УДК 004.02

ЗАДАЧА ПРОВЕРКИ Т-ВЫПОЛНИМОСТИ ДЛЯ ЛОГИЧЕСКОГО ЯЗЫКА VL1 СИСТЕМЫ VRS

В.Г. Тимофеев

Киевский национальный университет имени Тараса Шевченко,
кафедра теории и технологии программирования,
01601 Киев, Украина, ул. Владимирская 64, e-mail: tvalentyn@univ.kiev.ua

В данной работе дается короткое введение в задачу проверки Т-выполнимости формул относительно логических теорий, и показывается, что разработанные методы решения этой задачи могут применяться в технологии инсерционного моделирования, которая представлена в системе верификации требований VRS. Дается формализация логического языка, используемого в VRS для проведения формальных рассуждений, и показывается разрешимость проблемы выполнимости формул в этом языке. Обсуждаются особенности применения используемых методов, и описывается альтернативный алгоритм поиска выполнимой конъюнкции, основанный на множественном представлении операций в формулах.

In this paper we give a short introduction to the satisfiability modulo theories (SMT) problem and demonstrate how the methods developed in the SMT research field can be applied in the requirement verification tool VRS, which supports insertion modelling methodology. We formalize the logical language VL1 used for formal reasoning in VRS and justify decidability of satisfiability problem in that language. Besides we discuss the context of the problem being solved and indicate possible methods that can be used on different stages of solution. Finally, we present a satisfiable conjunction search method, which is based on a set-representation of logical connectives in formulas.

Введение

Задача проверки выполнимости логических формул является одной из фундаментальных и прикладных задач в математической логике. В терминах проверки выполнимости могут быть сформулированы некоторые задачи верификации программных и аппаратных систем, задачи теории графов, теории расписаний, статического анализа, автоматической генерации тестов и др. Для этого может быть достаточно даже простейшего языка логики высказываний. Однако применения формальных методов в разработке программного и аппаратного обеспечения, и задачах искусственного интеллекта требуют выразительных возможностей, которые уже выходят за рамки пропозициональной логики. Необходимость решения задачи выполнимости для более выразительных фрагментов логик в различных предметных областях привело к появлению задачи *T*-выполнимости, ее активному изучению и значительному прорыву в решении с точки зрения эффективности инструментальных средств и выразительных возможностей поддерживаемых языков. Об этом свидетельствуют ежегодные конференции, симпозиумы и соревнования программных комплексов в области исследований, которая получила название Satisfiability Modulo Theories (SMT) [1 – 3]. Название отражает главную особенность рассматриваемых проблем: приложения формальных методов требуют решения задачи выполнимости в логике первого порядка относительно одной или нескольких теорий *T*, ограничивающих допустимые интерпретации функциональных и предикатных символов. Например, говоря о выполнимости формулы линейной арифметики $x + 0 = y \to x + 1 \geq y + 1$ мы заинтересованы только в стандартных интерпретациях символов 0, 1, + и $\geq$ на множестве действительных чисел.

Актуальность и практическая потребность в решении задачи SMT обосновывается и в данной работе, где эта задача ставится в контексте технологии инсерционного моделирования, которая разрабатывается в Институте кибернетики им. В.М. Глушкова НАН Украины.

Инсерционное моделирование [4 – 6] – это технология проектирования систем, основанная на теории взаимодействий агентов и сред. Этот подход был успешно применен к задачам верификации спецификаций для распределенных систем реального времени в различных предметных областях [7, 8]. Одно из главных применений технологии инсерционного моделирования реализуется в системе VRS (Verification of Requirement Specifications) [5, 8]. Эта система предоставляет возможность проверки требований путем автоматического доказательства теорем, использованием техник символьной и дедуктивной проверки моделей, а также генерацией трасс для тестирования с различными критериями покрытия. Основные инструменты системы VRS разделены на две группы: статические и динамические. Статические инструменты включают проверку непротиворечивости и полноты спецификаций. Эти инструменты используют специальную процедуру-решатель (*solver*), которая обеспечивает проверку выполнимости формул в логическом языке, фрагменте логики первого порядка. В данной работе мы показываем, что задача, которая ставится перед VRS-решателем, является типичным примером задачи SMT. Это свидетельствует о том, что достижения в этой области могут быть использованы для решения задач инсерционного моделирования. Для этого в статье формулируется проблема проверки выполнимости формулы относительно теории (satisfiability modulo theory). Затем строится формализация логического языка VL1, используемого решателем VRS. Далее показывается разрешимость проблемы выполнимости в этом языке. Также приводится короткий обзор основных подходов, вариации которых используются в современных SMT-решателях и показывается, как они могут быть применяться для

решения проблемы выполнимости в языке VL1. Из достаточно большого арсенала средств разработанных в области SMT мы указываем конкретные методы и оценки сложности используемых алгоритмов. Наконец, описывается альтернативный эвристический метод поиска выполнимой конъюнкции и приводятся данные результатов экспериментов, демонстрирующие его применимость в контексте системы VRS. Этот метод, насколько нам известно, не исследовался в SMT-литературе.

Статья структурирована следующим образом. В разделе 1 дается необходимый логический формализм и описание языка VL1. В разделе 2 формулируется задача выполнимости относительно теории. Основные методы решения этой задачи, а также особенности их применения в контексте системы VRS, обсуждаются в разделе 3. В разделе 4 описывается метод поиска выполнимой конъюнкции, основанный на представлении формул с множественными операциями.

1. Логический язык VL1

Идеология инсерционного моделирования, представленная в системе VRS, предполагает, что исходные данные для анализа системы представлены в виде спецификации на языке базовых протоколов [9]. Каждая такая спецификация состоит из двух частей: описание среды и множество базовых протоколов. Описание среды доопределяет сигнатуру базового языка и возможные ограничения интерпретации этой сигнатуры. Множество базовых протоколов определяет требования к поведению системы. Во время процесса верификации, система VRS использует специализированный решатель как средство для проверки выполнимости формулы-свойства, сформулированной в специальном базовом языке. В частности, это требуется на этапе проверки условия применимости базового протокола в состоянии. На низком уровне это может быть реализовано путем проверки выполнимости формул в некотором логическом языке первого порядка. Мы делаем формализацию этого языка в виде языка VL1.

Анализ практических требований, которые ставятся к решателю системы VRS, показывает, что язык VL1 можно задать в терминах языка многосортной логики предикатов первого порядка.

