УДК 004.421.6

М.С. ЛЬВОВ, В.С. ПЕСЧАНЕНКО, А.А. ЛЕТИЧЕВСКИЙ, Ю.Г. ТАРАСИЧ, А.С. БАЕВ

АЛГОРИТМ И ИНСТРУМЕНТЫ ПОСТРОЕНИЯ КАНОНИЧЕСКИХ ФОРМ ЛИНЕЙНЫХ ПОЛУАЛГЕБРАИЧЕСКИХ ФОРМУЛ

Аннотация. Получены результаты испытаний инструментов упрощения формул, а также описан алгоритм построения канонических форм линейных полуалгебраических формул (ЛПФ). Основным результатом работы является определение канонической формы ЛПФ, обладающей свойством единственности и другими полезными свойствами, а также описание алгоритма ее построения.

Ключевые слова: системы линейных неравенств, алгебраическое программирование, верификация программного обеспечения, канонические формы, линейные полуалгебраические формулы.

ВВЕДЕНИЕ

Возрастающая сложность программного обеспечения (ПО) приводит к увеличению в нем количества ошибок. В связи с внедрением информационных технологий во все сферы деятельности человека это вызывает тяжелые последствия. Уже в настоящее время известны примеры серьезных ошибок в ПО, ставшие причиной гибели людей, космических аппаратов или масштабных нарушений работы инфраструктурных сетей. Для обеспечения корректности и надежности работы таких программных средств и систем большое значение имеют методы верификации, позволяющие не только выявлять ошибки до написания исходного кода системы, но и доказывать различные свойства моделей, построенных по уже написанному коду. В общем случае методы доказательства формальных моделей можно разделить на статические и динамические, конкретные и символьные. При доказательстве конкретных моделей получено большое колличество результатов в отличие от символьного моделирования, когда каждое состояние представляет собой некую формулу, покрывающую некоторое множество конкретных состояний. Но несмотря на то, что количество состояний символьной модели, которые задаются формулой, меньше, чем в конкретной, при символьном моделировании возникает проблема определения достижимости уже пройденных состояний в процессе использования динамических методов и нахождения довольно громоздких формул, сложность которых ввиду отсутствия некоторой канонической или нормальной формы может постепенно возрастать. Это означает, что в символьном моделировании проблема построения каноничеких и нормальных форм является одной из наиболее важных. В настоящее время проводится достаточно много исследований в вопросах реализации инструментов решения данной проблемы. В разд. 1 настоящей статьи показаны результаты экспериментов некоторых из них, в разд. 2 описан алгоритм построения канонических форм линейных полуалгебраических формул.

1. ИНСТРУМЕНТЫ УПРОЩЕНИЯ ФОРМУЛ: ЭКСПЕРЕМЕНТАЛЬНЫЕ РЕЗУЛЬТАТЫ

За последние два десятилетия разработано достаточно много как коммерческих, так и свободно распространяемых систем верификации для программных и аппаратных систем. Разрабатывается большое число верификаторов из-за отсутствия приемлемого для промышленности универсального метода проверки правильности разрабатываемых систем. Верификаторы решают узкоспециализированные проблемы, например проверку свойств, специфических для языков

реализации систем. Основное внимание при анализе систем построения доказательств было сосредоточено на наличии инструментов упрощения формул. В настоящей работе рассмотрены такие системы построения доказательств как CVC4 [1], MathSAT5 [2], QEPCAD [3, 22], Singular [4], COCOA [5], MiniZinc [6], STP [7], RedLog [8], Satallax [9], Isabelle [10], E-SETHEO [11], Minisat [12], SMTInterpol [13], Paradox [14], Gandalf [15], Vampire [16], Z3 [21].

Результаты, полученные при испытании инструментов упрощения формул, приведены в табл. 1. Формулы, которые использовались для проверки инструментов упрощения, представлены в Приложении 1. В таблице не были отображены системы, которые не упростили ни одной формулы, не имели соответствующего функционала или ввиду ряда причин не смогли быть использованы для испытания (отсутствие исходного кода, проблемы с установкой).

Как видно из табл. 1, полностью с поставленной задачей справилась только одна система — Isabelle2016 [10]. Эта система легко устанавливается и имеет хорошую документацию. Она поддерживает работу с логикой первого и высшего порядков. Относительно упрощения формул эта система является хорошей. Правила упрощения формул задаются пользователем. В этом плане она работает как система переписывающих правил, поэтому ответственность за упрощения лежит на самом пользователе. Система написана на языке SML, поэтому она интегрируется достаточно сложно с языком C++.

Система QEPCAD [3] с точки зрения упрощения формул работает хорошо, но имеет достаточно сложный и громоздкий синтаксис и не содержит хорошей документации. Для создания подобных систем наиболее используемыми языками являются C/C++ и ML-язык (OCaml, SML). Во всех системах сталкиваются с проблемой переписывания формул и приведением их к определенному формату, а это проблематично особенно, если все операции нужно выделить скобками, как, например, в языке SMT-LIB. Что касается таких инструментов как системы E-SETHEO [11], TPS/ETPS [13], Paradox 3.0. [14], Gandalf c-2.6.r1 [15], Vampire 2.6 [16], то некоторые из них, по-видимому, являются «заброшенными» проектами (E-SETHEO, Gandalf c-2.6.r1), к остальным пруверам доступ закрыт. Исходные коды перечисленных пруверов из CASC содержат ошибки, поэтому инструменты испытать не удалось. Соответственно разработка системы упрощения формул, которая позволяла бы решить поставленые задачи, остается актуальной. Не менее важной является также необходимость реализации соответствующих алгоритмов, позволяющих выполнять упрощение формул на более эффективном уровне.

Таблица 1

Номер формулы f	Результаты, полученные при упрощении формул системами			
	CVC4	QEPCAD	RedLog	Isabelle
f1	+	++	–	++
f2	+	++	–	++
f3	+	++	–	++
f4	+	++	–	++
f5	+	–	–	++
f6	+	–	–	++
f7	+	–	–	++
f8	+	++	++	++
f9	+	++	–	++
f10	++	++	–	++

Примечание. Формула упрощения неудовлитворительная (+); формула упрощена (++); не удалось упростить формулу или нет программных возможностей для ее упрощения (–).

