

Санкт-Петербургский государственный университет  
Математико-механический факультет

Кафедра системного программирования

Программная инженерия

Суханова Анжела Кирилловна

# Реализация элиминации кванторов для арифметики Пресбургера, обогащённой функцией возведения двойки в степень

Производственная практика

Научный руководитель:  
ассистент кафедры ИАС Смирнов К. К.

Санкт-Петербург  
2021

# Оглавление

<b>Введение</b>	<b>3</b>
<b>1. Постановка задачи</b>	<b>5</b>
<b>2. Теоретический обзор</b>	<b>6</b>
2.1. Элиминация кванторов для арифметики битовых векторов	6
2.2. Обогащённая арифметика Пресбургера и теория битовых векторов . . . . .	6
2.3. Алгоритм элиминации кванторов для расширенной арифметики Пресбургера . . . . .	8
2.3.1. Случай линейного вхождения связанной переменной	9
2.3.2. Случай экспоненциального вхождения связанной переменной . . . . .	10
2.4. Проблема рассматриваемого алгоритма . . . . .	11
<b>3. Описание реализации</b>	<b>14</b>
3.1. SMT-решатель . . . . .	14
3.2. Формат ввода формул . . . . .	14
3.3. Архитектура реализации . . . . .	15
3.4. Проверка корректности результата . . . . .	15
<b>4. Экспериментальные исследования</b>	<b>17</b>
<b>5. Текущие результаты</b>	<b>19</b>
<b>Список литературы</b>	<b>20</b>

# Введение

Современный этап развития индустрии программных систем характеризуется значительным усложнением процесса их разработки, что приводит к увеличению числа ошибок, возникающих при проектировании программного обеспечения. В таких условиях крайне важными оказываются проверка и доказательство корректности разрабатываемой программы, ведь они необходимы для контроля соответствия поведения программы ожидаемому и обеспечения её безопасности. Существуют разные методы, направленные на разработку качественного программного обеспечения, отвечающего поставленным требованиям: одним из них является формальная верификация.

Формальная верификация — это доказательство соответствия или несоответствия программы её формальному описанию или, что более распространено на практике, проверка формализованных условий, описывающих ожидаемое или недопустимое поведение программы (часто у разработчиков нет формального описания программы, но есть представление о «запрещённых» состояниях, в которые она не должна попадать). В основу этой проверки нередко ложится решение задачи выполнимости формул в теориях (SMT<sup>1</sup>), так как условия программы и ограничения, накладываемые на неё требованиями, могут быть сведены к определению выполнимости логических формул. Существуют SMT-решатели (SMT-солверы), автоматически определяющие выполнимость формулы в теориях. SMT-решатели используются инструментами формальной верификации, такими как [Frama-C](#), [BLAST](#), [Java Pathfinder](#) и другие.

Распространённым подходом к решению задачи выполнимости формул в теориях является bit-blasting. Он заключается в сведении SMT-задачи к соответствующей задаче выполнимости булевых формул, что достигается введением пропозициональных переменных для всех битов исходных термов. К сожалению, этот способ очень трудоёмкий. В ка-

---

<sup>1</sup>SMT — задача выполнимости формул в логике первого порядка, где функциональные и предикатные символы интерпретируются согласно конкретным теориям. В дальнейшем будет использоваться эта аббревиатура.

честве альтернативного подхода к решению SMT-задачи (или способа сократить bit-blasting) может быть рассмотрено упрощение исходных формул.

Обычно легче определить, выполнима ли формула без кванторов, поэтому полезно уметь преобразовывать формулы, содержащие кванторы, в семантически эквивалентные бескванторные. Процесс такого преобразования называется элиминацией кванторов (quantifier elimination, QE)<sup>2</sup>.

Так как в основе архитектуры компьютера лежат операции с битовыми векторами, то необходимо уметь решать SMT в теории битовых векторов, что вызывает интерес к упрощающим это решение алгоритмам исключения кванторов из формул над двоичными векторами. Операции над последними можно свести к вычислениям в арифметике Пресбургера<sup>3</sup>, обогащённой функцией  $2^x$ . Доказательство того, что эта теория допускает элиминацию кванторов, а также алгоритм элиминации, сопутствующий построению доказательства, впервые были представлены А. Л. Семёновым [7], а затем более подробно описаны в других работах [4, 6]. Однако идеи этого алгоритма не использовались для реализации элиминации кванторов в теории битовых векторов, чему и посвящена эта работа.

---

<sup>2</sup>Теория  $T$  допускает элиминацию кванторов, если для любой формулы этой теории  $\phi$  существует формула  $\psi$  без кванторов, такая что  $T \models \forall y. \phi(y) \leftrightarrow \psi(y)$

<sup>3</sup>Арифметика Пресбургера — это теория первого порядка, описывающая натуральные числа со сложением и названная в честь предложившего её Мойжеша Пресбургера.

