УДК 519.852.2

В.Н. ГЕРМАН

РЕШЕНИЕ ЛИНЕЙНЫХ ОГРАНИЧЕНИЙ НАД ПОЛЕМ ВЕЩЕСТВЕННЫХ И РАЦИОНАЛЬНЫХ ЧИСЕЛ

Ключевые слова: *верификация моделей, линейные неравенства, системы ограничений, ДНФ, элиминация кванторов, выпуклый полиэдр.*

ВВЕДЕНИЕ

В последние годы все больший интерес проявляется к практической верификации программ. Из-за постоянно возрастающей сложности программного обеспечения и программных проектов автоматическая верификация — все более востребованное и развивающееся направление теоретических и практических исследований. Верификация может рассматриваться как инструмент борьбы со сложностью, поскольку позволяет автоматизировать нахождение ошибок на ранних стадиях разработки программного обеспечения, минимизируя вероятность нахождения серьезных ошибок на поздних стадиях разработки программного обеспечения. Постоянно возрастающие размеры и сложность современных программных проектов делают очевидным утверждение, что борьба со сложностью — фундаментальная задача программирования. Таким образом, верификация — важная часть набора методов, для решения фундаментальных проблем программирования и уменьшения сроков разработки программного обеспечения. Важная составляющая процесса верификации — суждение об истинности или противоречивости формул в различных алгебрах и нахождения набора значений, на которых такие формулы истинны.

В таких системах верификации исходная модель программной системы описана языками спецификаций, такими как MSC, UML, SDL. Верификация таких спецификаций сводится к доказательству формул, полученных в результате преобразования спецификаций. Очевидно, что для решения и доказательства формул необходимо иметь решатели формул в заданных алгебрах (или наборах аксиом). В зависимости от ситуации и способа описания спецификаций для решения формул могут привлекаться различные решатели: формул логики предикатов первого порядка [1]; арифметики Пресбургера [2]; для оптимизационных задач; линейных неравенств и равенств над действительными числами. Входная формула анализируется системой верификации и разбивается на подформулы с пометкой типа. Затем в интерактивном режиме вызывается решатель для данного типа формул. На выходе решатель дает ответ об истинности формулы или ее упрощенную формулу. Таким образом, входная формула редуцируется, пока не будет получено решение или доказано, что система не имеет решений. В данной статье рассматривается набор методов и алгоритмов для построения одного из решателей, а именно решателя формул, состоящих из систем линейных ограничений над полем действительных или рациональных чисел. Входная формула для такого решателя содержит кванторы общности и существования: $\{\exists, \forall\}$ равенства, неравенства и отрицание равенства: $\{=, \geq, >, \leq, <\}$, логические связки — конъюнкция, дизъюнкция и отрицание: $\{\wedge, \vee, \neg\}$. Результат работы решателя — выходная формула, не содержащая кванторов, которая может содержать свободные, т.е. не связанные квантором переменные. Если входная формула замкнута, т.е. содержит только связанные переменные, на выходе решателя будет логический результат: истина или ложь. Если ресурсов для нахождения решения недостаточно или превышены значения одного из ограничений, таких как ограничения используемой памяти или ограничения времени вычисления решения, результат может быть неизвестен. Существуют системы, решающие специализированные задачи над упрощенным подмножеством входных фор-

мул. В этом случае оправдано применение более быстрых специальных алгоритмов для эффективного решения узкоспециализированной задачи. Представленный в настоящей статье алгоритм не накладывает ограничений на вид входной формулы. Формула должна быть линейная, т.е. переменная, связанная квантором или свободная, может быть умножена только на числовую константу, а значение неизвестных переменных и констант принадлежит классу действительных или рациональных чисел. Следствием такого подхода является как универсальность решателя, так и потеря быстродействия и/или невозможность получения решения для очень сложных формул. Сложность формулы прямо пропорциональна как количеству ограничений — равенств, неравенств и отрицаний равенства, так и количеству связанных переменных.

В разд. 1 представлен обзор существующих методов и алгоритмов решения проблемы разрешимости ограничений над полем вещественных чисел. В разд. 2 дана общая схема алгоритма. В разд. 3 рассмотрены алгоритмы элиминации переменной для равенства и систем ограничений. В разд. 4 предложены методы упрощения и удаления избыточности в промежуточных формулах.

1. ОБЗОР МЕТОДОВ

На данный момент существует множество различный способов решений линейных неравенств. В зависимости от подходов и целей условно их можно разделить на такие группы:

- алгоритмы, производные от алгоритма Фурье–Моцкина [3];
- симплекс-метод и его производные;
- цилиндрическая алгебраическая декомпозиция (Cylindrical Algebraic Decomposition — CAD) [4];
- быстрые алгоритмы для формул определенного вида.