2. КАНОНИЧЕСКИЕ ФОРМЫ ЛИНЕЙНЫХ ПОЛУАЛГЕБРАИЧЕСКИХ ФОРМУЛ

Введем основные обозначения: $X = \{x_1, \ldots, x_n\}$ — n-мерный вектор переменных; Q — конструктивное линейно-упорядоченное поле, называемое полем коэффициентов; Q^n — векторное пространство над Q; $\bar{a} = (a_1, \ldots, a_n) \in Q^n$ — вектор числовых коэффициентов; $b \in Q$ — числовой коэффициент; $LF(\bar{a}, X)$ — линейная форма от $X, \bar{a} : LF(a, X) \stackrel{\text{df}}{=} a_1 x_1 + \ldots + a_n x_n$.

Линейное неравенство (ЛН) $E(\bar{a}, X, c)$ имеет вид $LF(\bar{a}, X) \le c$ и обозначается $E(\bar{a}, X, c)$. Для сокращения записи линейное неравенство иногда записывается как $E(X)$, указывая только переменные, от которых оно зависит.

Определение 1. Линейной полуалгебраической формулой (ЛПФ) над Q от X называется бесквантерная логическая формула $F(X) = F(E_1(\bar{a}_1, X, c_1), \ldots, E_m(\bar{a}_m, X, c_m))$ от линейных неравенств $E_1(X), \ldots, E_m(X)$ в сигнатуре логических связок $\&, \vee$. Линейным полуалгебраическим множеством называется множество $M(F) = \{V \in Q^n \mid F(V)\}$.

Иными словами, линейная полуалгебраическая формула — это совокупности и системы линейных неравенств, а линейное полуалгебраическое множество — это многогранная область в Q^n, т.е. интерпретация этой формулы.

Задача построения области решения системы линейных неравенств (СЛН) достаточно известна. При размерности $n = 2$ ее можно переформулировать как задачу построения выпуклой линейной оболочки по системе линейных неравенств, задающих стороны многоугольника решения (одна из основных задач вычислительной геометрии и компьютерной графики) [22, 23], а в математических системах учебного назначения — как задачу реализации так называемого графического метода в линейном программировании [24]. Описываемый алгоритм решения СЛН строит каноническую форму этой системы. Кроме того, данный подход к задаче решения СЛН предполагает использование и других алгоритмов: алгоритм сведения кратного интеграла по многогранной области к повторным интегралам, алгоритм изменения порядка интегрирования, алгоритм решения задачи линейного программирования (поиск максимума линейной функции на многогранной области). Основная каноническая форма СЛН представляет многогранник решений такой системы в виде объединения конечного числа трапецоидов. Эта каноническая форма использует идею проектирования (элиминации квантора) методов Фурье–Моцкина [25], Черникова [26].

Определение 2. Пусть x — переменная, входящая в левую часть ЛН, с ненулевым коэффициентом и $Y = X - \{x\}$. Тогда ЛН можно преобразовать к виду

$$x \le LF(\bar{b}, Y) + c \text{ или } x \ge LF(\bar{b}, Y) + c. \tag{1}$$

Такую форму ЛН назовем разрешенной относительно x или x-разрешенной. Если каждое ЛН линейной полуалгебраической формулы, зависящее от x, представлено в разрешенной форме, то ЛПФ будем называть представленной в x-разрешенной форме.

Преобразовав формулу $F(E_1, \ldots, E_m)$ по правилам алгебры высказываний вычисления дизъюнктивной нормальной формы, можно получить ее представление в виде совокупности систем ЛН. Однако такое представление не является единственным и поэтому плохо адаптируется к алгоритмам компьютерной алгебры. Основной результат настоящей статьи — определение канонической формы ЛПФ, обладающей свойством единственности и другими полезными свойствами, а также описание алгоритма построения этой формы. Предлагаемая каноническая форма ЛПФ — прямое обобщение канонической формы представления системы ЛН и алгоритма ее построения, приведенных в [27, 28].

Определение 3. Формулой x-сегмента $s=[L(Y), x, R(Y)]$ над множеством переменных $Y, x \notin Y$, будем называть двойное линейное неравенство вида $L(Y) \le x \le R(Y)$. Если $Y = \{y_1, \ldots, y_m\}$, то формула $s=[L(Y), x, R(Y)]$ интерпретируется на пространстве Q^{m+1}, как элементарная двугранная область $M(s) = \{(q_0, \bar{q}) \in Q^{m+1} \mid L(\bar{q}) \le q_0 \le R(\bar{q})]\}$, которая являеися x-сегментом. Неравенства, представленные в x-разрешенной форме вида $x \le R(Y)$ или $x \ge L(Y)$, можно преобразовать в x-сегменты $s=[-\infty, x, R(Y)]$ или $s=[L(Y), x, +\infty]$.

Пусть $M \subset Q^m$ и $s=[L(Y), x, R(Y)]$. Определим ограничение $s \cdot M$ сегмента s на M следующим образом: $s \cdot M = [L(Y), x, R(Y)] \cdot M \overset{\text{df}}{\approx} (L(Y) \le x \le R(Y)) \& (Y \in M)$, $s \cdot M \ne \varnothing$ тогда и только тогда, когда выполняется условие $\forall \bar{q}\ (\bar{q} \in M) \to \to (L(\bar{q}) \le R(\bar{q}))$, которое запишем в форме $L(x) \le_M R(x)$. Очевидно, что имеет место свойство

$$M \cdot [L, x, R] \cdot M' = M \cdot [L, x, H] \cdot M' + + M \cdot [H, x, R] \cdot M'. \qquad (2)$$

Соотношение (2) определяет на множестве сегментов операцию приведения и обратную ей операцию разбиения множества на части. Точное определение разбиения приводится ниже.

Замечание 1. Отметка ++ конструктора сорта отмечает разбиение выпуклого многогранника на трапецоиды, непересекающиеся либо смежные по граням, меньших, чем n размерностей. Сорт ConPol (Convex Polygon) – сорт выпуклых многогранников в пространстве решений СЛН. В n-мерном пространстве элемент этого сорта (полигон, выпуклый многогранник) задан конструкцией $P = T_1 + + T_2 + + \ldots + + T_k$ [27, 28].