Язык многосортной логики (ЯМЛ) *предикатов первого порядка* определяется своей сигнатурой. *Сигнатура языка* задается четверкой $\Sigma = (\Sigma^S, \Sigma^C, \Sigma^F, \Sigma^P)$, где Σ^S – непустое конечное множество сортов, Σ^C, Σ^F, Σ^P – счетные множества констант, функциональных и предикатных символов соответственно. Каждый символ константы ассоциируется с некоторым сортом σ, каждый функциональный символ ассоциируется с его *типом* – объектом вида $\sigma_1 \times \cdots \times \sigma_n \to \sigma$, каждый предикатный символ ассоциируется с типом вида $\sigma_1 \times \cdots \times \sigma_n$; здесь $\sigma_i, \sigma \in S,\ 1 \le i \le n$.

Пусть *Var* – счетное множество переменных. Каждая переменная ассоциируется с некоторым сортом σ. Для каждого из сортов в множестве *Var* имеется бесконечное число переменных этого сорта.

Для данной сигнатуры Σ множества Σ-термов, Σ-атомов и Σ-формул определяются индуктивно. Каждая переменная x сорта σ есть Σ-терм сорта σ. Каждая константа c сорта σ, $c \in \Sigma^C$, есть Σ-терм сорта σ. Если $f \in \Sigma^F$ – функциональный символ типа $\sigma_1 \times \cdots \times \sigma_n \to \sigma$, $t_1, ..., t_n$ – Σ-термы сортов $\sigma_1, ..., \sigma_n$ соответственно, то $f(t_1, ..., t_n)$ – терм сорта σ.

Множество Σ-*атомов* состоит из выражений $p(t_1, ..., t_n)$ где $p \in \Sigma^P$ – предикатный символ типа $\sigma_1 \times \cdots \times \sigma_n$, $t_1, ..., t_n$ – Σ-термы сортов $\sigma_1, ..., \sigma_n$ соответственно.

Каждый Σ-атом является Σ-*формулой*. Формулы полученные применением к Σ-формулам пропозициональных связок $(\neg, \vee, \wedge, \to, \leftrightarrow)$ являются Σ-формулами. Если φ – Σ-формула, x – переменная сорта σ, то $\forall_\sigma x\, \varphi$ и $\exists_\sigma x\, \varphi$ – Σ-формулы. Приоритеты операций традиционные, для точного указания порядка применения операций могут использоваться скобки.

Формула ψ называется Σ-*литералом*, если $\psi = \varphi$ или $\psi = \neg\varphi$ для некоторого атома φ.

Множество переменных сорта σ, встречающихся в терме t, обозначается $vars_\sigma(t)$, множество всех переменных, встречающихся в терме t, обозначается $vars(t)$. Для формулы φ обозначим $vars_\sigma(\varphi)$ множество свободных (не связанных кванторами) переменных сорта σ. Множество всех свободных переменных в формуле обозначим $vars(\varphi)$. Если $vars(\varphi) = \varnothing$, то φ считается *замкнутой* формулой.

Семантика ЯМЛ сигнатуры Σ задается с помощью понятия Σ-интерпретации. Пусть Σ – сигнатура языка, X – множество переменных, сорта которых принадлежат Σ^S. Σ-*интерпретация I над X* – это отображение, которое ставит в соответствие каждому сорту $\sigma \in \Sigma^S$ непустое множество I_σ (*область значений*, *носитель* сорта), каждой переменной $x \in X$ сорта σ – элемент $x^I \in I_\sigma$, каждому символу константы $c \in \Sigma^C$ сорта σ – элемент $c^I \in I_\sigma$, каждому функциональному символу $f \in \Sigma^F$ типа $\sigma_1 \times \cdots \times \sigma_n \to \sigma$ – тотальную функцию $f^I : I_{\sigma_1} \times \cdots \times I_{\sigma_n} \to I_\sigma$, каждому предикатному символу $f \in \Sigma^P$ типа $\sigma_1 \times \cdots \times \sigma_n$ – некоторое подмножество $p^I : p^I \subseteq I_{\sigma_1} \times \cdots \times I_{\sigma_n}$. В случае, когда множество X ясно из контекста либо пусто, будем говорить просто о Σ-*интерпретации*.

Если I – Σ-интерпретация над X, t – Σ-терм такой, что $vars(t) \subseteq X$, то можно определить понятие *истинностной оценки* терма t *в интерпретации* I (обозначается $[t]^I$). Для переменной x полагаем $[x]^I = x^I$, для константы c полагаем $[c]^I = c^I$, для $f(t_1,...,t_n)$ полагаем $[f(t_1,...,t_n)]^I = f^I([t_1]^I,\ldots,[t_n]^I)$.

Пусть I, J – Σ-интерпретации над X, $x \in X$. Будем говорить, что J *x-варианта* I, если $I_\sigma = J_\sigma$ для всех $\sigma \in \Sigma^S$ и $r^I = r^J$ для всех объектов $r \in \Sigma^C \cup \Sigma^F \cup \Sigma^P \cup (X \setminus \{x\})$.

Будем считать, что множество истинностных значений состоит из двух элементов {*false*, *true*}. Если I – Σ-интерпретация над X, φ – Σ-формула такая, что $vars(\varphi) \subseteq X$, то можно говорить об *истинностной оценке формулы* φ *в интерпретации* I (обозначается $[\varphi]^I$). Для предикатного символа p, и термов $t_1,\ldots, t_n$ $[p(t_1,\ldots,t_n)]^I = true$ тогда и только тогда, когда $([t_1]^I,\ldots,[t_n]^I) \in p^I$. $[\forall_\sigma x\varphi]^I = true$ тогда и только тогда, когда $[\varphi]^J = true$ для любой x-варианты J интерпретации I; $[\exists_\sigma x\varphi]^I = true$ тогда и только тогда, когда $[\varphi]^J = true$ для некоторой x-варианты J интерпретации I; $[\neg\varphi]^I = true$ тогда и только тогда, когда $[\varphi]^I = false$. Истинностная оценка формулы, построенной с помощью бинарных пропозициональных связок $(\vee, \wedge, \rightarrow, \leftrightarrow)$, определяется естественным образом.

Пусть I – Σ-интерпретация над X, φ – Σ-формула такая, что $vars(\varphi) \subseteq X$. Будем писать $I \models \varphi$ для обозначения того, что $[\varphi]^I = true$. Будем говорить в таком случае, что φ истинна в интерпретации I, или что I – модель формулы φ.

