## Дискретная математика

# SMT

Константин Чухарев

Осень 2026

# SMT

> *Всякий хороший математик хотя бы наполовину философ, и всякий хороший философ хотя бы наполовину математик.*
>
> ***— Готлоб Фреге***

# От SAT к SMT

SAT-решатель (лекция про SAT) работает с булевыми переменными. SMT (от англ. Satisfiability Modulo Theories, выполнимость по модулю теорий) расширяет SAT: инженерные задачи оперируют целыми числами, массивами, битовыми векторами.

**Пример**

Условие `x + y <= 10 and x > 5` — не булева формула, а утверждение о целых числах.

- Можно закодировать числа битами и свести всё к чистому SAT — *bit-blasting*.
- При 64-битной арифметике одно умножение раздувается в тысячи вентилей и десятки тысяч дизъюнктов.
- Решатель тонет в битовой пыли.

SMT-солверы решают задачу архитектурно: оставляют теорию внутри, а не разворачивают её в биты на входе.

# Многосортная логика

- Чтобы говорить об арифметике и массивах, нужен язык, где у термов есть *тип*.
- Обычная логика первого порядка односортна: все термы — одного сорта.
- SMT работает с *многосортной* логикой (*many-sorted*).

**Определение 1 (Сигнатура)**

*Многосортная сигнатура* $\Sigma = (S, F)$:

- $S$ — множество *сортов* (типов): Bool, Int, Real, Array.
- $F$ — множество *функциональных символов* с *рангами*.

Ранг — кортеж сортов: $f$ принимает аргументы заданных сортов и возвращает значение заданного сорта.

# Ранги функциональных символов

**Пример (Ранги)**

- Арифметика: $S = \{\mathrm{Nat}\}$, $F = \{0, S, <, +\}$ с рангами $\mathrm{rank}(<) = (\mathrm{Nat}, \mathrm{Nat}, \mathrm{Bool})$.
- Массивы: $\mathrm{rank}(\mathrm{read}) = (\mathrm{Array}, \mathrm{I}, \mathrm{E})$, $\mathrm{rank}(\mathrm{write}) = (\mathrm{Array}, \mathrm{I}, \mathrm{E}, \mathrm{Array})$.

# Термы и интерпретации

**Определение 2 (Терм)**

*Корректно-сортированный терм* строится из переменных, констант и применений функций, уважающих ранг. $S(x < 0)$ — не терм: $<$ возвращает Bool, а $S$ ждёт Nat.

**Определение 3 (Интерпретация)**

Интерпретация $\mathcal{I}$ приписывает каждому сорту $\sigma$ непустое множество $\sigma^{\mathcal{I}}$ (например, $\mathrm{Int}^{\mathcal{I}} = \mathbb{Z}$), каждому символу $f$ — функцию $f^{\mathcal{I}}$.

Семантика связок и кванторов не меняется. Квантор $\forall x : \sigma$ пробегает по множеству $\sigma^{\mathcal{I}}$.

# Теории первого порядка

Выполнимость формулы зависит от того, *какие* интерпретации допустимы. В арифметике — только те, где Int означает $\mathbb{Z}$, а $+$ — сложение.

**Определение 4 (Теория)**

*Теория* — пара $\mathcal{T} = (\Sigma, \mathcal{M})$, где $\mathcal{M}$ — класс допустимых $\Sigma$-интерпретаций.

**Определение 5 (Выполнимость по модулю теории)**

Формула $\alpha$ *$\mathcal{T}$-выполнима*, если её выполняет некоторая интерпретация из $\mathcal{M}$. $\alpha$ *$\mathcal{T}$-общезначима*, если её выполняют все интерпретации из $\mathcal{M}$.

**Пример (Вещественная арифметика)**

$\mathcal{T}_{\text{RA}}$: сорт Real — это $\mathbb{R}$, а $+, -, \leq$ — обычные. $x + y \leq 1$ выполнима, а $\forall x, y. x + y \leq 1$ — нет.

# Аксиомы и разрешимость

Теорию часто задают *аксиомами*, а не классом моделей.

**Определение 6 (Аксиоматическая теория)**

Сигнатура $\Sigma$ + множество аксиом $\mathcal{A}$. Формула общезначима, если её выполняет всякая интерпретация, выполняющая все аксиомы.