Алгоритм Фурье–Моцкина — это первая разрешающая процедура для арифметики вещественных чисел. Позднее этот метод был адаптирован для решения арифметики Пресбургера, например алгоритм Omega Test [5]. Алгоритм Фурье–Моцкина достаточно прост и универсален. К недостаткам относится суперэкспоненциальная сложность алгоритма в худшем случае. Симплекс метод работает хорошо на практике и демонстрирует даже полиномиальное среднее время вычисления. Этот метод хорошо подходит для проверки, имеет ли решение система линейных неравенств, равенств и отрицаний равенства, не содержащих кванторы. Симплекс метод применим для быстрой проверки совместимости системы. Инкрементальный симплекс метод [6] позволяет эффективно проверять совместимость нового ограничения, добавленного в консистентную систему ограничений. Это свойство инкрементального симплекс метода можно применять на этапе минимизации промежуточных формул (разд. 4). Симплекс метод не позволяет упрощать формулу путем элиминации кванторов, поэтому не может использоваться как основной алгоритм для решения поставленной задачи. Алгоритм CAD, разработанный Колинсом [4], позволяет решать проблемы элиминации кванторов как для линейных, так и для нелинейных систем. Практическое применение ограничено из-за высокой вычислительной сложности. Следующий подкласс — это алгоритмы, которые работают только для формул определенного вида, например для формул, которые содержат только две переменные в каждой системе ограничений или, другими словами, конъюнкции неравенств. Такой подход позволяет решать задачу практически за линейное время и пространство [6]. Экспоненциальной сложностью элиминации переменных обладает метод экстремальных точек, предложенный Лассером [8].

2. ОБЩАЯ СХЕМА АЛГОРИТМА

Внутри системы формула эквивалентно преобразуется таким образом, что содержит только неравенства $\{<, \leq\}$ и квантор существования. Такой базис требует преобразования отрицания равенства в дизъюнкцию неравенств, что может при-

вести к экспоненциальному росту количества неравенств на этапе приведения формулы к дизъюнктивной нормальной форме (ДНФ). Методы редукции размера промежуточной формулы во время приведения к ДНФ рассмотрены в подразд. 3.4. На верхнем уровне решатель работает по такому алгоритму:

1) преобразование кванторов общности в кванторы существования, см. 3.1
2) приведение функции к ДНФ;
3) элиминация самого внутреннего квантора;
4) упрощение;
5) если имеются еще кванторы, то переход к п. 2;
6) вывод результата.

3. ЭЛИМИНАЦИЯ КВАНТОРОВ

3.1. Общие преобразования формулы. Формула приводится к минисферной нормальной форме следующим образом. Квантор общности заменяется квантором существования. Квантор существования переносится внутрь дизъюнкции. Отрицание дизъюнкции выносится за скобки и, наконец, двойное отрицание формулы заменяется самой формулой. Формальные правила такого преобразования:

$$\forall v\colon f \to \neg\exists v\colon \neg f\,,$$
$$\exists v\colon (f_1 \vee \ldots \vee f_n) \to \exists v\colon f_1 \vee \ldots \vee \exists v\colon f_n\,, \qquad (1)$$
$$\neg(f_1 \vee \ldots \vee f_n) \to \neg f_1 \wedge \ldots \wedge \neg f_n\,, \quad \neg\neg f \to f.$$

3.2. Решение равенств. Имеем систему линейных неравенств и равенств. Переменные могут входить как свободно, так и быть связанными квантором существования. Если связанная переменная входит в равенство, то ее элиминация осуществляется подстановкой значения такой переменной во все другие равенства и неравенства. Этот шаг повторяется, пока все равенства и/или связанные переменные не будут элиминированы. Все равенства и неравенства внутри системы представлены в виде матрицы. Одна подстановка переменной эквивалентна одному шагу в методе решения систем линейных равенств Гаусса [9]. Если система равенств содержит свободные переменные, то она должна быть упрощена и представлена в системе в решенном виде. Подробнее этот метод описан в подразд. 4.1.

3.3. Решение неравенств. Перейдем к методу элиминации переменных в системе линейных неравенств, открытым Фурье (1826 г.), его можно рассматривать как расширение метода Гаусса для неравенств 3.2. Этот метод был на время забыт и изобретен заново Динесом и Моцкиным [10].