2.1. Системы линейных неравенств и трапецоиды. На переменных вектора $X = \{x_1, \ldots, x_n\}$ определим порядок $x_1 < \ldots < x_n$ и обозначим $X_j \overset{\text{df}}{=} \{x_j, \ldots, x_n\}$. Пусть $s_j = [L_j(X_{j-1}), x_j, R_j(X_{j-1})]$, $j = 1, \ldots, n-1$; $s_n = [L_n, x_n, R_n]$, $L_n, R_n \in Q$.

Определение 4. Множество T, определенное формулой $T = s_1 \cdot s_2 \cdot \ldots \cdot s_n$ (скобки группируются по умолчанию вправо), называется трапецоидом в пространстве Q^n.

Обозначим T_j трапецоид $s_j \cdot s_{j+1} \cdot \ldots \cdot s_n$. Тогда $T = T_1$, $T_j = s_j \cdot T_{j+1}$, $j = 1, 2, \ldots, n-1$, при этом T_{j+1} — проекция T_j на подпространство Q^{n-j+1}, определенное координатами $\{x_{j+1}, \ldots, x_n\}$.

Таким образом, трапецоид T определен последовательностью проекций на убывающую последовательность подпространств $Q^n \supset Q^{n-1} \supset \ldots \supset Q^1$, причем каждая проекция также представляет трапецоид.

Представление трапецоида T в виде последовательности его проекций обозначим формулой $T = T^{(n)} \to T^{(n-1)} \to \ldots \to T^{(1)}$. Выпуклое множество $P \subseteq Q^n$, представляющее решение системы линейных неравенств, называется полигоном.

Определение 5. Разбиением множества $A \subset Q^n$ назовем его представление в виде такого объединения подмножеств $A = A_1 + + A_2 + + \ldots + + A_k$, что $i \ne j \to Dim(A_i \cap A_j) < n$.

Это означает, что различные элементы разбиения пересекаются по множеству, размерность которого меньше размерности пространства, т.е. элементы могут иметь только общую границу. Полигон P может быть представлен в виде разбиения $P = T_1 + + \ldots + + T_m$ на трапецоиды $T_1, \ldots, T_m$. Алгоритм этого представления описан в [28].

Определение 6 . Разбиение $P = T_1 + + \ldots + + T_m$ назовем неприводимым, если к любой паре трапецоидов T_i, T_j нельзя применить операцию приведения (2).

Рассмотрим упорядочение элементов разбиения полигона на трапецоиды. Неприводимое разбиение полигона на трапецоиды единственно с точностью до перестановок трапецоидов разбиения. Упорядочение последовательности можно осуществить в соответствии со следующими его свойствами. Если $s_1^{(n)}, \ldots, s_m^{(n)}$ — последовательность проекций неприводимого разбиения полигона $P = T_1 + + \ldots + + T_m$ на ось Ox_n, то либо $s_i^{(n)} = s_j^{(n)}$, либо $Dim(s_i^{(n)} \cap s_j^{(n)}) = 0$, т.е. проекции $s_i^{(n)}$ и $s_j^{(n)}$ либо совпадают, либо не пересекаются, либо являются смежными. Поэтому $s_1^{(n)}, \ldots, s_m^{(n)}$ могут быть переставлены так, чтобы быть расположенными на оси Ox_n в порядке возрастания. Если $s_{i_1}^{(n)}, \ldots, s_{i_l}^{(n)}$ — попарно различные числовые сегменты, то упорядочение имеет вид

$$s_1^{(n)} = \ldots = s_{i_1}^{(n)} = [L_1, x_n, L_2],\ s_{i_1+1}^{(n)} = \ldots = s_{i_2}^{(n)} = [L_2, x_n, L_3], \ldots$$
$$\ldots,\ s_{i_{l-1}+1}^{(n)} = \ldots = s_m^{(n)} = [L_l, x_n, L_{l+1}],$$

где $L_1 < L_2 < \ldots < L_{l+1}$. Тогда полигон P можно представить в виде

$$P = (T_1 + + \ldots + + T_{i_1}) + + (T_{i_1+1} + + \ldots + + T_{i_2}) + + \ldots + + (T_{i_l+1} + + \ldots + + T_m)$$

или

$$P = (T_1^{(n-1)} + + \ldots + + T_{i_1}^{(n-1)}) \cdot s_{i_1}^{(n)} + + (T_{i_1+1}^{(n-1)} + + \ldots + + T_{i_2}^{(n-1)}) \cdot s_{i_2}^{(n)} + + \ldots$$
$$\ldots + + (T_{i_l+1}^{(n-1)} + + \ldots + + T_m^{(n-1)}) \cdot s_m^{(n)}.$$

Обозначим разбиения трапецоидов в скобках через $P_{i_1}^{(n-1)}, P_{i_2}^{(n-1)}, \ldots, P_m^{(n-1)}$. Тогда $P = P_{i_1}^{(n-1)} \cdot s_{i_1}^{(n)} + + P_{i_2}^{(n-1)} \cdot s_{i_2}^{(n)} + + \ldots + + P_m^{(n-1)} \cdot s_m^{(n)}$.

Свойство разбиения полигона, описанное выше, имеет место для проекций любой размерности. Так, пусть $P = T_1 + + \ldots + + T_l$ — неприводимое разбиение полигона с общей $k+1$-й проекцией $T^{(k+1)}$:

$$T_1 = T_1^{(1)} \to \ldots \to T_1^{(k)} \to T^{(k+1)}, \ldots, T_l = T_l^{(1)} \to \ldots \to T_l^{(k)} \to T^{(k+1)}.$$

Тогда $T_1, \ldots, T_l$ можно упорядочить таким образом, чтобы выполнялось свойство каноничности:

$$s_1^{(k)} = \ldots = s_{i_1}^{(k)} = [L_1, x_k, L_2],\ s_{i_1+1}^{(k)} = \ldots = s_{i_2}^{(k)} = [L_2, x_k, L_3], \ldots$$
$$\ldots,\ s_{i_{l-1}+1}^{(k)} = \ldots = s_m^{(k)} = [L_l, x_k, L_{l+1}], \quad (3)$$
$$L_1 <_{T^{(k+1)}} L_2 <_{T^{(k+1)}} \ldots <_{T^{(k+1)}} L_{l+1}. \quad (4)$$

Неприводимое разбиение полигона на трапецоиды, упорядоченное в соответствии со свойством каноничности, назовем каноническим разбиением или канонической суммой.