Пусть φ – Σ-формула, $X = vars(\varphi)$. Формула φ *всюду истинна*, если $I \models \varphi$ для любой Σ- интерпретации I над X. Это будем обозначать $\models\varphi$. Формула φ *выполнима*, если существует Σ-интерпретация I над X такая, что $I \models \varphi$. Это будем обозначать $\models\!\approx\varphi$.

Пусть φ – Σ-формула, $X = vars(\varphi)$, M – множество Σ-*интерпретаций* над X. Будем говорить, что φ M-*истинна*, если $I \models \varphi$ для всех Σ-интерпретаций $I \in M$. Это будем обозначать $\models_M \varphi$. Будем говорить, что φ M-*выполнима*, если существует Σ-интерпретация $I \in M$ такая, что $I \models \varphi$. Это будем обозначать $\models\!\approx_M \varphi$. Формула M-*невыполнима* если она не M-выполнима.

Пусть S – множество Σ-формул. Будем писать $I \models S$, если $I \models \varphi$ для каждой формулы $\varphi \in S$.

Проблема выполнимости для ЯМЛ сигнатуры Σ состоит в определении выполняется ли $\models\!\approx\varphi$ для произвольной Σ-формулы φ.

Когда сигнатура языка фиксирована, символ Σ будем опускать, говоря просто о термах, формулах, интерпретациях и т. д.

Теперь в терминах ЯМЛ можно определить язык VL1 и рассматриваемые интерпретации этого языка. Язык VL1 используется для проведения рассуждений о формулах, выражающих свойства в линейной арифметике с равенством, линейным порядком, неинтерпретированными символами: неинтерпретированными константами и неинтерпретированными функциональными символами. Линейная арифметика в VL1 должна поддерживать переменные над целыми и рациональными числами.

Язык VL1 является подмножеством ЯМЛ сигнатуры $\Sigma = (\Sigma^S, \Sigma^C, \Sigma^F, \Sigma^P)$, где Σ^S содержит три сорта Σ^S = <*int*, *rat*, *symb*>. Множество констант $\Sigma^C = C_{rat} \cup C_{symb}$. Множество констант C_{rat} используется для задания целых и рациональных коэффициентов в формулах VL1. Считаем, что C_{rat} состоит из всех рациональных чисел, представленных в виде несократимых дробей. Множество констант C_{symb} состоит из всех идентификаторов, т. е., последовательностей из букв латинского алфавита и цифр, которые начинаются с буквы. Символы констант можно считать неинтерпретированными символами. Переменные сорта *int* называем целыми переменными, переменные сорта *rat* называем рациональными переменными, переменные сорта *symb* называем символьными переменными.

Множество функциональных символов следующее: $\Sigma^F = \{ +_{\text{int}\times\text{int}\rightarrow\text{int}}, +_{\text{int}\times rat\rightarrow rat}, +_{rat\times\text{int}\rightarrow rat}, +_{rat\times rat\rightarrow rat}, -_{rat\rightarrow rat}, -_{\text{int}\rightarrow\text{int}} \}$. Формулы VL1 могут также содержать неинтерпретированные функциональные символы, которые мы не включаем явно в Σ^F. Конечное множество таких функциональных символов доопределяется, когда фиксирован контекст языка. В терминах системы VRS, для заданного описания среды [10], можно указать конечное множество неинтерпретированных функциональных символов *UF* явно.

Множество предикатных символов $\Sigma^P = \{ <_{\text{int}\times\text{int}}, <_{\text{int}\times rat}, <_{rat\times\text{int}}, <_{rat\times rat}, =_{\text{int}\times\text{int}}, =_{\text{int}\times rat}, =_{rat\times\text{int}}, =_{rat\times rat}, =_{symb\times symb} \}$.

Язык VL1 состоит из замкнутых Σ-формул вида $\exists_{\sigma_1} q_1 \ldots \exists_{\sigma_n} q_n(\varphi)$, где φ – бесквантерная формула, $n \geq 0$.

2. Задача выполнимости относительно теорий

Особенность задачи проверки выполнимости относительно теории (теорий) состоит в ограничении возможных интерпретаций логических символов в формулах. Материал этой главы кратко излагаем на основе [11, 12], куда отсылаем заинтересованного читателя для получения исчерпывающей информации по данной теме.

Для данного ЯМЛ сигнатуры Σ , любое множество T замкнутых Σ-формул называется Σ-*теорией*. Σ-интерпретация I называется Σ-*моделью* T, если эта интерпретация является моделью каждой формулы φ в T. Множество M всех Σ-моделей для данной теории T называется *классом* Σ-*моделей* T. С другой стороны, для данной интерпретации I можно разделить множество всех замкнутых Σ-формул на две группы в зависимости от того, являются ли они истинными в интерпретации I. Множество замкнутых формулы, истинных в I, составляет *теорию для интерпретации* I. Если множество интерпретаций M является классом моделей теории T, то T – *множество аксиом* для M. Если две теории имеют один и тот же класс моделей, они *эквивалентны*. В частности, две замкнутые формулы эквивалентны, если они истинны в точности на одних и тех же интерпретациях. Отметим что в литературе иногда теория определяется как (возможно бесконечное) множество Σ-моделей. В контексте данной работы различие в этих способах определения не является существенным.

Зафиксировав некоторую Σ-теорию T можно сформулировать *задачу выполнимости относительно теории* T, или, задачу T-*выполнимости* следующим образом: для данной замкнутой Σ-формулы φ выяснить, существует ли в классе Σ-моделей теории T такая интерпретация, в которой формула φ истинна. Другими словами, проверить, существует ли такая интерпретация I, что $I \models T \cup \{\varphi\}$. В англоязычной литературе это называют *satisfiability with respect to theory T, satisfiability modulo theory T, T-satisfiability problem, SMT problem*. С другой стороны, если теория задается классом своих Σ-моделей M, можно сформулировать задачу выполнимости относительно M так: для произвольной замкнутой Σ-формулы φ проверить справедливость $\models_M \varphi$. Проблема выполнимости для множества Σ-формул S относительно класса Σ-моделей M *разрешима*, если существует алгоритм, который проверяет справедливость $\models_M \varphi$ для любой формулы $\varphi \in S$. Выполнимость формулы относительно теории T будем обозначать $\models_T \varphi$.