# 1. Постановка задачи

Целью данной работы является реализация элиминации кванторов для арифметики битовых векторов на основе элиминации кванторов в арифметике Пресбургера, расширенной функцией возведения двойки в степень<sup>4</sup>. Для её достижения были поставлены следующие задачи.

- Выбор SMT-решателя.
- Изучение алгоритма элиминации кванторов для расширенной арифметики Пресбургера и реализация его в рамках арифметики битовых векторов и выбранного SMT-решателя.
- Экспериментальное исследование элиминации: сравнение времени работы и длины результирующей формулы реализации и поддерживающего элиминацию кванторов SMT-решателя.

---

<sup>4</sup>Далее она иногда будет упоминаться как «расширенная арифметика Пресбургера», но имеется в виду, что теория расширяется функцией  $2^x$ .

## 2. Теоретический обзор

### 2.1. Элиминация кванторов для арифметики битовых векторов

В настоящее время большинство SMT-решателей работают только с бескванторными формулами. Арифметику битовых векторов с кванторами поддерживают SMT-решатели Boolector, CVC4, Yices<sup>5</sup>, Z3 и Q3B [8], однако процесс элиминации ни одного из них не основывается на идеях элиминации кванторов для арифметики натуральных чисел, расширенной  $2^x$ .

### 2.2. Обогащённая арифметика Пресбургера и теории битовых векторов

Говоря о расширенной арифметике Пресбургера, мы будем иметь в виду арифметику со следующей сигнатурой:  $\langle 0, 1, +, 2^x, \leq \rangle$ . Запись вида  $1 + 1 + \dots + 1$  обозначим соответствующим натуральным числом. Воспользуемся некоторыми обозначениями, введёнными на [6, р. 2-3, 5].

- $x < y \leftrightarrow x + 1 \leq y$  и  $x \geq (>)y \leftrightarrow y \leq (<)x$ .
- $x - y = 0$ , если  $x < y$ , и  $x - y = x - y$ , если  $y \leq x$ .
- $n \cdot x$ , где  $n \in \mathbb{N}^*$  — обозначение  $x + x + \dots + x$  ( $n$  раз).
- $t_1 + z \cdot x \leq (=, \geq) t_2$ , где  $z \in \mathbb{Z}$ , означает, что  $t_1 \leq (=, \geq) t_2 + (-z) \cdot x$  при  $z < 0$ .
- $\frac{x}{n} = y \leftrightarrow x = n \cdot y + z$ , где  $0 \leq z < n$ .
- $l_2(x) = y \leftrightarrow 2^y \leq x < 2^{y+1}$ .
- $x \equiv_d m$ , где  $0 \leq m \leq d - 1$ , если  $x - \frac{x}{d} \cdot d = m$ .

---

<sup>5</sup>В отличие от остальных упомянутых решателей Yices работает не с произвольными формулами, а только с формулами вида  $\exists x. \forall y. Q(x, y)$  [8].

Таблица 1 — Сравнение арифметики битовых векторов размер  $n$  и обогащённой арифметики Пресбургера

Обогащенная арифметика Пресбургера	Теория битовых векторов (синтаксис SMT-LIB)
Носитель: $\mathbb{N}$	Носитель: битовые векторы фикс. размеров ( <code>_ BitVec n</code> )
$t_1 \leq t_2$	<code>bvslte t<sub>1</sub> t<sub>2</sub></code> ( <code>bvulte t<sub>1</sub> t<sub>2</sub></code> )
$t_1 + t_2$	<code>bvadd t<sub>1</sub> t<sub>2</sub></code>
$2^t$	<code>bvshl 1 t</code>
	<code>bvand</code> , <code>bvor</code> , <code>bvnot</code> , <code>bvslt/bvult</code> , <code>bvsgt(e)/bvugt(e)</code> и другие

Несмотря на интуитивность сведения формул в арифметике битовых векторов к формулам в арифметике Пресбургера, расширенной двоичной экспонентой, эти теории имеют существенные различия (таблица 1).