Поскольку $T_i^{(j)}$ является проекцией $T_i^{(j+1)}$ на подпространство Q^j, определение 5 означает, что в канонической сумме любые проекции трапецоидов-слагаемых могут пересекаться только по их границе. Соотношение (2) можно назвать операцией приведения подобных.

Алгоритм 1 (пересечения трапецоидов). На множестве трапецоидов в пространстве Q^n естественно определить операцию пересечения $T_1 \cap T_2$. Как показано в [28], это пересечение можно представить в виде канонической суммы трапецои-

дов [29–31]. Рассмотрим алгоритм вычисления пересечения $T_1 \cap T_2 = T_1' + + \ldots + + T_k'$. Представим T_1 и T_2 в виде $T_1 = T_1^{(n-1)} \cdot s_{1n}$, $T_2 = T_2^{(n-1)} \cdot s_{2n}$. Тогда $T_1 \cap T_2$ вычисляется рекурсивно снизу–вверх.

Базис рекурсии.

Имеем

$$s_{1n} \cap s_{2n} = \varnothing \to T_1 \cap T_2 = \varnothing \quad else \quad T_1 \cap T_2 = (T_1^{(n-1)} \cap T_2^{(n-1)}) \cdot (s_{1n} \cap s_{2n}),$$

$$[\max(L_{1n}, L_{2n}) \le_T \min(R_{1n}, R_{2n})] \to s_{1n} \cap s_{2n} =$$

$$= [\max(L_{1n}, L_{2n}), x_1, \min(R_{1n}, R_{2n})] \tag{5}$$

$$else\, s_{1n} \cap s_{2n} = \varnothing.$$

Шаг рекурсии.

Предположим, что трапецоиды $T_1 = T_1^{(n-k)} \cdot T_1^{(k)}$, $T_2 = T_2^{(n-k)} \cdot T_2^{(k)}$ и пересечения $T_1^{(k)} \cap T_2^{(k)}$ уже вычислены, причем $T_1^{(k)} \cap T_2^{(k)} = T_1' + + \ldots + + T_l'$. Тогда ввиду дистрибутивности отметки $++$ относительно отметки $\cdot$ вычисление сводится к нахождению $(T_1^{(n-k)} \cap T_2^{(n-k)}) \cdot T_j'$, которое, в свою очередь, сводится к вычислениям $(s_{1n-k} \cap s_{2n-k}) \cdot T_j'$, $j = 1, \ldots, l$. Результат операции должен удовлетворять соотношению условия в (5).

Поскольку комбинации $\max(L_1, L_2)$, $\min(R_1, R_2)$ не являются линейными, результат операции пересечения сегментов $s_{1n-k} \cap s_{2n-k}$ на T_j' зависит от взаимного расположения этих сегментов на трапецоиде T_j'.

Соотношения $L(Y) \le_T R(Y)$ определим следующим образом: $L(Y) \le_T R(Y)$, если для любого вектора $Y_0 \in T$ имеет место неравенство $L(Y_0) \le R_k(Y_0)$. В канонической форме представления трапецоида T эти соотношения должны выполняться для любого значения $k = n, n-1, \ldots, 1$ на подпространствах Q^k.

Рассмотрим вначале задачу вычисления пересечения $s_1 \cap s_2$ на T в общем виде. Пусть $s_1 = [L_1, x, R_1]$, $s_2 = [L_2, x, R_2]$, $T_1 = s_1 \cdot T$, $T_2 = s_2 \cdot T$, причем $L_1 \le_T R_1$, $L_2 \le_T R_2$. Обозначим

$$LL_{ij} = T \cap (L_i \le_T L_j),\ RR_{ij} = T \cap (R_i \le_T R_j),\ LR_{ij} \overset{\text{df}}{=} L_i \le_T R_j,$$

$$[LR]_{ij} \overset{\text{df}}{=} [L_i, x, R_j],\ i, j = 1, 2, \tag{6}$$

и представим соотношения в виде переписывающих правил для всех вариантов взаимного расположения нижних и верхних шапок T_1, T_2, записывая формулы (5) в терминах (6):

$RR_{12} \to ($ // **варианты 1–6**

1. $LR_{12} \to T_1 \cap T_2 = \varnothing$, **2.** $(L_2 \cap R_1 \ne_T \varnothing) \& LL_{12} \to T_1 \cap T_2 = [LR]_{21} \cdot LR_{21}$,

3. $(L_2 \cap R_1 \ne_T \varnothing) \& (L_2 \cap L_1 \ne_T \varnothing) \to T_1 \cap T_2 = [LR]_{21} \cdot LL_{21} \& LR_{21} + + [LR]_{11} \cdot LL_{21}$,

4. $LR_{21} \& LL_{12} \to T_1 \cap T_2 = [LR]_{21} \cdot T$,

5. $LL_{21} \& (L_1 \cap L_2 \ne_T \varnothing) \to T_1 \cap T_2 = [LR]_{21} \cdot LL_{12} + + [LR]_{11} \cdot LL_{21}$,

6. $LL_{12} \to T_1 \cap T_2 = T_2$);

$(R_2 \cap R_1 \ne_T \varnothing) \& LR_{12} \to ($ // **варианты 7–11**

1. $(L_2 \cap R_1 \ne_T \varnothing) \& LL_{12} \to T_1 \cap T_2 = [LR]_{21} \cdot LR_{21} \& RR_{12} + + [LR]_{22} \cdot RR_{21}$,

2. $(L_2 \cap R_1 \neq_T \varnothing)\&(L_2 \cap_T L_1 \neq \varnothing) \to T_1 \cap T_2 = [LR]_{21} \cdot LL_{12} \,\&\, LR_{21} + +$

$+ + [LR]_{11} \cdot RR_{12} \,\&\, LL_{21} + + [LR]_{12} \cdot RR_{21}$,

3. $LR_{21} \,\&\, LL_{12} \to T_1 \cap T_2 = [LR]_{21} \cdot RR_{21} + + [LR]_{22} \cdot RR_{12}$,

