УДК 004.021
DOI: 10.15827/0236-235X.128.556-564
Дата подачи статьи: 02.04.19
2019. Т. 32. № 4. С. 556–564

Метод трансляции первопорядковых логических формул в позитивно-образованные формулы

А.В. Давыдов [1], *научный сотрудник, artem@icc.ru*
А.А. Ларионов [1], *программист, bootfrost@zoho.com*
Е.А. Черкашин [1], *старший научный сотрудник, eugeneai@icc.ru*

[1] *Институт динамики систем и теории управления им. В.М. Матросова СО РАН, г. Иркутск, 664033, Россия*

В статье рассматриваются логическое исчисление позитивно-образованных формул (ПОФ-исчисление) и построенный на его основе метод автоматического доказательства теорем. ПОФ-исчисление впервые появилось в работах академиков РАН С.Н. Васильева и А.К. Жерлова в результате рассмотрения и решения задач теории управления и было описано как логический формализм первого порядка. Имеются примеры описания и решения задач теории управления, эффективно (с точки зрения выразительности языка и производительности средств доказательств теорем) решенных с помощью ПОФ-исчисления, например, управление группой лифтов, наведение телескопа на центр планеты, находящейся в неполной фазе, управление мобильным роботом. ПОФ-исчисление выгодно отличается от возможностей других, логических, средств формализации предметной области и поиска логических выводов выразительностью в сочетании с компактностью представления знаний, естественным параллелизмом их обработки, крупноблочностью и меньшей комбинаторной сложностью выводов, высокой совместимостью с эвристиками и широкими возможностями для интерактивного доказательства. В выделенном классе формул возможно построение конструктивного доказательства. Данный класс формул существенно шире класса хорновских дизъюнктов, используемых в языке Пролог: на логическую формализацию аксиоматической базы предметной области не накладываются никакие ограничения, а целевое утверждение – это конъюнкция запросов (в смысле языка Пролог).

Для тестирования программной системы автоматического доказательства теорем (прувера), основанной на ПОФ-исчислении, использовалась библиотека задач TPTP (Thousands of Problems for Theorem Provers). Формат, в котором представлены задачи TPTP (называемые проблемами), де-факто стал стандартом среди сообщества, изучающего автоматизацию рассуждений. Возникает естественная необходимость в том, чтобы разрабатываемый прувер принимал на вход задачи в этом формате. Таким образом, возникла задача трансляции формул логики предикатов первого порядка, представленных в формате TPTP, в формат ПОФ. Эта задача нетривиальна из-за особой структуры формул ПОФ-исчисления.

В данной работе предложены более эффективный (в сравнении с ранее разработанным алгоритмом в первой реализации системы автоматического доказательства теорем для ПОФ-исчисления) метод трансляции формул первопорядкового языка исчисления предикатов с сохранением исходной эвристической структуры знаний и его упрощенная версия для задач, представленных на языке дизъюнктов. Под эффективностью понимаются количество шагов и длина получаемых формул. Предложенный метод был реализован в виде программной системы – транслятора языка первопорядковых логических формул в формате TPTP в язык ПОФ. Приведены результаты тестирования разработанного метода, которые позволяют сделать вывод о том, что существует определенный класс первопорядковых формул, не принимаемый во внимание как особый существующими системами автоматического доказательства теорем, в то время как в ПОФ-исчислении для данного класса формул существуют специальные стратегии, повышающие эффективность поиска вывода.

***Ключевые слова*:** *математическая логика, автоматическое доказательство теорем, алгоритмы трансляции.*

Сегодня активно развиваются *автоматическое доказательство теорем* (АДТ) и его приложения, о чем свидетельствуют, например, работы ежегодной конференции CADE, на секции System description которой или представляются новые системы для АДТ, или показывается развитие ранее разработанных.

В данной статье рассматриваются логическое исчисление *позитивно-образованных формул* (ПОФ-исчисление) и построенный на его основе метод АДТ. ПОФ-исчисление выгодно отличается от возможностей других, логических, средств формализации предметной области и поиска логических выводов вырази-

тельностью в сочетании с компактностью представления знаний, естественным параллелизмом их обработки, крупноблочностью и меньшей комбинаторной сложностью выводов, высокой совместимостью с эвристиками и более широкими возможностями для интерактивного доказательства. В выделенном классе формул возможно построение конструктивного доказательства. Данный класс существенно шире класса хорновских дизъюнктов, используемых в Прологе: на логическую формализацию аксиоматической базы предметной области не накладывается никаких ограничений, а целевое утверждение – это конъюнкция запросов в смысле языка Пролог.