Во-первых, множество натуральных чисел не ограничено сверху, а носитель теории битовых векторов — это множество двоичных векторов размера  $n$ . Сведение формул в арифметике битовых векторов размера  $n$  к формулам в расширенной арифметике Пресбургера может быть осуществлено по следующим правилам ( $t_i$  — терм,  $x$  — переменная в теории битовых векторов):

$$\begin{aligned}
 Tr(\varphi \wedge \psi) &= Tr(\varphi) \wedge Tr(\psi) \\
 Tr(\varphi \vee \psi) &= Tr(\varphi) \vee Tr(\psi) \\
 Tr(\varphi \rightarrow \psi) &= Tr(\varphi) \rightarrow Tr(\psi) \\
 Tr(\neg \varphi) &= \neg Tr(\varphi)
 \end{aligned}$$

$$Tr(t_1 \text{ op } t_2) = (Tr(t_1) \text{ op } Tr(t_2)) \bmod 2^n \quad (1)$$

$$Tr(x) = x$$

Заметим, что по рассматриваемому алгоритму можно проэлиминировать некоторое подмножество формул в арифметике битовых век-

торов без сведения к арифметике натуральных чисел, но оно весьма ограничено.

Во-вторых, битовые векторы могут быть знаковыми и беззнаковыми, из-за чего у некоторых операций с ними существует две версии. Однако числа, соответствующие знаковым битовым векторам, легко могут быть выражены через числа, соответствующие беззнаковым, по следующему неформальному правилу:

$$x_s = ite(x_u < 2^{n-1}, x_u, x_u - 2^n)$$

Также в арифметике двоичных векторов есть множество функции и предикатов, не имеющих аналогов в обогащённой арифметике Пресбургера. Эта проблема может быть решена рассмотрением таких функций и предикатов в отдельности и поиском для каждого из них эквивалентной последовательности операций в арифметике Пресбургера, расширенной  $2^x$ .

### 2.3. Алгоритм элиминации кванторов для расширенной арифметики Пресбургера

Заметим, что формулу, содержащую квантор всеобщности,  $\forall x.F$  можно заменить эквивалентной ей формулой  $\neg \exists x. \neg F$ , а все кванторы существования можно элиминировать последовательно от самого внутреннего к внешнему. Также любая бескванторная формула первого порядка может быть представлена в конъюнктивной нормальной форме. Таким образом, алгоритм элиминации сводится к устранению квантора существования из формулы вида  $\exists x. \theta(x, \bar{y})$ , где  $\bar{y}$  — свободные переменные, а сама функция  $\theta(x, \bar{y})$  — конъюнкт атомарных формул. Это устранение квантора будет построено по материалу, изложенному в [6]. Термины и обозначения согласованы с этой работой.

Первым шагом в элиминации кванторов для рассматриваемой арифметики является преобразование формулы под квантором в дизъюнкт конъюнктов неравенств между термами следующего вида:



- $\sum_{i=0}^n a_i \cdot 2^{d \cdot x_i} + \sum_{i=0}^n b_i \cdot x_i + c$ , где  $a_i, b_i, c \in \mathbb{Z}$ ,  $d \in \mathbb{N}$ , а  $x_i$  — связанные переменные, причём  $x_0$  — обозначение  $x$  (будем называть их  $S$ -термами);
- термы сигнатуры арифметики Пресбургера, обогащённой функцией возведения двойки в степень, без связанных переменных (будем называть их  $L$ -термами).

Это преобразование осуществляется посредством введения новых переменных, связанных кванторами существования — отсюда и возникают  $x_i$  (подробнее об этом шаге можно прочитать на [6, р. 5]).

Существует два варианта вида преобразованной формулы.

- $x$  появляется во всех неравенствах линейно. Тогда можно считать, что формула под кванторами выглядит следующим образом:

$$\bigwedge_{1 \leq j \leq q, 1 \leq k \leq s, 1 \leq i \leq p} f_j(\bar{x}) + g_j(\bar{y}) \leq d_k \cdot x \leq f_i(\bar{x}) + g_i(\bar{y}),$$

где  $d_k \in \mathbb{Z}$  и зависит от  $i$  и  $j$ ,  $\bar{y}$  — свободные переменные, а  $\bar{x} = (x_1, x_2, \dots, x_n)$ , то есть  $g_j(\bar{y})$ ,  $g_i(\bar{y})$  —  $L$ -термы, а  $f_j(\bar{x})$ ,  $f_i(\bar{x})$  —  $S$ -термы.

- Хотя бы в одно неравенство  $x$  входит в экспоненциальном терме, то есть существует неравенство, имеющее вид:

$$a_0 \cdot 2^{d \cdot x_0} + \sum_{i=1}^n a_i \cdot 2^{d \cdot x_i} + \sum_{j=0}^n b_j \cdot x_j + c \leq t(\bar{y}), \quad (2)$$

где  $t(\bar{y})$  —  $L$ -терм,  $a_i, b_j, c \in \mathbb{Z}$ ,  $a_0 \neq 0$ ,  $d \in \mathbb{N}^*$ .

Таким образом, реализация подразумевает разбор двух случаев: линейного и экспоненциального вхождения  $x$ .