По аналогии с задачей пропозициональной выполнимости (propositional satisfiability problem, SAT), процедуры решения задачи SMT называются SMT-*решателями* (SMT-*solvers*). Для процедур, решающих задачу выполнимости для конкретных теорий употребляется также термин *разрешающие процедуры* (*decision procedures*). Часто в SMT интерес представляет задача выполнимости относительно Σ-теории T для множества формул S, содержащих неинтерпретированные символы, т. е., дополнительные символы, не принадлежащие Σ. Например, множество констант может не ограничиваться Σ^C, с тем чтобы, для технического удобства, считать свободные переменные константами. В таких случаях можно рассматривать расширение исходной теории. Пусть Σ' – произвольная сигнатура, включающая Σ. Расширение Σ-модели A на Σ' – это такая Σ'-модель A', которая имеет тот же носитель и интерпретирует символы Σ таким же образом. Тогда класс моделей теории T' определяется как множество всех возможных расширений моделей T на Σ'. Для простоты обычно говорят о выполнимости относительно теории T, подразумевая ее подходящее расширение T', где это необходимо.

Рассмотрим примеры распространенных теорий, широко встречающихся при использовании SMT в различных формальных методах.

Теории линейной арифметики. Сигнатура Σ_{Ar} таких теорий включает константы 0, 1, предикатный символ $\leq$, функциональные символы +, –. *Теория* T_{Int} *линейной целочисленной арифметики* определяется стандартной интерпретацией сигнатуры Σ_{Ar} над множеством целых чисел. *Теория линейной арифметики над рациональными числами* T_{Real} определяется стандартной интерпретацией той же сигнатуры над множеством рациональных чисел.

Пример 1. Пусть $\Phi_1 := \exists x \exists y (x > 2 \,\&\, x < 4 \,\&\, 2 * y = x)$. Формула Φ_1 невыполнима относительно T_{Int} , но выполнима относительно T_{Real} .

Рассматривается также *теория* $T_{Int-Real}$ *смешанной линейной арифметики*, в которой вводятся два отдельных сорта (*int*, *rat*) для целых и рациональных чисел.

Пример 2. Формула $\Phi_2 := \forall x \,(x > 2 \,\&\, x < 4 \rightarrow x = 3 \,\&\, \exists y (2 * y = x))$ не выполнима в теориях T_{Int} и T_{Real} , но формула $\Phi' := \forall_{int} x \,(x > 2 \,\&\, x < 4 \rightarrow x = 3 \,\&\, \exists_{rat} y (2 * y = x))$ выполнима в теории $T_{Int-Real}$.

Отметим, что дополнительные символы выражаются через имеющиеся символы сигнатуры Σ_{Ar} . Например, $2*y = y+y$, $4 = 1+1+1+1$ и т. д.

Теория массивов. Сигнатура Σ_{Arr} содержит два функциональных символа *read*, *write*. Терм *read*(*a*, *i*) обозначает значение, находящееся по индексу *i* в массиве *a*. Терм *write*(*a*, *i*, *v*) обозначает массив, идентичный *a* для всех значений индекса за исключением *i*, которому соответствует элемент *v*. Класс моделей для T_{Arr} является классом моделей для следующего множества аксиом:

$\forall a \forall i \forall v (read(write(a, i, v), i) = v)$,

$\forall a \forall i \forall j \forall v (\neg(i = j) \rightarrow read(write(a, i, v), j) = read(a, j))$.

Иногда к ним добавляется *аксиома экстенсиональности*: $\forall a \forall b ((\forall i (read(a, i) = read(b, i))) \rightarrow a = b)$.

Теория массивов используется для моделирования массивов как структур данных. Несмотря на NP-полноту разрешающей процедуры в общем случае, разработанные T_{Arr}-решатели хорошо показывают себя на практике.

Иногда неинтерпретированные функциональные символы рассматриваются в контексте отдельной *теории неинтерпретированных функций с равенством*. Теория T_{EUF} также называется чистой теорией (*pure theory of equality with uninterpreted functions*) и не фиксирует в своей сигнатуре Σ_{EUF} никаких интерпретированных функциональных/предикатных символов. Класс ее моделей состоит из множества всех Σ-моделей.

Пример 3. Для формул

$\Phi_3 := \exists x \exists y (x = y \,\&\, y = z \,\&\, f(x) \neq f(z))$, $\Phi_4 := \exists x_1 \exists x_2 \exists x_3 (x_1 = x_2 \,\&\, f(x_1) \neq f(x_2) \vee f(x_1) = f(x_3))$ имеем, что Φ_3 невыполнима относительно T_{EUF}, в то время как Φ_4 – выполнима.

Подмена интерпретированных функций неинтерпретированными иногда позволяет упростить процесс доказательство выполнимости некоторых формул. В то же время такая абстракция не сохраняет свойство всюду истинности. Теория T_{EUF} может также использоваться для доказательства эквивалентности программ.

Практические применения SMT также включают использование теории битовых векторов для проведения рассуждений в области проектирования схем и программ, индуктивных типов данных для моделирования структур данных, теории строк, позволяющие, например, выражать свойства принадлежности к регулярным языкам конечной длины и многие другие теории.

3. Задача выполнимости для языка VL1

Возвращаясь к языку VL1 следует отметить, что нас интересуют не произвольные интерпретации символов VL1, а стандартные интерпретации, представляющие практическую ценность для решения задач верификации в контексте инсерционного моделирования, и того, как оно реализуется в VRS. Таким образом, будем рассматривать только такие интерпретации I формул VL1, которые удовлетворяют нижеследующим ограничениям (VL1-интерпретации).

1. Областью значений сорта *int* является множество целых чисел, область значений сорта *rat* – множество рациональных чисел. Область значений сорта *symb* – некоторое счетное бесконечное множество.

2. Независимо от типа функциональные символы + и – интерпретируются как сложение и унарный минус над рациональными числами. Предикат < имеет стандартную интерпретацию над множеством рациональных чисел, $=_{symb \times symb}$ интерпретируется как равенство над областью значений сорта *symb*, остальные предикаты равенства интерпретируются как равенство над множеством рациональных чисел.

3. Константы C_{rat} интерпретируются как соответствующие им рациональные константы. Константы C_{symb} интерпретируются произвольным образом так что $a, b \in C_{symb}$, $a \neq b \Rightarrow a^I \neq b^I$.

Итак, чтобы задать конкретную VL1-интерпретацию необходимо предоставить:

- область значений для сорта *symb*;
- интерпретацию функциональных символов *UF*;
- интерпретацию констант C_{symb}.

Обозначим V класс всех VL1-интерпретаций. Задача проверки выполнимости в VL1, рассматриваемая в данной работе, заключается в проверке справедливости $|\approx_V \varphi$ для произвольной формулы φ языка VL1.