4. $LR_{21} \,\&\, (L_1 \cap_T L_2) \neq \varnothing) \to T_1 \cap T_2 = [LR]_{21} \cdot LL_{12} + + [LR]_{11} \cdot LL_{21} \,\&\, RR_{12} + +$

$+ +\, [LR]_{12} \cdot RR_{21} \,\&\, LL_{21}$,

5. $LL_{21} \to T_1 \cap T_2 = [LR]_{11} \cdot RR_{12} + + [LR]_{12} \cdot RR_{21})$;

$(R_2 \cap R_1 \neq_T \varnothing)\&(R_2 \cap L_1 \neq_T \varnothing) \to ($ // **варианты 12–14**

1. $(R_1 \cap L_2 \neq_T \varnothing)\&(L_1 \cap L_2 \neq_T \varnothing) \to T_1 \cap T_2 = [LR]_{21} \cdot RR_{12} \,\&\, LL_{12} + +$

$+ +\, [LR]_{22} \cdot RR_{12} \,\&\, LL_{21} + + [LR]_{12} \cdot RR_{21} \,\&\, LL_{21}$,

2. $LR_{21} \,\&\, (L_1 \cap L_2 \neq_T \varnothing) \to T_1 \cap T_2 = [LR]_{21} \cdot RR_{12} \,\&\, LL_{12} + + [LR]_{11} \cdot RR_{12} \,\&\, LL_{21} + +$

$+ + [LR]_{12} \cdot RR_{21} \,\&\, LL_{21}$,

3. $LL_{21} \to T_1 \cap T_2 = [LR]_{12} \cdot RR_{21})$;

$RR_{21} \,\&\, LR_{12} \to ($ // **варианты 15–17**

1. $LL_{12} \to T_1 \cap T_2 = T_2$,

2. $L_1 \cap L_2 \neq_T \varnothing \to T_1 \cap T_2 = [LR]_{22} \cdot LL_{12} + + [LR]_{12} \cdot LL_{21}$,

3. $LL_{21} \to T_1 \cap T_2 = [LR]_{12} \cdot T$);

$RR_{21} \,\&\, (L_1 \cap R_2) \neq \varnothing \to ($ // **варианты 18–19**

1. $L_1 \cap L_2 \neq_T \varnothing \to T_1 \cap T_2 = [LR]_{22} \cdot LL_{12} + + [LR]_{12} \cdot LL_{21}$,

2. $LL_{21} \to T_1 \cap T_2 = [LR]_{12} \cdot LR_{21}$);

1. $RR_{21} \to T_1 \cap T_2 = \varnothing$ // **вариант 20**.

Алгоритм 2 (распознавание варианта). Все соотношения, определенные в алгоритме 1, — условные. Условия определены в терминах отношения частичного порядка $\leq_T$ и отношения $\neq_T$. Именно если на области T выполняется линейное неравенство типа $R(X) - L(X) \geq_T 0$, то это означает, что $F_{\max} = \max\limits_{X \in T} (L(X) - R(X)) \leq 0$. Если $R(X) \cap L(X) \neq_T \varnothing$, то это означает, что $\max\limits_{X \in T} (L(X) - R(X)) \geq 0$, $\min\limits_{X \in T} (L(X) - R(X)) \leq 0$.

Поскольку функция $F = L(X) - R(X)$ линейна и трапецоид T — выпуклая многогранная область, то из этого следует, что максимум и минимум достигаются в одной из вершин T. Форма представления трапецоида в виде последовательности его проекций на подпространства с уменьшающейся размерностью позволяет вычислить искомый максимум за время $O(n^2)$.

Алгоритм 3 (пересечение полигонов). Операция пересечения полигонов $P_1 \cap P_2$ описывается системой соотношений — переписывающих правил. Пусть

$$P_1 = T_{11} + + T_{12} + + \ldots + + T_{1k_1},\ P_2 = T_{21} + + T_{22} + + \ldots + + T_{2k_2}.$$

Представим P_1 и P_2 в виде $P_1 = g + G$, $P_2 = h + H$, где $g = T_{11}$, $h = T_{21}$, $G = T_{12} + + \ldots + + T_{1k_1}$, $H = T_{22} + + \ldots + + T_{2k_2}$.

Основное правило имеет вид

$$(g + + G)(h + + H) = gh + + (gH \cup hG) + + HG. \qquad (7)$$

Пусть $g = T' \cdot [l_g, r_g]$, $h = T'' \cdot [l_h, r_h]$. Тогда при $r_g \leq r_h$ имеем $gH = \varnothing$, а при $r_h \leq r_g$ имеем $hG = \varnothing$. Это позволяет упростить правило (7). Введем следующие функции доступа: при $T = T' \cdot s$ и $Base(T) \stackrel{\text{df}}{=} s$ для $s = [L, x, R]$ имеем $LP(s) = L$, $RP(s) = R$. Тогда (6) упрощается:

$$RP(Base(g)) \le RP(Base(h)) \to (g++G)(h++H) = gh++hG++HG$$
$$else\,(g++G)(h++H) = gh++gH++HG. \quad (8)$$

Производные правила выводятся из (7) для частных случаев $G = \varnothing$ или $H = \varnothing$:

$$RP(Base(g)) \le RP(Base(g)) \to g(h++H) = gh \;\; else\, g(h++H) = gh++gH, \quad (9)$$

$$RP(Base(g)) \le RP(Base(h)) \to (g++G)h = gh++hG \;\; else\,(g++G)h = gh. \quad (10)$$

Система переписываний (8)–(10) представляет алгоритм пересечения полигонов.

2.2. Линейные полуалгебраические формулы и политрапецоиды.

Определение 6. Пусть $Y = (y_1, \ldots, y_m)$, $x \notin Y$, $T \subseteq Q^m$. Алгеброй x-полисегментов (полисегментов) над основанием трапецоида T назовем конструктивное расширение алгебры x-сегментов, элементы носителя которой имеют вид

$$S \cdot T = ([L_1, x, R_1]++[L_2, x, R_2]++\ldots++[L_k, x, R_k])\cdot T,$$
$$L_i = L_i(Y), R_i = R_i(Y), \quad (11)$$

при этом выполняются следующие условия:

$$1)\; L_1 \le_T R_1 \le_T L_2 \le_T R_2 \le_T \ldots \le_T L_k \le_T R_k,\; R_i \ne L_{i+1}; \quad (12)$$

2) если для некоторого i имеем $R_i \equiv L_{i+1}$, то запись (5) сокращается в соответствии с (2): $[L_i, x, R_i]++[L_{i+1}, x, R_{i+1}] = [L_i, x, R_{i+1}]$.