### 2.3.1. Случай линейного вхождения связанной переменной

Работа над этим случаем была начата с упрощённого варианта, а именно с элиминации кванторов для формул с единственной связанной пе-

ременной:

$$\exists x. \bigwedge_{1 \leq j \leq q, 1 \leq k \leq s, 1 \leq i \leq p} g_j(\bar{y}) \leq d_k \cdot x \leq g_i(\bar{y}), \text{ где } q, p \in \mathbb{N}.$$

Если для всех  $k$   $d_k = 1$ , то эквивалентная ей формула без квантора:

$$\bigwedge_{1 \leq j \leq q, 1 \leq i \leq p} g_j(\bar{y}) \leq g_i(\bar{y}).$$

Если  $\exists k \ d_k > 1$ , то неточность предложенного алгоритма (её описание можно найти в 2.4) не позволяет нам найти эквивалентную формулу без квантора.

### 2.3.2. Случай экспоненциального вхождения связанной переменной

Опишем то подмножество этого случая, которое не требует применения операции сведения (1). С этого подмножества была начата работа над экспоненциальным случаем.

Введём обозначение для формулы под квантором из прошлого случая:

$$\psi(x, \bar{y}) = \bigwedge_{1 \leq j \leq q, 1 \leq k \leq s, 1 \leq i \leq p} g_j(\bar{y}) \leq d_k \cdot x \leq g_i(\bar{y}).$$

Рассмотрим следующее подмножество формул, в которых элиминируемая переменная встречается в экспоненциальном терме

$$\exists x. \psi(x, \bar{y}) \wedge \bigwedge_i (2^x \leq g_i(\bar{y}) \vee g_i(\bar{y}) \leq 2^x).$$

Обозначим формулу  $2^x \leq g_i(\bar{y}) \vee g_i(\bar{y}) \leq 2^x$  под квантором  $\tau_i(x, \bar{y})$ .

Заметим, что формулы  $2^x \leq g_i(\bar{y})$  и  $g_i(\bar{y}) \leq 2^x$  — частные случаи формулы (2). Для обеих верно  $n = 0$ ,  $d = 1$ ,  $b_0 = 0$ ,  $c = 0$ , но в первой  $a_0 = 1$  и  $l_2(a_0) = 0$ , а во второй  $a_0 = -1$ . Для них по [6, р. 7] верно следующее:

- $J = \{0 \dots n\} = \{0\}$ ,  $J_1 = \{j \in J : b_j \geq 0\} = \{0\}$ ;

- $b_+ < 0$ , так как

$$b_+ = \begin{cases} 2 \cdot (l_2(\sum_{j \in J_1} b_j) + 3), & J_1 \neq \emptyset \\ 0, & J_1 = \emptyset \end{cases}$$

- $b_- = 0$ , так как

$$b_- = \begin{cases} 2 \cdot (l_2(\sum_{j \in J \setminus J_1} -b_j) + 4), & J \setminus J_1 \neq \emptyset \\ 0, & J \setminus J_1 = \emptyset \end{cases}$$

- $c_+ = 0$ , так как

$$c_+ = \begin{cases} l_2(c) + 3, & c > 0 \\ 0, & c \leq 0 \end{cases}$$

- $c_- < 0$ , так как

$$c_- = \begin{cases} 0, & c > 0 \\ l_2(-c) + 4, & c \leq 0 \end{cases}$$

Таким образом,  $N = \max\{b_+, b_-, c_+, c_-\} = 0$ . Искомая формула без квантора в данном случае:

$$\bigwedge_i \bigvee_{0 \leq k \leq M} \tau_i(k, \bar{y}),$$

где  $M = \max\{a_0 > 0, \max\{N, l_2(g_i(\bar{y})), l_2(g_i(\bar{y})) + 1\}, N\}$ .

## 2.4. Проблема рассматриваемого алгоритма

В процессе изучения алгоритма элиминации кванторов для расширенной арифметики Пресбургера была обнаружена неточность, ставящая под сомнение завершаемость описанных в [6] действий.

По предложенному методу входная формула приводится к определённому виду, для чего вводится конечное число новых переменных, связанных кванторами существования. Утверждается, что после этого переменные, требующие элиминации, вводятся не будут, и так как на

каждой итерации алгоритма происходит устранение одной из них, то мы получим искомую формулу за конечное число шагов.

Термы вида  $\tau(x'_i, \bar{y}) \equiv_d m$  (где  $x'_i$  — переменная, связанная квантором существования), в некоторых случаях возникающие в формуле по завершении итерации, приводятся к  $\tau(x'_i, \bar{y}) - \frac{\tau(x'_i, \bar{y})}{d} \cdot d = m$ . Частное  $\frac{\tau(x'_i, \bar{y})}{d}$  должно быть заменено на переменную, связанную  $\exists$ . Таким образом, некоторые формулы требуют повторного введения переменных под кванторами.