Таким образом, задача проверки выполнимости формул в языке VL1 является задачей SMT. Действительно, задача заключается в проверке выполнимости формул этого языка как формул языка логики первого порядка, относительно следующих теорий:

- линейная смешанная арифметика (сорта *int*, *rat*);
- теория неинтерпретированных функциональных символов с равенствами (сорта *int*, *rat*, *symb*).

Для решения задачи выполнимости в VL1 мы используем известные методы решения задачи SMT, и, тем самым, показываем, что задача проверки выполнимости VL1-формул относительно класса VL1 интерпретаций разрешима.

Бóльшую часть SMT-методов, согласно устоявшейся терминологии [11], можно разделить на две группы – «жадные» (eager) и «ленивые» (lazy).

«Жадный» подход заключается в построении по формуле логики первого порядка эквивыполнимой пропозициональной формулы. Для многих логических теорий, использующихся в практических задачах SMT, это возможно. Правда, кодирование всех существенных формул-выводов теории может приводить к значительному увеличению размера формулы. Преимуществом данного подхода является возможность использования уже разработанных SAT-решателей, способных справиться с достаточно большими формулами. Эффективность подхода будет зависеть, в том числе, от особенностей работы алгоритма SAT-решателя над ограничениями, которые получаются в процессе кодирования формул рассматриваемой теории.

«Ленивый» подход состоит в создании процедур построения логического вывода, специализированных для данной теории T. Данный подход позволяет учитывать особенности конкретных теорий, использовать наиболее эффективные для рассматриваемой теории алгоритмы и структуры данных, что положительно влияет на производительность. Общепринятая практика состоит в разработке решателей для конъюнкции литералов теории T. С помощью решателя для конъюнкции литералов простейший алгоритм проверки выполнимости формулы заключается в построении дизъюнктивной нормальной формы, и решении задачи для каждого

конъюнкта по отдельности. Это называется синтаксическим разделением на подзадачи (syntactic case-splitting). Действительно, $I \mid\approx \exists_{\sigma_1} v_1 \ldots \exists_{\sigma_n} v_n (\varphi_1 \vee \varphi_2)$ тогда и только тогда, когда $I \mid\approx \exists_{\sigma_1} v_1 \ldots \exists_{\sigma_n} v_n \varphi_1$ либо $I \mid\approx \exists_{\sigma_1} v_1 \ldots \exists_{\sigma_n} v_n \varphi_2$.

Реализация такого алгоритма, конечно, не отличается эффективностью как с точки зрения временных затрат, так и с точки зрения объемов потребляемой оперативной памяти. Существует много различных вариантов «ленивых» SMT-решателей. Одна из центральных идей в их разработке состоит в интегрировании решателей для теории в процедуры решения задачи пропозициональной выполнимости (SAT engines). Известна следующая схема. Пусть φ_p – пропозициональная абстракция входной формулы φ. Формула φ_p получаемая путем замены атомов формулы φ пропозициональными символами отдается на вход SAT решателю. Если φ_p невыполнима, значит φ тоже невыполнима. В противном случае SAT-решатель возвращает некоторое решение, которое можно представить в виде конъюнкции μ_p пропозициональных литералов. Этой конъюнкции соответствует некоторая конъюнкция μ атомов из φ или их отрицаний. Затем μ проверяется на выполнимость уже *T*-решателем. Если μ выполнима относительно *T*, то φ тоже выполнима. В противном случае, к формуле φ конъюнктивно добавляется $\neg\mu$ и процесс проверки продолжается.

Более эффективные (соответственно, более сложные) схемы предполагают встраивание *T*-решателя непосредственно в алгоритмы SAT-решателя, наиболее популярным из которых является алгоритм DPLL (Davis-Putnam-Logemann-Loveland) [13]. Схемы такого вида получили обозначение DPLL(T) [14]. При этом, для эффективной интеграции *T*-решатель должен обладать дополнительными функциями, которые позволят DPLL-алгоритму сузить пространство перебора, используя дополнительную информацию из теории *T*. Важными являются следующие функции.

1. Способность генерации модели для выполнимой конъюнкции литералов μ.

2. Способность генерации противоречивого подмножества невыполнимой конъюнкции μ.

3. Свойство инкрементности: пусть решателем доказана выполнимость конъюнкции μ_1, и на следующем шаге необходимо проверить выполнимость $\mu_1 \,\&\, \mu_2$. Инкрементность означает, что выполнимость последней формулы будет проверяться с учетом имеющейся информации, полученной при проверке выполнимости μ_1, с целью сокращения вычислительных затрат. Для этого *T*-решатель поддерживает некоторое состояние, меняющееся с учетом формул, которые ему приходится доказывать.

4. Способность отката: *T*-решатель может эффективно возвращаться к предыдущему состоянию в процессе доказательств.

При решении практических SMT задач часто приходится сталкиваться с формулами, которые принадлежат сразу нескольким логическим теориям. В особенности это касается верификации, когда формулировки, требующие доказательства, накладывают ограничения, характеризующимися несколькими типами данных, где каждый тип данных моделируется своей собственной теорией.

Пример 4 [15]. Формула $\Phi_5 := b + 2 = c \,\&\, f(read(write(a,b,3),c-2)) \neq f(c-b+1)$ содержит символы сигнатуры теории T_{Int} (–, +), теории T_{Arr} (*read*, *write*), T_{EUF} (неинтепретированный символ *f*). Рассматривая интерпретацию функциональных и предикатных символов формулы согласно определениям соответствующих теорий можно убедиться, что формула Φ_5 не выполнима. Действительно, рассмотрим цепочку преобразований сохраняющих выполнимость:

$$b + 2 = c \,\&\, f(read(write(a,b,3),c-2)) \neq f(c-b+1) \Rightarrow$$
$$b + 2 = c \,\&\, f(read(write(a,b,3),b+2-2)) \neq f(b+2-b+1) \Rightarrow$$
$$b + 2 = c \,\&\, f(read(write(a,b,3),b)) \neq f(3) \Rightarrow$$
$$b + 2 = c \,\&\, f(3) \neq f(3).$$

Для указанных теорий, или их бесквантворных фрагментов, проблема выполнимости разрешима. Для них построены относительно эффективные *T*-решатели. Возникает вопрос: возможно ли комбинировать решатели для нескольких теорий так, чтобы получить решатель, проверяющий выполнимость формулы относительно объединенной теории? В общем случае ответ отрицательный, поскольку существуют теории выполнимые по отдельности, но не выполнимые в комбинации. Например, проблема выполнимости относительно комбинации T_{Int} и T_{EUF} разрешима для бесквантворного фрагмента, но не разрешима в общем случае.