Основная идея метода довольно проста и состоит в применении к исходной системе неравенств таких преобразований:

$$a \le bx \wedge cx \le d \leftrightarrow \frac{a}{b} \le x \le \frac{d}{c} \to \frac{a}{b} \le \frac{d}{c}\,. \qquad (2)$$

Здесь x — переменная для элиминации; a, b, c, d — числовые константы. Приведенные выше преобразования необходимо применить ко всем сочетаниям право- и левосторонних неравенств. Под правосторонним неравенством подразумевается такое неравенство, в котором переменная для элиминации или, другими словами, связанная переменная с положительным коэффициентом находится справа. Под левосторонним неравенством будем понимать такое неравенство, в котором переменная для элиминации с положительным коэффициентом находится слева. В предложенном решателе применяются эквивалентные преобразования, неравенства заданы в матричной форме. Рассмотрим систему неравенств, заданную в виде матрицы. В этой системе могут содержаться как равенства, так и неравенства. Тип определенной системы определяется вектором типов. Длина такого вектора совпадает с количеством неравенств и строк в матрице. Рассмотрим

алгоритм Фурье–Моцкина в матричной форме. Предположим, имеется система неравенств $S[m, n] = Ax \le b$, которая содержит m линейных неравенств и n переменных. Эта системе задана матрицей коэффициентов (A) и вектором констант (b). Для простоты будем полагать, что в системе встречаются только нестрогие неравенства, затем рассмотрим решение для общего случая:

$$\begin{bmatrix} a_{11} & a_{21} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \le \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ bm \end{bmatrix}. \tag{3}$$

Для нахождения решения системы неравенств нужно исключать переменные из данной системы. Для этого возьмем произвольный столбец матрицы, пусть это будет первый столбец. Вектор коэффициентов такой матрицы $a_1^T = [a_{11}\ a_{21}\ \dots a_{m1}]$. В этом векторе могут содержаться как положительные, так и отрицательные коэффициенты. Если вектор a_1^T содержит только положительные или только отрицательные коэффициенты, то такая система всегда имеет решение, независимо от переменной. Поэтому этот столбец вычеркивается, обнуляя все значения коэффициентов вектора a_1^T. Если система не решается тривиально, то строим элиминирующую матрицу $C_1 \ge 0$ такую, что

$$\frac{C_1 \cdot Ax \le C_1 \cdot b}{A'x' \le b'} \tag{4}$$

после умножения матрицы коэффициентов и вектора констант на элиминирующую матрицу C_k полученная матрица A' не содержит столбца с индексом k. Вектор b' содержит обновленные коэффициенты. Это означает, что результирующая система эквивалентна исходной, но уже не зависит от переменной, которая была на позиции k. Рассмотрим построение такой матрицы. Возьмем все возможные сочетания (произведение) положительных и отрицательных коэффициентов.

Например, пусть $a_{ij}^+ > 0$, $a_{ij}^- < 0$, $i \in [1 \dots m]$, $j \in [1 \dots n]$:

$$A = \begin{bmatrix} a_{11}^+ & a_{12} & \dots & a_{1n} \\ a_{21}^- & a_{22} & \dots & a_{2n} \\ a_{31}^+ & a_{32} & \dots & \dots \\ a_{41}^- & a_{42} & \dots & a_{4n} \end{bmatrix}, \tag{5}$$

$$C = \begin{bmatrix} -a_{21}^- & a_{11}^+ & 0 & 0 \\ -a_{41}^- & 0 & 0 & a_{11}^+ \\ 0 & a_{31}^+ & -a_{21}^- & 0 \\ 0 & 0 & -a_{41}^- & a_{31}^+ \end{bmatrix}. \tag{6}$$

Пусть в векторе a_1^T количество положительных коэффициентов k^+, количество отрицательных коэффициентов — k^- и количество нулевых коэффициентов — k^0. Тогда количество строк в матрице C равно

$$m' = k^+ k^- + k^0. \tag{7}$$

Количество столбцов равно количеству столбцов матрицы A:

$$C_1 A = \begin{bmatrix} -a_{21}^- & a_{11}^+ & 0 & 0 \\ -a_{41}^- & 0 & 0 & a_{11}^+ \\ 0 & a_{31}^+ & -a_{21}^- & 0 \\ 0 & 0 & -a_{41}^- & a_{31}^+ \end{bmatrix} \begin{bmatrix} a_{11}^+ & a_{12} & \dots & a_{1n} \\ a_{21}^- & a_{22} & \dots & a_{2n} \\ a_{31}^+ & a_{32} & \dots & \dots \\ a_{41}^- & a_{42} & \dots & a_{4n} \end{bmatrix}. \tag{8}$$

Результирующая система A' имеет следующий вид:

$$A' = \begin{bmatrix} -a_{21}^{-}a_{11}^{+} + a_{11}^{+}a_{21}^{-} & -a_{21}^{-}a_{12} + a_{11}^{+}a_{22} & \dots & -a_{21}^{-}a_{1n} + a_{11}^{+}a_{2n} \\ -a_{41}^{-}a_{11}^{+} + a_{11}^{+}a_{41}^{-} & -a_{41}^{-}a_{12} + a_{11}^{+}a_{42} & \dots & -a_{41}^{-}a_{1n} + a_{11}^{+}a_{4n} \\ a_{31}^{+}a_{21}^{-} - a_{21}^{-}a_{31}^{+} & a_{31}^{+}a_{22} - a_{21}^{-}a_{32} & \dots & \dots \\ a_{41}^{-}a_{31}^{+} + a_{31}^{+}a_{41}^{-} & a_{41}^{-}a_{32} + a_{31}^{+}a_{42} & \dots & a_{41}^{-}a_{32} + a_{31}^{+}a_{42} \end{bmatrix}. \quad (9)$$