Продемонстрируем проблему на примере.

Рассмотрим формулу  $\exists x_0. (\frac{2 \cdot x_0}{3} \leq 2 \cdot x_0 \leq 10)$  и воспроизведём для неё шаги по элиминации квантора.

- *Первый шаг* [6, р. 5]. Обозначив  $\frac{2 \cdot x_0}{3}$  новой связанной переменной  $x_1$  по пункту (3), получаем:

$$\exists x_0. \exists x_1. \bigvee_{0 \leq m < 3} (x_1 \leq 2 \cdot x_0 \leq 10 \wedge 3 \cdot x_1 + m = 2 \cdot x_0)$$

$\leftrightarrow$

$$\exists x_0. \exists x_1. \bigvee_{0 \leq m < 3} (x_1 \leq 2 \cdot x_0 \leq 10 \wedge 3 \cdot x_1 + m \leq 2 \cdot x_0 \leq 3 \cdot x_1 + m)$$

Вид формулы позволяет перейти ко второму шагу. Введём обозначение:

$$\theta_0(x_0, x_1) = \bigvee_{0 \leq m < 3} (x_1 \leq 2 \cdot x_0 \leq 10 \wedge 3 \cdot x_1 + m \leq 2 \cdot x_0 \leq 3 \cdot x_1 + m)$$

- *Второй шаг* [6, р. 5]. Рассмотрим все перестановки  $\{x_0, x_1\}$ : для каждой перестановки  $\sigma$  определена  $\theta_{0,\sigma}(x_0, x_1)$ .
- *Третий шаг* [6, р. 6]. Рассмотрим случай  $\sigma = id$ ,  $m = 0$ :

$$x_1 \leq 2 \cdot x_0 \leq 10 \wedge 3 \cdot x_1 \leq 2 \cdot x_0 \leq 3 \cdot x_1 \wedge x_1 \leq x_0. \quad (3)$$

Переменная  $x_0$  встречается в каждом неравенстве линейно, то есть нас интересует случай А).

НОК  $d = 2$ . Поскольку  $d = 2^r \cdot d_0$ , то  $r = 1$ ,  $d_0 = 1$ ,  $\phi(d_0) = 1$ ,  $0 \leq k < 2$ .

Домножим обе части неравенства  $x_1 \leq x_0$  на  $d$  и произведём замену  $x_1 \rightarrow 1 + k + x'_1$ . Теперь формула 3 выглядит следующим образом:

$$\bigvee_{0 \leq k < 2} (1 + k + x'_1 \leq 2 \cdot x_0 \leq 10 \wedge 3 \cdot (1 + k + x'_1) \leq 2 \cdot x_0 \leq 3 \cdot (1 + k + x'_1) \wedge 2 \cdot (1 + k + x'_1) \leq 2 \cdot x_0)$$

Так как  $1 + k + x'_1 < 2 \cdot (1 + k + x'_1) < 3 \cdot (1 + k + x'_1)$ , то эквивалентная формула без  $x_0$  это

$$\bigvee_{0 \leq k < 2} (3 \cdot (1 + k + x'_1) \leq 10 \wedge 2 \leq 10 - 3 \cdot (1 + k + x'_1) \wedge (\bigvee_{0 \leq c < 2} 10 - 3 \cdot (1 + k + x'_1) = c \wedge (\bigvee_{0 \leq c' \leq c} 3 \cdot (1 + k + x'_1) \equiv_2 2 - c')))$$

Перейдём к элиминации  $x'_1$ . Терм  $3 \cdot (1 + k + x'_1) \equiv_2 2 - c'$  эквивалентен

$$3 \cdot (1 + k + x'_1) - \frac{3 \cdot (1 + k + x'_1)}{2} \cdot 2 = 2 - c'$$

По пункту (3) [6, р. 5] производится замена  $\frac{3 \cdot (1 + k + x'_1)}{2}$  на новую переменную  $x_2$ , связанную квантором существования:

$$3 \cdot (1 + k + x'_1) = (2 - c') \cdot 2 + x_2.$$

Таким образом, мы вынуждены ввести новую переменную.

В настоящее время по описанной проблеме ведётся переписка с Франсуазой Поинт, автором статьи [6].

Из-за обнаруженной неточности не совсем понятно, как должно осуществляться сведение формул в теории битовых векторов к формулам в расширенной арифметике Пресбургера, так как по правилу (1) возникают описанные термы со сравнением по модулю. Пока сведение одной арифметики к другой не получило реализации.