На алгебре полисегментов определены теоретико-множественные операции пересечения и объединения. Очевидно, что

$$([L_1, x, R_1]++\ldots++[L_k, x, R_k])\cdot T = [L_1, x, R_1]\cdot T \cup \ldots \cup [L_k, x, R_k]\cdot T.$$

Определение 7. Пусть $X_j = (x_1, \ldots, x_j)$, $j = 1,2,\ldots,n$. Определим последовательность S_j рекурсивно:

1) $s_{1i} = [a_i, x_1, b_i]$, a_j, $b_j \in Q$, $i = 1, \ldots, k_1$; $S_1 = s_{11}++\ldots++s_{1k_1}$ — базовый x_1-полисегмент, $T_1 \overset{\text{df}}{=} S_1$;

2) S_{j+1} для любого $j = 1,\ldots,n-1$ является x_{j+1}-полисегментом над T_j, а $T_{j+1} \overset{\text{df}}{=} S_{j+1}\cdot T_j$.

Множество T_n (или просто T) назовем политрапецоидом в пространстве Q^n. Очевидно, что $T = S_n \cdot S_{n-1} \cdot \ldots \cdot S_1$, причем T_j — проекция T_{j+1} на подпространство Q^j, определенное координатами $x_1, \ldots, x_j$.

Политрапецоид T определен последовательностью проекций на убывающую последовательность подпространств $Q^n \supset Q^{n-1} \supset \ldots \supset Q^1$, причем каждая проекция также является политрапецоидом.

Алгоритм 4 (пересечение политрапецоидов). Пусть PT_1, PT_2 — политрапецоиды и $P = PT_1 \cap PT_2$. Очевидно, P можно представить в виде канонической суммы политрапецоидов. Алгоритм пересечения политрапецоидов вычисляет P рекурсивно снизу–вверх.

Базис рекурсии. Пусть $PT_1 = S_{11}\cdot\ldots\cdot S_{1n}$, $PT_2 = S_{21}\cdot\ldots\cdot S_{2n}$. Тогда

$$PT_1 \cap PT_2 = (PT_1^{(n-1)} \cap PT_2^{(n-1)})\cdot(S_{1n} \cap S_{2n}). \quad (13)$$

Соотношение (13) согласно (12) определяет базис рекурсии — алгоритм вычисления $S_{1n} \cap S_{2n}$, который определяется переписывающими правилами (8)–(10) и представляет собой сумму сегментов числовой оси Ox_x, удовлетворяющую (12).

Шаг рекурсии. Пусть $PT_1 = S_{11} \cdot \ldots \cdot S_{1n-k} \cdot ST^{(k)}$, $PT_2 = S_{21} \cdot \ldots \cdot S_{2n-k} \cdot ST^{(k)}$ для некоторого k, где $ST^{(k)}$ — сумма политрапецоидов, представляющая $(S_{1n-k} \cdot \ldots \cdot S_{1n}) \cap (S_{2n-k} \cdot \ldots \cdot S_{2n})$, т.е. $ST^{(k)} = (S_{1n-k} \cdot \ldots \cdot S_{1n}) \cap (S_{2n-k} \cdot \ldots \cdot S_{2n})$, а также

$$PT_1 \cap PT_2 = ((S_{11} \cdot \ldots \cdot S_{1n-k}) \cap (S_{21} \cdot \ldots \cdot S_{2n-k})) \cdot ST^{(k)}. \qquad (14)$$

Тогда (14) перепишем в виде

$$PT_1 \cap PT_2 = ((S_{11} \cdot \ldots \cdot S_{1n-k-1}) \cap (S_{21} \cdot \ldots \cdot S_{2n-k-1}))(S_{1n-k} \cap S_{2n-k}) \cdot ST^{(k)},$$

сведя задачу к шагу рекурсии — вычислению $(S_{1n-k} \cap S_{2n-k}) \cdot ST^{(k)}$.

Поскольку $ST^{(k)}$ представляет собой сумму политрапецоидов, то

$$ST^{(k)} = PT_1^{(k)} + + \ldots + + PT_l^{(k)},$$

$$(S_{1n-k} \cap S_{2n-k}) \cdot ST^{(k)} = (S_{1n-k} \cap S_{2n-k}) \cdot PT_1^{(k)} + + \ldots + + (S_{1n-k} \cap S_{2n-k}) \cdot PT_l^{(k)}.$$

Для вычисления $(S_{1n-k} \cap S_{2n-k}) \cdot PT_j^{(k)}$ применяются алгоритмы 1–3.

Алгоритм 5 (объединение политрапецоидов). Пусть PT_1 и PT_2 — политрапецоиды, $P = PT_1 \cup PT_2$ и P можно представить в виде канонической суммы политрапецоидов. Пусть также $PT_1 = PT_1^{(n-1)} \cdot S_{1n}$, $PT_2 = PT_2^{(n-1)} \cdot S_{2n}$. Положим $S_{12n} = S_{1n} \cap S_{2n}$, $S'_{1n} = S_{1n} - S_{12n}$, $S'_{2n} = S_{2n} - S_{12n}$. Тогда $S_{1n} = S'_{1n} + + S_{12n}$, $S_{2n} = S'_{2n} + + S_{12n}$, а также

$$PT_1 \cup PT_2 = Ord(PT_1^{(n-1)} \cdot S'_{1n} + + PT_2^{(n-1)} \cdot S'_{2n} + + (PT_1^{(n-1)} \cup PT_2^{(n-1)}) \cdot S_{12n}, \qquad (15)$$

где функция $Ord()$ упорядочивает слагаемые. Алгоритм нахождения S'_{1n}, S'_{2n}, S_{12n} в (15) вычисляет S_{12n}, используя алгоритмы 1–3, и рекурсивно вычисляет $PT_1^{(n-1)} \cup PT_2^{(n-1)}$. Алгоритм вычисления $S'_{1n} = S_{1n} - S_{12n}$, $S'_{2n} = S_{2n} - S_{12n}$ аналогичен алгоритму (5).