После упрощения очевидно, что первый столбец матрицы A' состоит только из нулевых элементов. Таким образом, результирующая система больше не зависит от переменной, задающей первый столбец:

$$A' = \begin{bmatrix} 0 & a'_{12} & \dots & a'_{1n} \\ 0 & a'_{22} & \dots & a'_{2n} \\ 0 & a'_{32} & \dots & a'_{3n} \\ 0 & a'_{42} & \dots & a'_{4n} \end{bmatrix}, \; b' = C_1 b = \begin{bmatrix} b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \end{bmatrix}. \quad (10)$$

В приведенном примере два положительных и два отрицательных коэффициента и отсутствуют нулевые коэффициенты. В результате количество неравенств в результирующей и исходной матрице одинаково. В частности этим свойством отсутствия роста ограничений в системах с двумя переменными и двумя неравенствами пользуются некоторые специализированные решатели. Рассчитаем количество результирующих неравенства для произвольного количества неравенств. Пусть количество неравенств — m, а количество положительных и отрицательных коэффициентов равно и отсутствуют независимые ограничения. Тогда количество строк результирующей матрицы после элиминации одной переменной равно

$$\left(\frac{m}{2}\right)^2. \quad (11)$$

При количестве связанных переменных n размер результирующей формулы может быть

$$\left(\frac{m}{2}\right)^{2^n}. \quad (12)$$

Таким образом, данный алгоритм обладает суперэкспоненциальной сложностью, в худшем случае, для произвольного количества ограничений и переменных. Улучшить практическую реализацию можно благодаря тому, что зачастую после элиминации переменной результирующее множество неравенств избыточно. Таким образом, применение эффективного алгоритма нахождения и удаления избыточных неравенств позволяет использовать алгоритм Фурье–Моцкина для решения реальных задач. Один из таких алгоритмов рассмотрен в 4.2.

В начале раздела предполагалось, что данная система может содержать только нестрогие неравенства. В общем случае в системе могут также быть равенства и нестрогие неравенства. Тип ограничения определяется дополнительным вектором типов, размер которого совпадает с количеством строк матрицы A, т.е. количеством ограничений в системе. Все равенства будут перенесены в результирующую систему, поскольку коэффициент при элиминируемой переменной для равенства всегда равен нулю. Это утверждение справедливо вследствие того, что равенства с ненулевым коэффициентом уже были элиминированы на предыдущем шаге (см. 3.2). Будет результирующее неравенство строгим или нестрогим, определяется при построении элиминирующей матрицы C. Если два выбранных неравенства имеют одинаковый тип, то результирующее неравенство будет иметь такой же тип, в противном случае результирующее неравенство будет строгим.

Альтернатива элиминации Фурье-Моцкина предложена в [11]. Исключение переменной в системе линейных неравенств осуществлено с помощью расчета экстремальных точек. Этот метод исключает все переменные одновременно с помощью

квазидуального представления данной системы линейных ограничений S с учетом переменных, которые должны быть исключены. Метод следует из обобщенной проблемы линейной оптимизации с векторнозначной (vector valued) целевой функцией Ф над ограниченным полиэдром. Затем экстремальные точки многогранника $\Phi(P)$ определяют конечную систему линейных неравенств, которые описывают проекцию решений множества S в пространство оставшихся переменных. Еще одна эффективная альтернатива предложена в [12].

3.4. Отрицания равенств. Отрицания равенств играют важную роль при решении линейный неравенств, поскольку от того, как будут решаться отрицания равенств, зависит вычислительная сложность и без того сложного (см. 3.3) алгоритма элиминации кванторов для действительных чисел. Наивная реализация — это исключение неравенств из системы путем представления отрицаний равенства, будем их также называть негативными ограничениями, через строгие неравенства:

$$x \neq b \leftrightarrow b < x \vee x < b\,. \tag{13}$$

Таким образом, результирующая формула больше не будет содержать неравенств. Недостаток такого подхода заключается в его сложности. Предположим, исходная система имеет вид

$$x_1 \neq b_1 \wedge x_2 \neq b_2 \wedge \ldots \wedge x_n \neq b_n \wedge f, \tag{14}$$

где f — произвольная линейная формула, не содержащая отрицания равенства. Тогда после одного применения правила и вынесения дизъюнкции формула будет иметь вид

$$(x_1 < b_1 \wedge x_2 \neq b_2 \wedge \ldots \wedge x_n \neq b_n \wedge f) \vee (x_1 > b_1 \wedge x_2 \neq b_2 \wedge \ldots \wedge x_n \neq b_n \wedge f)\,. \tag{15}$$