Понятие ПОФ-исчисления появилось в работах в результате описания и решения задач теории управления [1, 2]. ПОФ-исчисление описано в них как логический формализм первого порядка; приводятся примеры описания и решения задач теории управления, эффективно (с точки зрения выразительности языка и производительности средств доказательств теорем) решенных с помощью ПОФ-исчисления, например, управление группой лифтов, наведение телескопа на центр планеты, находящейся в неполной фазе, управление мобильным роботом, а также доказательства корректности и полноты исчисления.

Для тестирования программной системы АДТ [3], основанной на ПОФ-исчислении, использовалась библиотека задач TPTP (Thousands of Problems for Theorem Provers) [4]. Формат, в котором представлены задачи TPTP (называемые проблемами), де-факто стал стандартом среди сообщества, изучающего автоматизацию рассуждений. На сайте библиотеки TPTP представлены многие системы АДТ, например, имеющая более чем двадцатилетнюю историю система Vampire [5] или одна из новых систем Scavenger [6] и др. А в работе [7] рассматривается программа-конвертер из формата SMT-LIB в формат TPTP. Появляется естественная необходимость в том, чтобы разрабатываемый прувер принимал на вход задачи в этом формате. Таким образом, возникла необходимость трансляции формул логики предикатов первого порядка, представленных в формате TPTP, в формат ПОФ. Эта задача нетривиальна из-за особой структуры формул ПОФ-исчисления.

В данной работе предложен более эффективный (в сравнении с ранее разработанным алгоритмом в первой реализации системы автоматического доказательства теорем для ПОФ-исчисления [8]) метод трансляции. Под эффективностью понимаются количество шагов и длина получаемых формул. Приведены результаты тестирования разработанного метода, которые позволяют сделать вывод о том, что существует определенный класс первопорядковых формул, не принимаемый во внимание как особый системами АДТ, в то время как в ПОФ-исчислении для данного класса формул есть специальные стратегии, повышающие эффективность поиска вывода. Использование специальных стратегий вывода является перспективным направлением в области АДТ, в частности, в работе [9] приводится язык стратегий для прувера Isabelle/HOL.

Предварительные определения. Будем говорить о языке *первопорядковых логических формул* (ПЛФ), построенных на основе атомарных формул с помощью связок $\&$, $\vee$, $\neg$, $\rightarrow$, $\leftrightarrow$, кванторов $\forall$, $\exists$ и констант *True*, *False*. Понятия «терм», «атом», «литера» определяются обычным образом.

Пусть $X = \{x_1, \ldots, x_k\}$ – множество переменных, $A = \{A_1, \ldots, A_m\}$ – множество атомарных формул, $F = \{F_1, \ldots, F_n\}$ – множество подформул. Тогда формулы

$$((\forall x_1)\ldots(\forall x_k)(A_1 \,\&\ldots\&\, A_m \rightarrow (F_1 \vee \ldots \vee F_n))),$$

$$((\exists x_1)\ldots(\exists x_k)(A_1 \,\&\ldots\&\, A_m \,\&\, (F_1 \,\&\ldots\&F_n)))$$

будем обозначать как $\forall_X A{:}F$ и $\exists_X A{:}F$ соответственно, имея в виду, что $\forall$-квантор соответствует $\rightarrow F^{\vee}$, где $F^{\vee}$ означает дизъюнкцию всех подформул из F, а $\exists$-квантор соответствует $\rightarrow F^{\&}$, где $F^{\&}$ означает конъюнкцию всех подформул из F.

Если $F = \varnothing$, формулы имеют вид $\forall_X A{:}\varnothing \equiv \equiv \forall_X A \rightarrow \mathit{False}$ и $\exists_X A{:}\varnothing \equiv \exists_X A \,\&\, \mathit{True}$, так как дизъюнкция пустого количества элементов соответствует *False*, в то время как конъюнкция пустого количества элементов соответствует *True*. Будем записывать такие формулы, соответственно, $\forall_X A$ и $\exists_X A$. Если $X = \varnothing$, тогда $\forall A{:}F$ и $\exists A{:}F$ также являются сокращением.

Множество атомов A называется конъюнктом. Еще раз отметим, что пустой конъюнкт эквивалентен *True*.

Переменные из X связаны соответствующими кванторами и называются соответственно $\forall$-переменными и $\exists$-переменными. В $\forall_X A$ переменные из X, которые не встречаются в конъюнкте A, называются неограниченными переменными.

Конструкции $\forall_X A$ и $\exists_X A$ называются позитивными типовыми кванторами. Поскольку A является конъюнкцией только лишь положи-

тельных атомов, эту конъюнкцию также называют типовым условием для X. На практике такие конструкции обозначают, например, фразы: «для любого X, удовлетворяющего A, следует…», «существует X, удовлетворяющее условию A, такое что…» или «Для всех целых чисел x, y, z и $n > 2$ выполнимо $x^n + y^n \neq z^n$». Изначально термин «типовый квантор» был предложен Николя Бурбаки как часть обозначений для формализации математики, однако типовые кванторы оказались применимыми и в других областях.

