z3guide/playground/Freeform%20Editing

- SMT решатель Z3