## 3. Описание реализации

### 3.1. SMT-решатель

В настоящее время существует несколько поддерживаемых, конкурентоспособных SMT-решателей, работающих с двоичными векторами: [Boolector](#), [Z3](#), [CVC4](#) и другие. В рамках этой курсовой работы будет использоваться Boolector, так как он специализируется на теории битовых векторов, а также в течение многих лет побеждал в The SMT Competition<sup>6</sup> (за исключением 2020-го года, в котором актуальная версия Boolector-а не принимала участие в соревновании).

### 3.2. Формат ввода формул

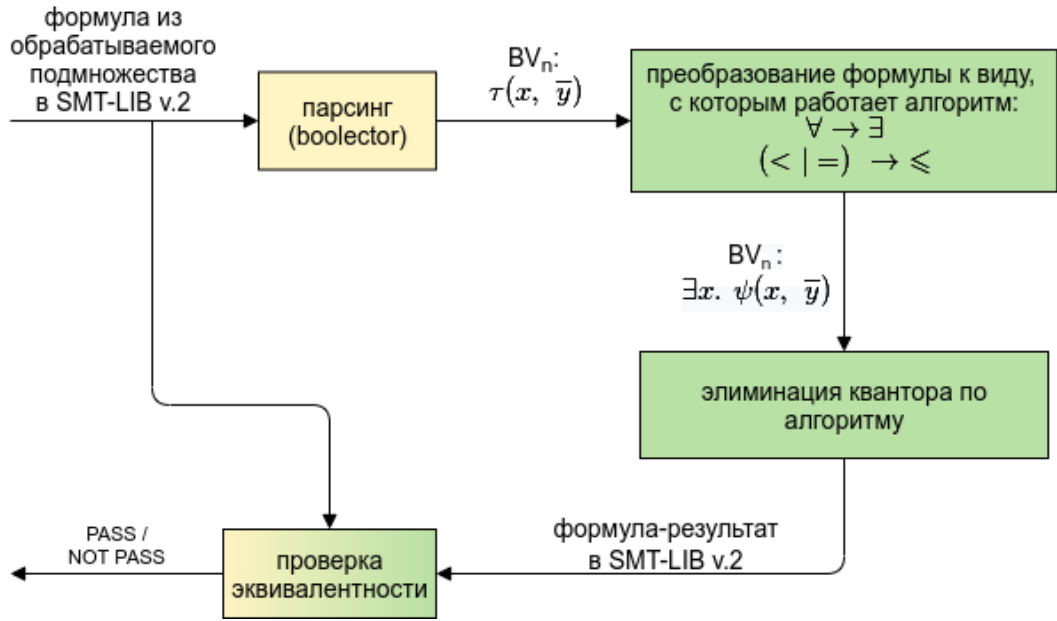
Изначально в качестве формата ввода рассматривались стандарты Btor [2] и Btor2 [3]. Оба формата не поддерживают работу с кванторами, и хотя это неудобство может быть преодолено (например, договорённостью, что первые введённые переменные связаны кванторами существования), было принято решение работать со стандартом SMT-LIB v.2 [1], в котором можно записывать формулы с кванторами. Парсеры всех вышеупомянутых стандартов записи встроены в Boolector.

Работая с Boolector-ом, можно задать любую формулу над битовыми векторами (даже с кванторами) программно, используя функции решателя, но такой ввод не очень удобен. К тому же важно, чтобы программа могла работать с общепринятым, распространённым форматом, ведь в нём записывались и пишутся условия и ограничения, накладываемые на реальное программное обеспечение. Идея написать собственный парсер выражений над битовыми векторами также была отброшена из-за неоправданной трудоёмкости.

---

<sup>6</sup>The SMT Competition или SMT-COMP — ежегодное соревнование между SMT-солверами: <https://smt-comp.github.io/2020/index.html>

Рисунок 1 — Архитектура решения



### 3.3. Архитектура реализации

Для достижения поставленной цели была разработана следующая архитектура (рис. 1).

В настоящий момент алгоритм элиминации кванторов реализован для следующих формул теории битовых беззнаковых векторов (вместо  $\leq$  может быть  $<$  или  $=$ , а вместо  $\exists - \forall$ ):

1.  $\exists x. \bigwedge_{i,j} (g_i(\bar{y}) \leq x \wedge x \leq g_j(\bar{y}))$ , где  $g_i(\bar{y}), g_j(\bar{y})$  — термы в арифметике битовых векторов, представляющие из себя линейные комбинации констант, свободных переменных  $(\bar{y})$  и сдвигов  $1 \ll y_k$ ;
2.  $\exists x. \bigwedge_i ((1 \ll x) \leq g_i(\bar{y}) \vee g_i(\bar{y}) \leq (1 \ll x))$ .