Не всякая теория аксиоматизируема первого порядка. Арифметика натуральных чисел: любые аксиомы (Пеано) имеют нестандартные модели с бесконечными элементами $\omega, \omega + 1, \ldots$.

**Пример (Пеано и Пресбургер)**

- *Пеано* $\mathcal{T}_{\mathrm{PA}}$: символы $\{0, S, +, \times\}$ — неразрешима.
- *Пресбургер* $\mathcal{T}_{\mathrm{Pres}}$: то же *без умножения* — разрешима.
- Умножение — источник неразрешимости арифметики.

# Разрешимые фрагменты

Полная логика первого порядка неразрешима. Но кванторно-свободные фрагменты многих теорий разрешимы.

**Определение 7 (Фрагмент)**

*Фрагмент* теории — синтаксически ограниченное подмножество формул. *Кванторно-свободный* фрагмент — формулы без кванторов.

SMT нацелен на разрешимые кванторно-свободные фрагменты выбранных теорий. Когда свойство затрагивает несколько теорий — их решатели комбинируются в одну процедуру.

# Сложность фрагментов

| Теория | Что описывает | Кванторно-свободный фрагмент |
|---|---|---|
| $\mathcal{T}_{\text{EUF}}$ | равенство, неинтерпретируемые функции | $O(n \log n)$ |
| LRA | линейная арифметика над $\mathbb{R}$ | полиномиально |
| LIA | линейная арифметика над $\mathbb{Z}$ | NP-полон |
| DL | разности $x - y \leq c$ | полиномиально |
| BV | битовые векторы | NP-полон |

# Theory-солверы

**Определение 8 (Theory-солвер)**

*Theory-солвер* ($\mathcal{T}$-солвер) — процедура решения выполнимости *конъюнкций литералов* в теории $\mathcal{T}$. Вход — набор литералов вида $x + y \leq 10$, $f(a) = b$. Выход — совместен он или нет. Термины «совместный» и «выполнимый» — синонимы.

**Теорема 1 (Сведение к конъюнкциям литералов)**

Выполнимость кванторно-свободной формулы эквивалентна выполнимости некоторой конъюнкции литералов.

**Доказательство (Доказательство)**

Приводим формулу к ДНФ: выполнимость сводится к выполнимости хотя бы одной конъюнкции литералов. Обратно — конъюнкция литералов есть частный случай.

# Разностная логика

**Определение 9 (Разностная логика)**

*Разностная логика* (DL, от англ. difference logic) — фрагмент линейной арифметики: конъюнкция литералов вида $x - y \ \mathtt{op} \ c$, где $\mathtt{op} \in \{=, <, \leq, >, \geq\}$. Ограничения читаются и над целыми, и над вещественными числами: DL — фрагмент одновременно LIA и LRA.

Процедура — три шага:

1. нормализация;
2. построение графа;
3. поиск отрицательного цикла.

# Отрицательный цикл

**Пример (Отрицательный цикл)**

Формула:

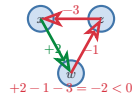
$$(x - y = 5) \land (z - y \geq 2) \land (z - x > 2) \land (w - x = 2) \land (z - w < 0)$$

.

Нормализуем к $u - v \leq c$:

| Исходный литерал | Нормализованный |
|---|---|
| $z - y \geq 2$ | $y - z \leq -2$ |
| $z - x > 2$ | $x - z \leq -3$ |
| $z - w < 0$ | $z - w \leq -1$ |

# Противоречие из цикла




Ребро $v \xrightarrow{c} u$ для каждого $u - v \le c$: от вычитаемого к уменьшаемому.

**Пример (Отрицательный вес)**
Цикл $x \xrightarrow{+2} w \xrightarrow{-1} z \xrightarrow{-3} x$ имеет вес

$$+2 - 1 - 3 = -2 < 0,$$

откуда противоречие $0 \le -2$.

# Кратчайшие расстояния

- Путь $v_0 \overset{c_1}{\to} v_1 \overset{c_2}{\to} \ldots \overset{c_k}{\to} v_k$ накапливает $v_k - v_0 \leq c_1 + \ldots + c_k$.
- Для цикла отрицательная сумма даёт $0 \leq W < 0$.
- Без отрицательных циклов кратчайшие расстояния дают выполняющее присваивание.