Каждый из конъюнктив дизъюнкции породит еще два неравенства. Таким образом, результирующее количество неравенств после замены всех отрицаний равенства неравенствами будет:

$$2^n\,. \tag{16}$$

Уже при исходных, всего при десяти ($n = 10$) отрицаниях равенства, результирующая система будет содержать 1024 неравенства. Для действительных и рациональных чисел есть лучший способ. Система обобщенных линейных ограничений [13] — это конечная конъюнкция позитивных и негативных линейных ограничений. Ключ для обработки обобщенных линейных ограничений — независимость негативных линейных ограничений между собой. Система имеет решения тогда и только тогда, когда позитивные ограничения f вместе с одним из негативных ограничений $x \neq b_i \wedge f$, $i \in [1 \ldots n]$, имеют решения. Геометрически это означает, что полиэдр принадлежит конечному множеству аффинных множеств тогда и только тогда, когда он принадлежит одному из них. В [13] показано, что метод Фурье-Моцкина можно легко адаптировать для вычисления аффинной оболочки полиэдра. При переходе к системе обобщенных линейных ограничений необходима модификация алгоритма элиминации переменой. Два варианта алгоритма представлены в [14].

4. УДАЛЕНИЕ ИЗБЫТОЧНОСТИ

4.1. Упрощение системы равенств. Конъюнкция линейных равенств $Ex = f$ является частью системы ограничений. В исходной формуле или формуле, полученной в результате элиминации переменной, система линейных равенств может быть избыточна и/или находиться в нерешенном виде. Такое представление избыточно, поскольку бесконечное количество равенств может задавать одно и то же решение. Поэтому система равенств должна быть представлена в решенной форме (solved form). Известно, что система линейных равенств не противоречива тогда и только тогда, когда она может быть представлена в решенной форме. Для решения системы равенств можно применить классические алгоритмы ли-

нейной алгебры [9]. С практической точки зрения существует множество библиотек для решения системы линейных равенств, например почти стандартная для C++ библиотека Boost Libraries [15].

При реализации может возникнуть ситуация, когда одна система равенств содержит несколько независимых систем равенств, в то время как стандартная функция решения линейных равенств ожидает на входе только одну систему равенств, без избыточных неравенств. Для решения этой проблемы необходимо:

- декомпозировать систему $Ex = f$ на независимые системы равенств $Ex = f \to \{ E_1 x = C_1 , \ldots, E_n x = f_n \}$;
- нормализировать и удалить избыточность для каждой из независимых систем, $Simplify\,(Normalize\,(E_i x = f_i))$, $i \in [1 \ldots n]$;
- объединить все решения в одну систему $SolvedForm\,(Ex = f)$.

Предполагается, что $SolvedForm\,(Ex = f)$ возвращает значение ложь, если система не может быть решена. Замечательным свойством такой системы является возможность получения ее решения путем подстановки конкретного значения вместо одной из переменных.

4.2. Упрощение системы неравенств и равенств. Вторая часть системы линейных ограничений — это система линейных неравенств и равенств $\{Ax < B_1, Ax \le B_2, Ex = f\}$. Упрощение системы неравенств позволяет расширить область практического применения решателя линейных неравенств. Упрощение неравенств не может рассматриваться без равенств, так как в результате таких упрощений могут образоваться новые равенства. Поскольку количество неравенств на каждом шаге элиминации переменной может возрастать стремительно (см. п. 3.3), то борьба с избыточностью — неотъемлемая задача, которую нужно эффективно решать. Система ограничений должна быть нормализирована. Это значит, что все неравенства, содержащие одинаковые переменные, должны иметь одинаковые по модулю коэффициенты относительно одной из переменных. Такая нормализация гарантирует, что коэффициенты параллельных ограничений будут в точности одинаковы или отличаться знаком. Очевидно, что приведенное утверждение распространяется только на целочисленные и рациональные коэффициенты. В основе техники лежит идея параллельности полупространств (гиперплоскостей), которые задают исходную систему:

$$\begin{aligned} &a_1 x_1 + f_1 < b_1 \wedge a_1 x_1 + f_1 < b_2 \to a_1 x_1 + f_1 < \min(b_1 , b_2), \text{ если } a_1 > 0, \\ &a_1 x_1 + f_1 < b_1 \wedge a_1 x_1 + f_1 < b_2 \to a_1 x_1 + f_1 < \max(b_1 , b_2), \text{ если } a_1 < 0. \end{aligned} \tag{17}$$