### 3.4. Проверка корректности результата

Для тестирования программы была реализована проверка эквивалентности исходной и результирующей формул. Обратим внимание на следующий факт: пусть  $\varphi$  — исходная формула, а  $\theta$  — результат, тогда  $\varphi \oplus \theta$  должна быть невыполнима. Таким образом, проверка формул на

эквивалентность заключается в решении SMT-задачи для строгой дизъюнкции исходной формулы и результата: если эта формула выполнима, то результат некорректен, иначе — полученная формула эквивалентна исходной.



## 4. Экспериментальные исследования

Замеры проводились на ноутбуке с Ubuntu 20.10, Intel Core i5-7300HQ CPU, 2.50GHz, DDR4 8Gb RAM. В таблице 2 приведено сравнение элиминации квантора по времени и длине итоговой формулы для SMT-решателя Z3 и созданной реализации. Заметим, что замерялось только время элиминации без учёта парсинга входной формулы и вывода результата. В таблице 2 представлены средние значения и стандартные отклонения ( $\sigma$ ) по 20 запускам (в миллисекундах), а также длина найденной формулы (в символах<sup>7</sup>).

Таблица составлена по результатам запуска программ на следующих тестах ( $n$  — размер битовых векторов):

1.  $\exists x. x \geq 9505$  ( $n = 16$ ) — тест */conjunction\_level\_benchmarks/DeltaTR\_RFRNC\_OUT\_QESMT\_benchmark\_conjunction\_38.smt*<sup>8</sup> из набора тестов [Benchmarks](#), на которых построено экспериментальное исследование [5].
2.  $\exists x. y \leq x \wedge 2 \leq x \wedge z \leq x$  ( $n = 4$ )
3.  $\forall x. 3 \cdot y \leq x \wedge x \leq 12 \cdot y$  ( $n = 4$ )
4.  $\exists x. x \leq 997 \cdot y \wedge z \leq x \wedge x \leq t$  ( $n = 10$ )
5.  $\exists x. x \leq 2 \cdot y + z \wedge 10 \cdot y \leq x$  ( $n = 6$ )
6.  $\exists x. x \leq 5 \cdot y + 7 \wedge 8 \cdot (y + z) \leq x$  ( $n = 8$ )
7.  $\exists x. y + 15 < x \wedge x < 1$  ( $n = 4$ )
8.  $\exists x. 3 \cdot (1 \ll y) \leq x \wedge x \leq 7 \cdot (1 \ll y)$  ( $n = 4$ )
9.  $\forall x. (1 \ll y) \leq x \wedge 2 \leq x \wedge z \leq x$  ( $n = 4$ )
10.  $\exists x. 3 \cdot (1 \ll y) \leq x \wedge x \leq 12 \cdot y$  ( $n = 6$ )

---

<sup>7</sup>Служебные слова и символы не учитываются (в Z3 «(goals (goal» и закрывающая скобка), а длину тождественно верных/ложных формул будем считать равной 1 для единообразия (на самом деле они могут выводиться как «true», «false», «»).

<sup>8</sup>Формула была переведена из формата SMT-LIB в SMT-LIB v.2.

11.  $\exists x. x \leq 3 \cdot (1 \ll y) \wedge (1 \ll z) \leq x \wedge x \leq t \ (n = 6)$
12.  $\forall x. x \leq 2 \cdot (1 \ll y) + (1 \ll z) \wedge 10 \cdot (1 \ll y) \leq x \ (n = 8)$
13.  $\exists x. x \leq 5 \cdot (1 \ll y) + 7 \wedge 8 \cdot ((1 \ll y) + z) \leq x \ (n = 8)$
14.  $\exists x. (1 \ll x) \leq (1 \ll y) + 11 \cdot y + 4 \ (n = 4)$
15.  $\exists x. (1 \ll x) \leq y + 3 \cdot z + 8 \ (n = 6)$
16.  $\exists x. (1 \ll x) \leq 7 \cdot y \wedge (1 \ll x) \leq z \wedge (1 \ll x) \leq (1 \ll t) \ (n = 8)$

Таблица 2 — Сравнение времени элиминации (мс) и длины формулы-результата (количество символов).