# Равенство и EUF

**Определение 10 (Теория EUF)**

*Равенство с неинтерпретируемыми функциями* $\mathcal{T}_{\mathrm{EUF}}$: равенство $=$ и произвольные функции $f, g, h, \ldots$. Аксиомы:

- рефлексивность;
- симметричность;
- транзитивность;
- конгруэнтность: $x_i = y_i$ для всех $i$ влечёт $f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$.

Кванторно-свободная выполнимость разрешима за полиномиальное время алгоритмом *congruence closure*:

1. разбить термы на классы эквивалентности;
2. сливать классы, когда равные аргументы дают равные значения.

# Congruence closure

**Пример (Congruence closure)**

$$(a = b) \wedge (f(a) = b) \wedge \neg(g(a) = g(f(a)))$$

.

Из $a = b$ и $f(a) = b$ получаем класс $\{a, b, f(a)\}$. По конгруэнтности $a = f(a)$ влечёт $g(a) = g(f(a))$ — противоречие. Формула невыполнима.

# Линейная арифметика

**Определение 11 (LRA и LIA)**

- LRA — линейная арифметика над вещественными: задача совместности полиномиальна (эллипсоиды, методы внутренней точки), хотя симплекс в худшем случае экспоненциален.
- LIA — над целыми: симплекс даёт вещественное решение, для целочисленности нужны дополнительные шаги (ветвление и границы, отсечения); NP-полна.

На практике симплекс быстр, и солверы (в том числе Z3) используют симплекс-подобные процедуры.

# Битовые векторы

**Определение 12 (Битовые векторы)**

BV — арифметика фиксированной разрядности, в точности как в процессоре. Решается *bit-blasting*: каждый бит — булева переменная, арифметика разворачивается в схему, которую решает SAT.

Для битовых векторов bit-blasting неизбежен — сама семантика битовая. Для линейной арифметики его избегают: симплекс работает прямо с числами.

# Теория массивов

**Определение 13 (Теория массивов)**

- Сорта: Array, I (индексы), E (элементы).
- Функции: read и write.
- $\mathrm{read}(a, i)$ — значение по индексу.
- $\mathrm{write}(a, i, v)$ — массив с заменой по индексу $i$.

Аксиомы — read-over-write и экстенсиональность:

- $\mathrm{read}(\mathrm{write}(a, i, v), i) = v$ — запись сразу читается;
- $i \neq j \to \mathrm{read}(\mathrm{write}(a, i, v), j) = \mathrm{read}(a, j)$ — запись не трогает другие индексы;
- если $\forall i.\ \mathrm{read}(a, i) = \mathrm{read}(b, i)$, то $a = b$ — экстенсиональность.

Полная теория неразрешима, кванторно-свободный фрагмент NP-полон и решается подстановкой в аксиомы.

# DPLL(T)

У каждой теории есть своя процедура решения. Осталось показать, как SAT-решатель руководит ими всеми.

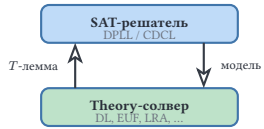
**Определение 14 (Архитектура DPLL(T))**

SAT-решатель на *булевом скелете* + theory-солвер как оракул совместности.

- Каждый атом теории (например, `x + y <= 10`) заменяется свежей булевой переменной.
- SAT-решатель ищет модель булева скелета.
- Theory-солвер проверяет совместность атомов, помеченных истинными.
- При конфликте возвращает конфликтное подмножество — SAT-решатель добавляет запрещающий дизъюнкт.

Тот же цикл «поиск, конфликт, обучение», что в CDCL, но с теорией в роли оракула. Всё, что работает в CDCL — watched literals, VSIDS, выученные дизъюнкты — работает и внутри SMT-солвера.

# Архитектура DPLL(T)



# Комбинация теорий (Nelson-Oppen)

Формула `f(x) = g(y) and x < y` смешивает равенство с неинтерпретируемыми функциями и арифметику: ни один theory-солвер по отдельности её не решит. Решатели теорий соединяются методом Nelson-Oppen.