Для Σ_1 -теории T_1 и Σ_2 -теории T_2 их комбинация $T = T_1 \oplus T_2$ определяется как Σ -теория такая, что $\Sigma = \Sigma_1 \cup \Sigma_2$, а T задается множеством аксиом $T_1 \cup T_2$. В терминах моделей в несколько упрощенном виде можно сказать, что класс Σ -моделей T совпадает с классом Σ -моделей для $T_1 \cup T_2$. Понятие комбинации может быть расширено на несколько теорий.

Наложив некоторые ограничения на теории T_1, T_2 можно построить комбинированный решатель для $T = T_1 \oplus T_2$, используя решатели для T_1 и T_2. Большинство методов комбинирования теорий в SMT являются расширениями и усовершенствованиями метода Нельсона – Оппена [16]. Ограничения, налагаемые методом, на

практике не являются чрезмерно строгими, и это позволят строить эффективные имплементации. Один из вариантов метода комбинирования налагает следующие ограничения на теории [12]:

- теории T_1 и T_2 являются бесквантормыми теориями с равенством;
- для T_1 и T_2 существует решатели;
- сигнатуры T_1 и T_2 не пересекаются;
- теории T_1 и T_2 интерпретируются на бесконечном носителе.

Без ограничения общности метод опишем для случая конъюнкции литералов $\varphi \in T_1 \oplus T_2$. Сначала производится разделение термов в исходной формуле φ путем введения дополнительных переменных. После этого получается эквивыполнимая формула $\varphi' = \varphi_1 \,\&\, \varphi_2$, где каждый литерал уже принадлежит какой-то одной из теорий, т. е., $\varphi_1 \in T_1$ и $\varphi_2 \in T_2$. Рассмотрим для примера формулу $x_1 + f(x_1) > 0$. В этой формуле единственный предикат содержит символы сигнатуры двух теорий T_{Int} и T_{EUF}. Введением новой переменной x_2 получаем формулу $x_1 + x_2 > 0 \,\&\, x_2 = f(x_1)$, где $x_2 = f(x_1) \in T_{EUF}$ и $x_1 + x_2 > 0 \in T_{Int}$. Обозначим $Var = \{v_1, ..., v_n\}$ множество переменных встречающихся одновременно в φ_1 и φ_2. На абстрактном уровне метод комбинации Нельсона – Оппена состоит в поиске такой формулы $\psi = \mathop{\&}\limits_{i,j \in \{1,...,n\}} v_i \cong v_j$, что одновременно выполняется $|\approx_{T_1} \varphi_1 \,\&\, \psi$ и $|\approx_{T_2} \varphi_2 \,\&\, \psi$. Это будет означать, что $|\approx_T \varphi'$, и следовательно, $|\approx_T \varphi$. Если такую формулу ψ построить нельзя, то φ не выполнима в *T*. Формула ψ состоит из атомов вида $v_i = v_j$ для каждой пары переменных $v_i, v_j \in Var$, где каждый такой атом может входить в чистом виде, либо с отрицанием $\neg(v_i = v_j)$.

Более детальное обсуждение общих методов решения SMT задачи выходит за рамки данной работы, но хотелось бы отметить следующие выводы:

- при построении *T*-решателей достаточно иметь в наличии алгоритм проверки выполнимости для конъюнкции литералов теории *T*;
- если $Var = \varnothing$, то для установления $|\approx_T \varphi$ достаточно проверить $|\approx_{T_1} \varphi_1$ и $|\approx_{T_2} \varphi_2$;
- поскольку алгоритмы DPLL-решателей рассчитаны на запись формулы в виде конъюнктивной нормальной формы (КНФ), на практике исходная формула вначале приводится к эквивыполнимой формуле представленной в КНФ.

Рассмотрим специализированные SMT методы решения проблемы выполнимости в языке VL1. Пусть φ – формула языка VL1. Проверка выполнимости φ может быть произведена следующим образом.

Сначала исходная формула φ может быть приведена к эквивыполнимой формуле без неинтерпретированных функциональных символов. Для этого рассмотрим каждую (синтаксически различную) пару термов $f(t_1, ..., t_n)$ и $f(u_1, ..., u_n)$, $f \in UF$ типа $\sigma_1 \times \cdots \times \sigma_n \to \sigma$, и конъюнктивно добавим к φ формулу $t_1 =_{\sigma_1} u_1 \,\&\, \ldots \,\&\, t_n =_{\sigma_n} u_n \to f(t_1, ..., t_n) =_\sigma f(u_1, ..., u_n)$. В полученной формуле заменим каждый (самый внешний) терм $f(t_1, ..., t_n)$ сорта σ, новой переменной $v_{f(t_1,...,t_n)}$ сорта σ. Синтаксически идентичные термы заменяются одной и той же переменной. Хотя для вложенных термов, содержащих неинтерпретированные функциональные символы, новой переменной заменяется только внешний терм, вложенные подтермы тоже рассматриваются при построении импликаций. В итоге, каждую новодобавленную переменную необходимо связать квантором существования в префиксе формулы. Полученную формулу обозначим φ'. Эквивыполнимость $|\approx_V \varphi \Leftrightarrow |\approx_V \varphi'$ может быть доказана непосредственно: для данной интерпретации $I \in V$ такой, что $I |\approx \varphi'$, мы всегда можем построить интерпретацию $J \in V$ такую, что $J |\approx \varphi$ и наоборот. Это преобразование известно под названием редукция Аккермана [17]. Отметим, что после применения редукции атомы, содержащие переменные или константы сорта *symb*. не содержат термов других сортов в качестве своих подтермов.

Без ограничения общности будем считать φ конъюнкцией литералов. Пусть φ имеет вид $\varphi = \exists_{\sigma_1} v_1 \ldots \exists_{\sigma_k} v_k \exists_{symb} s_1 \ldots \exists_{symb} s_l (\chi_1 \,\&\, \ldots \,\&\, \chi_n \,\&\, \psi_1 \,\&\, \ldots \,\&\, \psi_m)$, где $k,\ l,\ n,\ m \geq 0$, $\sigma_i \in \{int, rat\}$, литералы χ_i не содержат термов сорта *symb*, а литералы ψ_i представляют собой равенства термов типа *symb* или отрицания таковых. Обозначим $\chi = \exists_{\sigma_1} v_1 \ldots \exists_{\sigma_k} v_k (\chi_1 \,\&\, \ldots \,\&\, \chi_n)$ и $\psi = \exists_{symb} s_1 \ldots \exists_{symb} s_l (\psi_1 \,\&\, \ldots \,\&\, \psi_m)$. Формулы χ и ψ являются замкнутыми. Поскольку в формулах χ и ψ не может быть общих переменных, имеем $|\approx_V \varphi$ тогда и только тогда, когда $|\approx_V \chi$ и $|\approx_V \psi$.