Выберем два неравенства, отличающиеся только константой, и в зависимости от коэффициента при выбранной переменной заменим эти два неравенства одним, константа которого минимальна, если коэффициент при выбранной переменной положительный, или неравенство с максимальной константой в противном случае. В рассмотренном выше примере предполагалось, что имеем дело только со строгими неравенствами. Но ситуация несколько усложняется, когда рассматриваются попарные сочетания строгих и нестрогих неравенств и равенства между собой. В этом случае нужно дополнительно рассматривать сочетания типов неравенств T_1, T_2 или, другими словами, транзитивное замыкание ограничений. В этом случае тип результирующего неравенства будет вычислен по следующей таблице

T_1	T_2		
	$a = b$	$a \le b$	$a < b$
$=$	$=$	$=$	Ложь
$\le$	$=$	$\le$	$<$
$<$	Ложь	$<$	$<$

Некоторые сочетания вызывают противоречие в исходной системе. В таблице это помечено как ложь. В этом случае $RemRedundant\,(\{Ax < B_1, Ax \le B_2, Ex = f\})$

возвращает ложь. После того, как все параллельные полупространства и гиперплоскости, имеющие в точности одинаковые коэффициенты при переменных, упрощены, можно приступить к упрощению параллельных ограничений, отличающихся знаком при коэффициентах:

$$a_1x_1 + f_1 \le b_1 \wedge -a_1x_1 - f_1 \le b_1 \to a_1x_1 + f_1 = b_1. \qquad (18)$$

Если неравенства, имеющие противоположные коэффициенты при выбранной переменной, содержат различные константы b, то проверяем, задают ли эти два ограничения выпуклый полиэдр. В этом случае ничего упростить нельзя. Если нет, имеем противоречие. Если и константы равны, то результирующее ограничение вычисляется по следующей таблице

T_1	T_2		
	$a = b$	$a \le b$	$a < b$
$=$	$=$	$\le$	Ложь
$\ge$	$\ge$	$=$	Ложь
$>$	Ложь	Ложь	Ложь

В приведенной выше таблице имеются неравенства $\{>, \ge\}$ только для наглядности, подразумевая, что мы имеем дело с отрицательным коэффициентом при выбранной переменной и неравенствами $\{<, \le\}$. На этом этапе пару неравенств можно заменить равенством (18) или найдено противоречие. При реализации важно быстро находить параллельные ограничения как с одинаковыми, так и с противоположными коэффициентами. Одним из возможных решений есть применение хеширования таким образом, чтобы по данному неравенству и системе ограничений функция поиска возвращала итератор по списку гиперплоскостей, параллельных заданному ограничению.

4.3. Операция *gist*. Ранее рассматривали упрощения конъюнкции или системы линейных ограничений, но только этой техники недостаточно для получения результатов и промежуточных вычислений, не содержащих избыточности. Операция *gist* предложена в [16] и может использоваться, как показано ниже, для упрощения дизъюнкции систем линейных ограничений.

Определение. *gis p given q* — конъюнкция, которая содержит такое минимальное подмножество ограничений из p, что $((gist\ p\ given\ q) \wedge q) = (p \wedge q)$.

Интуитивно *gis p given q* — это новая информация, которая содержится в p при условии, что q известно. По определению *gist* всегда содержит не больше ограничений, чем исходная система ограничений.

Положим, что система ограничений q разрешима, тогда алгоритм вычисления gist p given q :=

1) if (p is *true*) than
2) return *true*
3) else
4) let c is constraint in p
5) if $(p^c_{\sim(c)}) \wedge q$ is *consistent* than
6) return ($c \wedge$ gist p^c_{True} *given* $(q \wedge c)$)
7) else
8) return (*gist* p^c_{True} *given* q)

Здесь p^{oldc}_{newc} — это p, в котором элемент *oldc* заменен элементом *newc*. Такой алгоритм трудоемкий, поскольку содержит многократные проверки на непротиворечивость систем линейных ограничений, каждая из которых достаточно трудоемка. Более эффективно проблема решается проверкой специальных случаев.

- Для каждого ограничения e в множестве ограничений p проверяем, вытекает ли e из любого ограничения в p или q. Если так, то e избыточно и не попадет в *gist*.

- Проверяем, имеет ли некоторая из переменных верхние ограничения в p, но не имеет в q. Если так, то хотя бы одно из верхних ограничений должно быть в *gist*. Аналогичная проверка делается для нижних ограничений.
- Если не существует такое ограничение e' в p или q, что скалярное произведение нормалей e и e' позитивно, то e должно быть в *gist*.

При этом если ограничение определено как избыточное, то оно больше не участвует в дальнейших выводах об избыточности остальных неравенств. Зачастую такая процедура быстрой проверки специальных случаев полностью вычисляет *gist* или значительно упрощает проблему перед вызовом общего алгоритма.

4.4. Упрощение дизъюнкции ограничений. Определенная в 4.3 операция *gist* может использоваться для проверки тавтологий. Если $(gist\ \ p\ given\ q) == Истина$, то $p \Rightarrow q$ является тавтологией.