Проследим метод на мотивирующей формуле $f(x) = g(y) \wedge x < y$. *Пурификация*: термы $f(x)$ и $g(y)$ выносятся в свежие переменные $u$ и $v$, арифметика получает $x < y$, а EUF — равенства $u = f(x)$, $v = g(y)$, $u = v$. *Обмен* здесь тривиален: солверы не выводят новых равенств об общих переменных $x$, $y$, и неподвижная точка достигается сразу. *Склейка моделей*: общим переменным $x$, $y$ назначаются общие значения, и равенство $u = v$ согласуется с $f$ и $g$.

# Межтеоретическое противоречие

**Пример (Межтеоретическое противоречие)**

Рассмотрим $x \leq y \wedge y \leq x \wedge f(x) = f(y) \wedge \neg(x = y)$. Линейная арифметика выводит $x = y$ из $x \leq y$ и $y \leq x$. Теория EUF хранит $f(x) = f(y)$ и $\neg(x = y)$ — сама по себе она выполнима. Когда арифметика передаёт выведенное $x = y$, EUF видит противоречие с $\neg(x = y)$. Ни один солвер по отдельности не увидит невыполнимость: арифметика не видит $f$, а EUF — не видит $\leq$.

# Условия Nelson-Oppen

**Определение 15 (Nelson-Oppen)**

*Пурификация*: общие подтермы выносятся в свежие переменные, и каждая теория решает свою часть. Солверы обмениваются выведенными равенствами и неравенствами об общих переменных до неподвижной точки. Противоречие в любой части означает невыполнимость, иначе модели частей склеиваются в одну.

Нужны два условия:

- *непересекающиеся сигнатуры*: теории делят только равенство $=$;
- *стабильная бесконечность*: каждая выполнимая кванторно-свободная формула имеет бесконечную модель.

Стабильная бесконечность нужна для склейки моделей. Каждая теория строит модель на своём носителе, а склейка требует биекции носителей на общих переменных. Если у теории все модели конечны, такую биекцию провести нельзя, и общий носитель не построить.

# SMT-LIB

Общепринятый входной язык SMT-солверов — SMT-LIB: Lisp-подобные S-выражения.

**Пример (SMT-LIB)**

Проверка выполнимости $x + y \leq 10 \land x > 5$:

```
(declare-const x Int)
(declare-const y Int)
(assert (<= (+ x y) 10))
(assert (> x 5))
(check-sat)
(get-model)
```

Солвер отвечает `sat` и выдаёт модель, например $x = 6$, $y = 0$.

# Z3 и приложения

- Z3 (Microsoft Research, Леонардо де Моура и Никола Бьёрнер, 2008) — главный SMT-солвер индустрии.
- Другие — CVC5, Yices, Alt-Ergo.

SMT — рабочая лошадка автоматического рассуждения:

- *символьное исполнение* — путь программы кодируется формулой, которую проверяет SMT;
- *верификация* — условия корректности доказываются SMT-солвером;
- *синтез* — поиск программы по спецификации сводится к SMT-запросам;
- *model checking* — свойства системы проверяются SMT-запросами к символьной модели.

# Что мы поняли?

- SAT решает булеву выполнимость; SMT — выполнимость по модулю теории: числа, массивы, битовые векторы.
- Многосортная логика даёт термам типы; теория сужает класс допустимых интерпретаций.
- Полная арифметика неразрешима, но кванторно-свободные фрагменты разрешимы — SMT нацелен именно на них.
- Theory-солвер решает конъюнкции литералов: разностную логику — отрицательным циклом, EUF — congruence closure, арифметику — симплексом.
- DPLL(T) встраивает theory-солвер в цикл CDCL.
- Z3 делает SMT рабочим инструментом верификации, символьного исполнения и синтеза.

# Вопросы для самопроверки

- Чем SMT отличается от чистого SAT? Почему bit-blasting всей арифметики — плохая идея?
- Что такое ранг функционального символа? Приведите ранг read.
- Чем теория отличается от чистой логики первого порядка?
- Почему арифметика Пеано неразрешима, а Пресбургера — разрешима?
- Как разностную логику сводят к поиску отрицательного цикла?
- Что делает theory-солвер в DPLL(T), и что возвращает SAT-решателю при конфликте?
- Почему для битовых векторов применяют bit-blasting, а для линейной арифметики — нет?