Таким образом, алгебра линейных полуалгебраических множеств построена в виде цепочки расширений алгебр трапецоидов, полигонов и политрапецоидов перегрузкой теоретико-множественных операций $\&, \vee$.

ЗАКЛЮЧЕНИЕ

Как показывает анализ существующих систем построения доказательств теорем, разработка системы, которая позволяла бы решить поставленые авторами задачи, остается актуальной. Примеры серьезных ошибок работы сложных информационных систем представлены в [32–34]. Соответственно актуальной является и необходимость реализации соответствующих алгоритмов, позволяющих выполнять упрощение формул на более эффективном уровне. Основной результат настоящей статьи — определение канонической формы линейной полуалгебраической формулы, обладающей свойством единственности и другими полезными свойствами, а также описание алгоритма ее построения. В дальнейшем планируется описание реализации предложенного алгоритма для построения канонических форм логических формул над типами данных методом инсерционного моделирования.

ПРИЛОЖЕНИЕ 1

formula (f1)

```
Exist x3 ((1 = -1 * y + x)&(0 <= R(x3)&
R(x3) <= 3|/9 <= R(x3)&
R(x3) <= 18)&(11 <= F(x3)&
F(x3) <= 20|/2 <= F(x3)&
F(x3) <= 5)&-1 * x3 + y <= (-1))
```

formula (f2)

```
Exist x3 ((1 = -1 * y + x)&(0 <= R(x3)&
R(x3) <= 3|/9 <= R(x3)&
R(x3) <= 18)&(11 <= F(x3)&
F(x3) <= 20|/2 <= F(x3)&
F(x3) <= 5)&-1 * x3 + y <= (-1)&
-1 * x3 + y <= (-2)|/(1 = -1 * y + x)&
(-1 = -1 * x3 + y)&(7 <= R(x3)&
R(x3) <= 16|/-2 <= R(x3)&
R(x3) <= 1)&(13 <= F(x3)&
F(x3) <= 22|/4 <= F(x3)&
F(x3) <= 7))
```

formula (f3)

```
Exist x3 ((1 = -1 * y + x)&(0 <= R(x3)&
R(x3) <= 3|/9 <= R(x3)&
R(x3) <= 18)&(11 <= F(x3)&
F(x3) <= 20|/2 <= F(x3)&
F(x3) <= 5)&-1 * x3 + y <= (-1)&
-1 * x3 + y <= (-2)|/(1 = -1 * y + x)&
(-1 = -1 * x3 + y)&(5 <= R(x3)&
R(x3) <= 14|/-4 <= R(x3)&
R(x3) <= (-1))&(6 <= F(x3)&
F(x3) <= 9|/15 <= F(x3)&
F(x3) <= 24))
```

formula (f4)

```
Exist x3 ((1 = -1 * y + x)&
(0 <= R(x3)&R(x3) <= 3|/
9 <= R(x3)&R(x3) <= 18)&
(11 <= F(x3)&F(x3) <= 20|/
2 <= F(x3)&F(x3) <= 5)&
-1 * x3 + y <= (-1)&-1 * x3 + y <= (-2)|/
(1 = -1 * y + x)&(-1 = -1 * x3 + y)&
(1 <= R(x3)&R(x3) <= 10|/
-8 <= R(x3)&R(x3) <= (-5))&
(10 <= F(x3)&F(x3) <= 13|/
19 <= F(x3)&F(x3) <= 28)
 )
```

formula (f5)

```
Exist (x0,x2,x4)(
F(x) <= 4&1 <= F(x)&
(F(x4) = 4)&(F(x2) = 4)&
(F(x0) = 4))
```

formula (f6)

```
Exist x6(F(x) <= 5&
2 <= F(x)&(-1 * x + x6 <= (-1)|/
-1 * x6 + x <= (-1))&(F(x6) = 4))
```

formula (f7)

```
Exist (x!11 : int) (Forall (x!9 : int) (
(~(x!11 + -1 * x!9 <= (-1))&~(-1 * x!11 + x!9 <= (-
1))|/
(Mesi(x!11) = E)&(Mesi(x!9) = I))&
(~(x!11 + -1 * x!9 <= 0)|/~(x!9 + -1 * x!11 <=
0)|/
(Mesi(x!11) = E))))
```

formula (f8)

```
~(x + -1 * y <= (-4))&~(y + -1 * x <= (-4))&
~(-1 * y + -1 * x <= (-4))&~(x + y <= (-4))&
~(x + -1 * y <= (-2))&~(y + -1 * x <= (-2))&
~(-1 * y + -1 * x <= (-2))&~(x + y <= (-2))|/
~(x + -1 * y <= 8)&~(y + -1 * x <= (-16))&
~(-1 * y + -1 * x <= (-16))&~(x + y <= 8)&
~(x + -1 * y <= (-2))&~(y + -1 * x <= (-2))&
~(-1 * y + -1 * x <= (-2))&~(x + y <= (-2))
```

formula (f9)

```
~(x + -1 * y <= (-4))&~(y + -1 * x <= (-4))&
~(-1 * y + -1 * x <= (-4))&~(x + y <= (-4))&
~(x + -1 * y <= 0)&~(y + -1 * x <= (-8))&
~(-1 * y + -1 * x <= (-8))&~(x + y <= 0)|/
~(x + -1 * y <= 8)&~(y + -1 * x <= (-16))&
~(-1 * y + -1 * x <= (-16))&~(x + y <= 8)&
~(x + -1 * y <= 0)&~(y + -1 * x <= (-8))&
~(-1 * y + -1 * x <= (-8))&~(x + y <= 0)
```