Пусть имеем формулу $A \vee B$, где A, B — произвольные системы ограничений. Проверим с помощью операции *gist* справедливость $A \Rightarrow B$. Если да, то $A \vee B \xrightarrow{A \Rightarrow B} A$.

Используя такой метод, можно упрощать дизъюнкции произвольного количества систем ограничений. Для проверки тавтологии используется функция *gist*, которая прекратит работу, как только станет очевидно, что результат операции не будет равен Истина.

5. ПРИВЕДЕНИЕ К ДНФ

Формула должна быть приведена к нормальной форме для начала работы алгоритма элиминации кванторов для каждого из конъюнктов. Недостатком «наивного» алгоритма приведения к ДНФ является факт, что количество литералов может возрастать экспоненциально во время работы алгоритма. Для борьбы с увеличением размера как промежуточной, так и результирующей формулы в решателе необходимо реализовать приведение к ДНФ с учетом того факта, что истинность конечной формула может быть доказана. В случае систем линейных ограничений можно однозначно доказать, задает конъюнкция ограничений выпуклый многогранник или пустое множество. Следующие проверки выполняются при вычислении ДНФ:

- при добавлении нового ограничения в систему ограничений проводится проверка на противоречивость новой системы, если возникло противоречие, вся система удаляется;
- проверка и упрощение избыточности при объединении двух разных конъюнкций ограничений в один выпуклый полиэдр.

Реализованы также более сложные эвристики с использованием операции *gist*:

- удаление повторяющихся дизъюнктов и тавтологий (см. 4.4);
- минимальное представление формул типа $A \vee \neg B$ с использованием *gist*.

По определению $A \wedge (gist\ B\ given\ A) \equiv (A \wedge B)$. Тогда если формула содержит конъюнкцию системы ограничений и отрицание другой системы ограничений, то при раскрытии отрицания можно использовать операцию *gist* для уменьшения количества ограничений в отрицании:

$$A \wedge \neg B \equiv A \wedge \neg(A \wedge B),$$

$$A \wedge \neg (A \wedge B) \equiv A \wedge \neg (A \wedge (gist\ B\ given\ A)) ,$$

$$A \wedge \neg (A \wedge (gist\ B\ given\ A)) \equiv A \wedge \neg (gist\ B\ given\ A) .$$

Результирующая система $A \wedge \neg (gist\ B\ given\ A)$ зачастую имеет меньшее количество ограничений, чем система $A \wedge \neg B$. Это очевидно, поскольку $gist\ B\ given\ A$ по определению всегда имеет равное или меньшее количество выражений, чем B.

Кроме представленных методов для дальнейшей редукции формул можно использовать алгоритмы, гарантировано имеющие меньшую сложность, чем наивный

алгоритм. Одна из первых разработок — это субэкспоненциальный алгоритм, описанный в [17]. В поздних работах [18 и 19] предлагаются различного рода усовершенствования, позволяющие ускорить нахождение ДНФ.

6. ПРАКТИЧЕСКИЕ РЕЗУЛЬТАТЫ

Большинство из представленных методов реализовано в решателе линейных ограничений для действительных и рациональных чисел. Этот решатель встроен в систему алгебраического программирования АПС [20]. Существует также не зависимая от системы АПС версия решателя, доступная для свободной загрузки Linear Solver [21]. На базе АПС с использованием решателя успешно проведена верификация двух разных моделей модели протокола TTP [22] (подробнее см. в [23]). Трехузловая модель представлена в [24] и состоит из 21-й переменной и 33-х ограничений. Такая формула решается за несколько минут на современном ПК с использованием разработанного решателя. Время решения зависит от значения коэффициентов: tau1, tau2, tau3 $\in$ (0 ... 1], варьируется от нескольких секунд до нескольких минут.

Приведем пример двухузловой модели:

```
Resolve[ForAll [{d, p, c, newc, newp},
(
    !((newc == – D*(c + d) – p) &&
    (newp == p + c + d) &&
    (–1 ≤ d) && (d ≤ 1) && (–M ≤ p) && (p ≤ M) &&
    (1 – M ≤ p + c) && (p +c ≤ M – 1) &&
    (– M ≤ c) && (c ≤ M)
) || (
    (1 – M ≤ newp + newc) && (newp + newc ≤ M – 1)
)
) && (M > 1) && (D > 0) && (D <= 1)
], {M, D}, Reals]

Out = (M > 1 && 2/(1 + M) ≤ D ≤ 1)
```

Требуется найти зависимость свободной переменной M от параметра D, при котором представленная выше формула истинна. Приведення система записана с использованием языка пакета Wolfram Mathematica 5.2. Mathematica использует CAD-алгоритм для элиминации кванторов и дает возможность элиминировать кванторы и для задач высшего порядка. Как Mathematica, так и Linear Solver успешно справились с двухузловой моделью, причем Mathematica позволяет использовать коэффициент D в символьном виде. Трехузловая модель [24] проблемная для пакета Mathematica 5.2 на машине с 2 Gb оперативной памяти. Вычисления в течение нескольких минут заканчиваются сообщением о нехватке памяти. В то же время Linear Solver решает трехузловую модель, как отмечалось выше.