Тест	Boolector (PA + $2^x$ )			Z3		
	Среднее время	$\sigma$	Длина формулы	Среднее время	$\sigma$	Длина формулы
1	0.005	$< 10^{-3}$	1	1.115	0.025	1
2	0.010	0.005	1	4.345	0.038	1
3	0.028	0.001	49	4.814	0.040	2210
4	0.011	0.001	62	26077.601	232.020	287204
5	0.013	0.004	61	21.335	0.144	14431
6	0.012	0.001	66	127.824	0.462	62412
7	0.023	0.004	1	0.820	0.026	1
8	0.008	0.001	41	4.332	0.037	2361
9	0.026	0.006	1	4.831	0.040	1
10	0.009	0.001	51	18.155	0.398	14837
11	0.011	0.001	95	16.147	0.072	15171
12	0.031	0.002	76	126.020	0.700	88648
13	0.011	0.001	79	70.078	0.294	44904
14	0.050	0.006	1	4.760	0.041	1
15	0.057	0.009	395	20.058	0.094	716
16	0.107	0.013	1	122.765	0.424	1

По результатам замеров видно, что для подмножества формул теории двоичных векторов 3.3 созданная реализация значительно обгоняет SMT-решатель Z3 по времени, а также на большинстве тестов выдаёт более короткую формулу.

## 5. Текущие результаты

- Выбран SMT-решатель, в рамках которого осуществлялась реализация описанного алгоритма.
- Реализован<sup>9</sup> алгоритм элиминации кванторов для следующих формул (вместо  $\leq$  может быть  $<$  или  $=$ , а вместо  $\exists - \forall$ ):
  1.  $\exists x. \bigwedge_{i,j} (g_i(\bar{y}) \leq x \wedge x \leq g_j(\bar{y}))$ , где  $g_i(\bar{y}), g_j(\bar{y})$  — термы в арифметике битовых векторов, представляющие из себя линейные комбинации констант, свободных переменных  $(\bar{y})$  и сдвигов  $1 \ll y_k$ ;
  2.  $\exists x. \bigwedge_i ((1 \ll x) \leq g_i(\bar{y}) \vee g_i(\bar{y}) \leq (1 \ll x))$ .
- Проведено сравнение реализации с элиминацией кванторов SMT-решателем Z3 и выяснено, что на указанном подмножестве она выдаёт более короткую формулу и работает быстрее, чем Z3.

---

<sup>9</sup>[https://github.com/AnzhelaSukhanova/QE\\_expPA](https://github.com/AnzhelaSukhanova/QE_expPA)

## Список литературы

- [1] Barrett Clark, Stump A., Tinelli Cesare. The SMT-LIB standard — version 2.0 // Proceedings of the 8th international workshop on satisfiability modulo theories, Edinburgh, Scotland,(SMT '10). — 2010.
- [2] Brummayer Robert, Biere Armin, Lonsing Florian. [BTOR: Bit-Precise Modelling of Word-Level Problems for Model Checking](#) // Proceedings of the Joint Workshops of the 6th International Workshop on Satisfiability Modulo Theories and 1st International Workshop on Bit-Precise Reasoning. — SMT '08/BPR '08. — New York, NY, USA : Association for Computing Machinery, 2008. — P. 33–38. — Access mode: <https://doi.org/10.1145/1512464.1512472> (online; accessed: June 9, 2021).
- [3] Btor2 , BtorMC and Boolector 3.0 / Aina Niemetz, Mathias Preiner, Clifford Wolf, Armin Biere // Computer Aided Verification / Ed. by Hana Chockler, Georg Weissenbacher. — Cham : Springer International Publishing, 2018. — P. 587–595.
- [4] Cherlin Gregory, Point Françoise. On extensions of Presburger arithmetic // Proc. 4th Easter Model Theory conference, Gross Körös. — 1986. — P. 17–34.
- [5] John Ajith K., Chakraborty Supratik. A layered algorithm for quantifier elimination from linear modular constraints // [Formal Methods in System Design](#). — 2016. — Dec. — Vol. 49, no. 3. — P. 272–323. — Access mode: <https://doi.org/10.1007/s10703-016-0260-9> (online; accessed: June 9, 2021).
- [6] Point Françoise. On the expansion  $(\mathbb{N}, +, 2^x)$  of Presburger arithmetic. — 2007. — 01.
- [7] Semenov A L. ON CERTAIN EXTENSIONS OF THE ARITHMETIC OF ADDITION OF NATURAL NUMBERS // [Mathematics of the USSR-Izvestiya](#). — 1980. — apr. — Vol. 15, no. 2. — P. 401–418. — Access

mode: <https://doi.org/10.1070/im1980v015n02abeh001252> (online; accessed: June 9, 2021).

- [8] Solving Quantified Bit-Vectors Using Invertibility Conditions / Aina Niemetz, Mathias Preiner, Andrew Reynolds et al. // Computer Aided Verification / Ed. by Hana Chockler, Georg Weissenbacher. — Cham : Springer International Publishing, 2018. — P. 236–255.