formula (f10)

```
Exist v ((v=1)&(x=v))
```

СПИСОК ЛИТЕРАТУРЫ

1. CVC4. URL: http://cvc4.cs.stanford.edu/web/.
2. Cimatti A., Griggio A., Schaafsma B.J., Sebastiani R. The mathSAT5 SMT solver. *Tools and Algorithms for the Construction and Analysis of of Systems*: 19th Intern. Conf. TACAS 2013. Berlin; Heidelberg: Springer, 2013. P. 93–107.
3. QEPCAD. URL: https://www.usna.edu/CS/qepcadweb/B/WhatisQEPCAD.html.
4. Singular. URL: https://www.singular.uni-kl.de/.
5. COCOA. URL: http://cocoa.dima.unige.it/.
6. MiniZinc. URL: http://www.minizinc.org/.
7. STP constraint solver. URL: http://stp.github.io/.
8. Redlog — Computing with logic. URL: http://www.redlog.eu/.
9. Satallax. URL: http://www.ps.uni-saarland.de/~cebrown/satallax/.
10. Isabelle. URL: https://isabelle.in.tum.de/.
11. E-SETHEO. URL: http://www4.informatik.tu-muenchen.de/~schulz/WORK/e-setheo.html.
12. MiniSat. URL: http://minisat.se/.
13. SMTInterpol an interpolating SMT solver. URL: http://ultimate.informatik.uni-freiburg.de/ smtinterpol/.
14. Paradox (theorem-prover). URL: https://wikivisually.com/wiki/Paradox_%28theorem_prover%29.
15. Gandalf. URL: http://deepthought.ttu.ee/it/gandalf/.
16. Vampire. URL: http://www.vprover.org/download.cgi.
17. QEPCAD — The solution formula construction phase. URL: https://www.usna.edu/CS/qepcadweb/B/user/Solution.html.
18. The E theorem prover. URL: http://wwwlehre.dhbw-stuttgart.de/~sschulz/E/E.html.
19. iProver. URL: http://www.cs.man.ac.uk/~korovink/iprover/.
20. LEO-II. URL: http://page.mi.fu-berlin.de/cbenzmueller/leo/.
21. Z3Prover. URL: https://github.com/Z3Prover/z3.
22. Препарата Ф., Шеймос М. Вычислительная геометрия. Введение. Москва: Мир, 1989. 478 с.
23. Ласло М. Вычислительная геометрия и компьютерная графика на С++. Пер. с англ. Москва: Бином, 1997. 304 с.
24. Солодовников А.С. Системы линейных неравенств. Москва: Наука, 1997. 112 с.
25. Моцкин Т.С., Райфа Х., Томпсон Дж.Л., Тролл Р.М. Метод двойного описания. *Матричные игры.* Москва: Физматгиз, 1961. С. 81–109.

26. Черников С.Н. Линейные неравенства. Москва: Наука, 1968. 490 с.
27. Львов М.С. Алгебраический подход к задаче решения систем линейных неравенств. *Кибернетика и системный анализ*. 2010. № 2. С. 175–188.
28. Львов М.С., Песчаненко В.С. Метод трапецоидов решения систем линейных неравенств и его реализация инсерционным моделированием. *Кибернетика и системный анализ*. 2012. № 6. С. 144–156.
29. Goguen J., Meseguer J. Ordered-sorted algebra I: Partial and overloaded operations. Errors and inheritance. SRI international. *Computer Science Lab.* 1987. Vol. 105, N 2. P. 217–273.
30. Goguen J.A., Thatcher J.W., Wagner E.G. An initial algebra approach to the specification, correctness and implementation of abstract data types — IBM Res. *Rep. RC6487, Yorktown Heights*. 1976. Vol. 156. P. 80–149.
31. Львов М.С. Синтез інтерпретаторів алгебраїчних операцій в розширеннях багатосортних алгебр. *Вісник Харьківського національного університету*. 2009. № 847. С. 221–238.
32. The economic impacts of inadequate infrastructure for software testing. NIST Report. May 2002 URL: http://www.nist.gov/director/prog-ofc/report02-3.pdf.
33. Sagdeev R.Z., Zakharov A.V. Brief history of the Phobos mission. *Nature*. 1989. Vol. 341, N 6243. P. 581–585.
34. Lewis G.N., Fetter S., Gronlund L. Casualties and damage from scud attacks in the 1991 Gulf war. Defense and Arms Control Studies Program. Cambridge, MA: Massachusetts Institute of Technology, 1993. 51 p.

Надійшла до редакції 26.12.2017

М.С. Львов, В.С. Песчаненко, О.О. Летичевський, Ю.Г. Тарасіч, А.С. Баєв
АЛГОРИТМ ТА ІНСТРУМЕНТИ ПОБУДОВИ КАНОНІЧНИХ ФОРМ ЛІНІЙНИХ НАПІВАЛГЕБРАЇЧНИХ ФОРМУЛ

Анотація. Відображено результати випробувань інструментів спрощення формул, а також описано алгоритм побудови канонічних форм лінійних напівалгебраїчних формул (ЛНФ). Основним результатом цієї роботи є визначення канонічної форми ЛНФ з властивістю унікальності та іншими корисними властивостями, а також опис алгоритму її побудови.

Ключові слова: системи лінійних нерівностей, алгебраїчне програмування, верифікація програмного забезпечення, канонічні форми, лінійні напівалгебраїчні формули.

M. Lvov, V. Peschanenko, O. Letychevskyi, Y. Tarasich, A. Baiev
ALGORITHM AND TOOLS FOR CONSTRUCTING CANONICAL FORMS OF LINEAR SEMI-ALGEBRAIC FORMULAS

Abstract. The results of tests of formula simplification tools are presented in the first part of the paper. In the second part, the algorithm for constructing canonical forms of linear semi-algebraic formulas is described. The main result of the study is the definition of the canonical form of linear semi-algebraic formula, which has the property of uniqueness and other useful properties. The algorithm of its construction is described.

Keywords: systems of linear inequalities, algebraic programming, software verification, canonical forms, linear semialgebraic formulas.

Львов Михаил Сергеевич,
доктор физ.-мат. наук, профессор, заведующий кафедрой Херсонского государственного университета, e-mail: lvov@ksu.ks.ua.

Песчаненко Владимир Сергеевич,
доктор физ.-мат. наук, доцент, профессор кафедры Херсонского государственного университета, e-mail: vladim@ksu.ks.ua.

Летичевский Александр Александрович,
доктор физ.-мат. наук, старший научный сотрудник Института кибернетики им. В.М. Глушкова НАН Украины, Киев, e-mail: lit@iss.org.ua.

Тарасич Юлия Геннадиевна,
аспирантка Херсонского государственного университета, e-mail: yutarasich@ksu.ks.ua, yutarasich@gmail.com.

Баев Андрей Святославович,
студент Херсонского государственного университета, e-mail: andreybayev95@gmail.com.