Задача проверки выполнимости $|\approx_V \chi$ может быть решена с помощью обобщенного симплекс-метода [18] или алгоритмов элиминации кванторов в теории смешанной арифметики над целыми и вещественными числами [19]. Обработка целочисленных переменных является более дорогостоящей по вычислительной сложности. Известно [20], что проверка непротиворечивости системы линейных неравенств в рациональных числах может быть выполнена за полиномиальное время, в то время как аналогичная задача в целых числах является NP-полной. В [19] утверждается, что проверка выполнимости замкнутых формул в

языке смешанной арифметики может быть выполнена за время $2^{l^{n^{O(a)}}}$, где l – длина формулы, n – число переменных, a – число перемен кванторов в пренексной нормальной форме формулы.

Задача проверки выполнимости $|\approx_V \psi$ может быть решена построением фактор-множества по отношению эквивалентности следующим образом. Сначала для каждой пары различных констант c_1, c_2 встречающихся в ψ конъюнктивно добавим к ψ дополнительный литерал $\neg(c_1 =_{symb} c_2)$. Полученную формулу обозначим ψ'. Пусть $W = \{w_1,...,w_q\}$ – множество переменных и констант формулы ψ'. Ассоциируем с каждым элементом w_i класс эквивалентности $\{w_i\}$, состоящий из этого элемента. Затем, для каждого литерала $w_i =_{symb} w_j$ в формуле ψ объединяем классы $\{w_i\}$ и $\{w_j\}$. Далее, пока это возможно, объединяем классы эквивалентности содержащие общие элементы. Впрочем, исходная формула ψ выполнима тогда и только тогда, когда в ψ для каждого литерала вида $\neg(w_i =_{symb} w_j)$ имеем, что w_i и w_j принадлежат разным классам эквивалентности. Расширение этого метода может использоваться для проверки выполнимости конъюнкции формул теории T_{EUF}. Можно построить соответствующий алгоритм [21], имеющий сложность O($n \log n$) для конъюнкции длины n.

Вместо редукции Аккермана могут также использоваться SMT-методы, решающие задачу комбинации теорий.

4. Метод поиска выполнимой конъюнкции для формулы с множественными операциями

В разделе 3 мы ограничились решением задачи выполнимости для конъюнкции литералов языка VL1. Как было отмечено, при решении проблемы выполнимости общепринятой практикой считаются переборные методы, использующие предварительное представление исходной формулы в КНФ. Это мотивируется использованием требующей КНФ достаточно развитой схемы *DPLL*(*T*), для которой разработаны многочисленные эвристики и усовершенствования. В то же время, для дизъюнктивной нормальной формы задача выполнимости решается проще, и сводится к простой проверке выполнимости каждого конъюнкта, а в случае пропозициональной формулы задача вообще решается тривиально. На этой почве возник вопрос: всегда ли имеет смысл проводить поиск выполнимой конъюнкции, выполняя безоговорочное приведение формулы к КНФ, в особенности, если представление формулы близко к дизъюнктивной нормальной форме?

Этот вопрос послужил мотивацией для разработки альтернативного подхода к поиску выполнимой конъюнкции литералов, являющейся импликантой исходной формулы. Суть подхода состоит в следующем. Формула представляется в виде дерева, вершинами которого являются множественные операции конъюнкции (&) и дизъюнкции (∨), а листами – литералы либо их отрицания. Множественность подчеркивает то, что количество аргументов операции (арность) не ограничивается, а порядок аргументов не устанавливается. С каждой (∨)-вершиной ассоциируется индекс, определяющий одного из ее потомков. Комбинация значений всех индексов (∨)-вершин однозначно определяет конъюнкцию предикатов, которая проверяется на предмет выполнимости. На множестве возможных комбинаций вводится линейный порядок, согласно которому производится последовательный поиск выполнимой конъюнкции.

При этом, для сокращения поиска, каждая невыполнимая конъюнкция анализируется с целью нахождения противоречивого подмножества ее литералов. Задача состоит в обнаружении противоречивого подмножества, встречающегося в как можно большем числе конъюнкций, еще не проанализированных, в соответствии с порядком их рассмотрения алгоритмом. После выделения такого подмножества, часть конъюнкций удается отбросить, как заведомо противоречивые.

Рассмотрено несколько реализаций предлагаемого подхода, которые отличались способом упорядочивания множества аргументов (∨)-вершин. В первом варианте порядок вершин был произвольным и определялся представлением исходной формулы. Во втором и третьем варианте аргументы сортировались в порядке возрастания и, соответственно, убывания числа вершин в своих поддеревьях. Для обнаружения противоречивого подмножества предикатов для невыполнимой конъюнкции литералов $p_1\&\ldots\&p_n$ бинарным поиском находился такой индекс i, что формула $p_i\&...\&p_n$ не выполнима, но $p_{i+1}\&...\&p_n$ выполнима. Непосредственно следующие конъюнкции, содержащие все предикаты $p_i...p_n$ исключались из рассмотрения. Реализации алгоритмов были опробованы в процессе работы системы VRS. При этом оценивалось соотношение числа дополнительных вызовов разрешающей процедуры для выделения противоречивого подмножества предикатов, и количество обнаруженных противоречивых конъюнктов исходной формулы. Для лучшей из реализаций (третий вариант) число вызовов разрешающей процедуры для выполнимых формул составило 2% от общего числа конъюнктов, составляющих ДНФ, для невыполнимых формул – 4%.

В данный момент было бы не совсем адекватно считать предложенный метод конкурентом схеме *DPLL*(*T*), которая активно совершенствуется более 10 лет, но отметим, что оба метода являются переборными и имеют выраженную эвристическую природу. Построение множественного представления выполняется за линейное время. Преимуществом предложенного метода поиска выполнимой конъюнкции является линейная сложность проверки выполнимости формулы для ДНФ-представления, без учета сложности разрешающей процедуры. Проведенные эксперименты продемонстрировали возможность применения предложенного

подхода на практических примерах из системы верификации требований VRS. Показатели эффективности различных способов его реализации отличаются, поэтому вопрос об эффективности конкретных реализаций и эвристик может быть исследован дополнительно.