Поддержка символьных коэффициентов не предусмотрена в текущей версии Linear Solver и может быть реализована в будущих версиях. Интересной задачей для предстоящих исследований может быть также сравнительное тестирование производительности с существующими решателями.

ЗАКЛЮЧЕНИЕ

Создание хорошего решателя для натуральных и вещественных чисел — довольно сложная задача, для решения которой не существует единого эффективного алгоритма. Общий алгоритм элиминации кванторов совместно с наивным алгоритмом приведения к ДНФ, в худшем случае, имеет суперэкспоненциальную сложность, что делает его пригодным для решения только ограниченного круга задач. Нетривиальные алгоритмы зачастую требуют поиска и обработки большого количества источников, исследований и экспериментов. В данной работе рас-

смотрен набор методов, на основе которых построен решатель для натуральных и вещественных чисел, который применяется для решения многих практических задач верификации. Созданная реализация решателя позволяет получать практические результаты для реальных приложений, хотя и обладает ограничениями, которые еще предстоит устранить для дальнейшего расширения использования, также приведены ссылки на современные методы, альтернативы и пути дальнейших исследований.

СПИСОК ЛИТЕРАТУРЫ

1. http://en.wikipedia.org/wiki/First-order_logic
2. http://coq.inria.fr/V8.1pl4/refman/Reference-Manual022.html
3. http://en.wikipedia.org/wiki/Fourier%E2%80%93Motzkin_elimination
4. Collins G. E. Quantifier elimination by cylindrical algebraic decomposition — twenty years of progress // Quantifier elimination and cylindrical algebraic decomposition (Ed. B. F. Caviness and J. R. Johnson). — New York: Springer–Verlag, 1998. — P. 8–23.
5. http://www.umiacs.umd.edu/~hismail/Omega/node3.html
6. Badros G., Borning A., Stuckey P. The Cassowary linear arithmetic constraint solving algorithm // ACM transactions on computer-human interaction (TOCHI). — 2001. — **8**(4) — P. 267–306.
7. http://research.microsoft.com/en-us/um/redmond/projects/z3/index.html
8. Lassez J. Querying constraints, 1990 ACM 089791-352-3/90/0004/0288.
9. http://ru.wikipedia.org/wiki/Метод_Гаусса
10. Eaves B.C. and Rothblum U.G. 1992. Dines–Fourier–Motzkin quantifier elimination and an application of corresponding transfer principles over ordered fields // Math. Program. — 1992. — **53**, N 3. — P. 307–321.
11. Huynh T., Joskowicz L., Lassez C., and Lassez J. Practical tools for reasoning about linear constraints. Fundam. — 1991. — **3–4.** — P. 357–380.
12. Lassez C. and Lassez J-L. Quantifier elimination for conjunctions of linear constraint via a convex hull algorithm: (Techn. Rep.), IBM T.J. Watson / Research Center, Yorktown Heights, 1991.
13. Lassez J.L. and McAloon K. A canonical form of generalized linear constraints // J. of Symbolic Comput. — 1992. — **33**. — 147–160 p.
14. Imbert J-L. Variable elimination for disequations in generalised liner constraint systems // The computer J. — 1993. — **36**, N 5.
15. Boost Libraries www.boost.org
16. Pugh W., Wonnacott D. Going beyond integer programming with the omega test to eliminate false data dependences. — December, 1992.
17. Bshouty N. A subexponential exact learning algorithm for DNF using equivalence queries. Information Processing Letters. — 1996. — **59.** — P. 3739.
18. Tarui J. and Tsukiji T. Learning DNF by approximating inclusionexclusion formulae // Proc. IEEE conf. on Comput. Complexity. — 1999. — 215–220 p.
19. Adam R. Klivans Rocco A. Servedio. Learning DNF in Time.
20. http://www.springerlink.com/content/htq68m20q121t3jw/
21. http://code.google.com/p/constraints-solver-over-r/
22. http://en.wikipedia.org/wiki/Time-Triggered_Protocol
23. Годлевский А.Б., Свиргуненко С.Н. О применении линейных моделей при оптимизации параметров TTP протоколов // Кибернетика и системный анализ. — 2006. — № 3. — С. 24–31.
24. http://www.risc.uni-linz.ac.at/projects/intas/Timisoara/Presentations/German/ Grman.pdf

Поступила 15.04.2010