Выводы

В настоящей работе мы описали логический язык VL1 и показали, что задача выполнимости в этом языке разрешима. Предложена схема разрешающей процедуры для этого языка, которая использует вспомогательные методы такие, как редукция Аккермана, обобщенный симплекс метод, процедура построения замыкания по отношению эквивалентности. Указанная схема может применяться в комбинации с пропозициональными решателями согласно «ленивому» подходу к решению задачи выполнимости относительно теории. Также мы предложили альтернативный метод поиска выполнимой конъюнкции для формулы с множественными операциями, не требующей приведения формулы к конъюнктивной нормальной форме. Метод позволяет существенно сократить число вызов разрешающей процедуры в процессе поиска выполнимой конъюнкции и может использоваться для решения типичной задачи SMT при наличии решателя для конъюнкции литералов.

Язык VL1 используется в системе верификации требований VRS как язык для проведения формальных рассуждений. Отметим, что требования, предъявляемые новой версией системы VRS к SMT-решателю, требуют усовершенствования выразительных возможностей языка VL1. В частности, предполагается использовать операции-конструкторы для символьных термов и ограниченную операцию универсальной квантификации. В последующих работах планируется исследовать вопросы выполнимости в расширенном языке, находя компромисс между выразительными возможностями и сложностью разрешающих процедур. Поскольку введение универсальной квантификации уже делает проблему выполнимости неразрешимой в общем случае, мы заинтересованы в изучении и выделении разрешимых фрагментов языка, которые будут достаточными для проведения рассуждений для при практическом использовании системы VRS.

В заключение отметим, что современные успехи исследований в направлении выполнимости относительно теорий (satisfiability modulo theories, SMT) свидетельствуют о широком спектре применимости разработанных в этой области методов. Построены разрешающие процедуры для теорий конечных множеств, мультимножеств, решеток, конечных, регулярных и бесконечных деревьев, списков, записей, очередей, свободных термов, хэш-таблиц, массивов, линейной арифметики [1, 11]. Разработаны эффективные решатели, среди которых отметим программные системы Z3, CVC4, Barcelogic, MathSAT, OpenSMT, SatEEn, STP, UCLID, Yices. Некоторые из них являются свободно распространяемыми системами с открытым кодом. Список решателей и поддерживаемых ими теорий продолжает увеличиваться, и, безусловно, достижения в области SMT не могут оставаться без внимания при разработке систем верификации. Некоторые из указанных разрешающих процедур планируется применить и в системе VRS.

1. *de Moura L., Bjørner N.* Satisfiability Modulo Theories: Introduction and Applications // Communications of the ACM. –2011. – Vol. 54 (9). – P. 69–77.
2. The 15-th International Conference on Theory and Applications of Satisfiability Testing – Режим доступа: http://sat2012.fbk.eu/
3. Satisfiability Modulo Theories Competition (SMT-COMP) – Режим доступа: http://www.smtcomp.org
4. *Letichevsky A.* Algebra of behavior transformations and its applications // in V.B. Kudryavtsev and I.G. Rosenberg editors. Structural theory of Automata, Semigroups, and Universal Algebra, NATO Science Series II. Mathematics, Physics and Chemistry. – Springer. – 2005. – Vol. 207. – P. 241–272.
5. *Letichevsky A., Kapitonova J., Kotlyarov V. et al.* Insertion Modeling in Distributed System Design // Problems of Programming. – 2008. – Vol. 4. – P. 13–39.
6. *Letichevsky A.A., Letychevskyi O.A., Peschanenko V.S.* Insertion Modeling System // Lecture Notes in Computer Science. – Springer. – 2011. – Vol. 7162. – P. 262–274.
7. *Kapitonova J., Letichevsky A., Volkov V. et al.* System Validation // In R. Zurawski, editor. The Embedded Systems Handbook. – Miami: CRC Press, 2005. – P. 6.1 – 6 – 57.
8. *Letichevsky A., Kapitonova J., Letichevsky A. (jr). et al.* Basic Protocols, Message Sequence Charts, and the Verification of Requirements Specifications // Computer Networks. – 2005. – N 47. – P. 662– 675.
9. *Летичевский А.А., Капитонова Ю.В., Волков В.А. и др.* Сертификация систем с помощью базовых протоколов // Кибернетика и системный анализ. – 2005. – № 4. – С. 256–268.
10. *Letichevsky A., Gilbert D.* A Model for Interaction of Agents and Environments // In D. Bert, C. Choppy, P. Moses, editors. Recent Trends in Algebraic Development Techniques. Lecture Notes in Computer Science. – Springer. – 1999. – Vol. 1827. – P. 311–328.
11. *Barrett C., Sebastiani R., Seshia S. A., Tinelli C.* Satisfiability Modulo Theories // In: A. Biere, M. Heule., H. van Maaren, T. Walsh editors. Handbook of Satisfiability. – IOS Press. – 2009. – P. 737–797.
12. *Kroening D., Strichman O.* Decision Procedures – an Algorithmic Point of View. – Berlin: Springer-Verlag, 2008. – 304 p.
13. *Davis M., Logemann G., Loveland D.* Machine Program for Theorem-Proving // Comm. of the ACM. –1962. – № 5(7). – P. 7:394–397.
14. *Ganzinger H., Hagen G., Nieuwenhuis R. et al.* DPLL (T): Fast decision procedures // Proc. of the 16-th Intl. Conf. on Computer Aided Verification, Lecture Notes in Computer Science, 3114. – 2004. – P. 175–188.
15. *de Moura L., Bjørner N.* Satisfiability Modulo Theories: An Appetizer // Lecture Notes in Computer Science. –2009. – Vol. 5902. – P. 23–36.
16. *Nelson G., Oppen D.C.* Simplification by cooperating decision procedures // ACM Transactions on Programming Languages and Systems. – 1979. – Vol. 1. – P. 245–257.
17. *Ackermann W.* Solvable cases of the Decision Problem. – Amsterdam: North-Holland, 1954. – 114 p.
18. *Dutertre B., de Moura L.* Integrating Simplex with DPLL(T) // Technical report, CSL-06-01, SRI International. – 2006. – 35 p.
19. *Weispfenning V.* Mixed Real-Integer Linear Quantifier Elimination // Proceedings of the international symposium on Symbolic and algebraic computation ISSAC'99. – 1999. – P. 129–136.
20. *Schrijver A.* Theory of Linear and Integer Programming. – John Wiley & sons. – 1998. – 471 p.
21. *Nieuwenhuis R., Oliveras R.* Fast Congruence Closure and Extensions // Information and Computation. – 2007. – Vol. 205(4). – P. 557–580.