Язык ПОФ

Определение ПОФ. Пусть X – множество переменных, A – конъюнкт.

• $\exists_X A$ и $\forall_X A$ называются ∃-ПОФ и ∀-ПОФ соответственно.

• Если $F = \{F_1, \ldots, F_n\}$ является ∀-ПОФ, тогда $\exists_X A{:}F$ будет называться ∃-ПОФ.

• Если $F = \{F_1, \ldots, F_n\}$ является ∃-ПОФ, тогда $\forall_X A{:}F$ будет называться ∀-ПОФ.

• Всякая ∃-ПОФ и ∀-ПОФ будет называться ПОФ.

Такое представление логических формул называется ПОФ, так как они записываются только с помощью позитивных типовых кванторов. Формулы не содержат знак отрицания в явном виде. Любая ПЛФ может быть представлена в виде ПОФ [2]. Таким образом, ПОФ есть особая форма записи классических формул языка предикатов подобно КНФ, ДНФ и др.

Те ПОФ, которые начинаются с $\forall\varnothing$, называются ПОФ в канонической форме. Любая ПОФ может быть приведена к канонической. Пусть F – это ∃-ПОФ, тогда формула $\forall\varnothing{:}\ F \equiv \equiv True \rightarrow F \equiv F$. Если F – это неканоническая ∀-ПОФ, тогда $\forall\varnothing{:}\ \{\exists\varnothing{:}\ F\} \equiv True \rightarrow \{True\ \&\ \&\ F\} \equiv F$. Типовые кванторы $\forall\varnothing$ и $\exists\varnothing$ называются фиктивными, так как они не влияют на истинностное значение исходной ПОФ, а только служат конструкциями, сохраняющими корректную запись ПОФ.

Для удобства читаемости формул будем представлять их в древовидной форме. Запись $Q_X A{:}\{F_1, \ldots, F_n\}$ эквивалентна

$$Q_X A \begin{cases} F_1 \\ \cdots, \\ F_n \end{cases}$$

где Q – некоторый квантор. Элементы дерева называются как обычно: узел, корень, лист, ветвление и т.д. Поскольку ∀-кванторы соответствуют дизъюнкциям подформул $\{F_1, \ldots, F_n\}$, а кванторы ∃ соответствуют конъюнкции, все ∀-узлы соответствуют дизъюнктивному ветвлению, а ∃-узлы конъюнктивному.

Некоторые части канонической ПОФ будем называть следующим образом:

– корень ПОФ $\forall\varnothing$ называется корнем ПОФ;

– каждый непосредственный последователь $\exists_X A$ корня ПОФ называется базой; конъюнкт A называется базой фактов; ПОФ, корнем которой является база, называется базовой подформулой;

– каждый непосредственный последователь $\forall_Y B$ базы ПОФ называется вопросом к своей базе; если вопрос является листом дерева, то он называется целевым вопросом;

– поддеревья вопросов называются консеквентами; пустая консеквента эквивалента $False$.

Пример. Рассмотрим ПОФ-представление некоторой ПЛФ:

$F = \neg(\forall x \exists y\, P(x, y) \rightarrow \exists z\, P(z, z))$.

Образом F в языке ПОФ является

$F' = \forall{:}\varnothing\{\forall x{:}\varnothing\{\exists y{:}P(x, y)\},\ \forall z{:}P(z, z)\{\exists{:}False\}\}$.

Последняя ПОФ в древовидной форме имеет следующий вид:

$$\forall{:}\varnothing - \exists{:}\varnothing \begin{cases} \forall x : \varnothing - \exists y : P(x, y), \\ \forall z : P(z, z) - \exists : False. \end{cases}$$

Описание шагов метода трансляции. Метод трансляции состоит из следующих шагов.

Шаг 1. Преобразование задачи ТРТР в ПЛФ. Полученная ПЛФ представляет собой отрицание конъюнкции всех формул, входящих в аннотированные формулы. Отрицание производится с той целью, что система АДТ опровергает отрицание исходной формулы, тем самым подтверждая ее доказуемость.

Шаг 2. Удаление связок $\rightarrow$, $\leftrightarrow$, снятие двойного отрицания и применение законов де Моргана. Пусть F (возможно, с индексом) – произвольная неатомарная ПЛФ, A – атомарная ПЛФ. Тогда имеем правила преобразований (назовем эту группу преобразований $Tr1$), представленные в таблице 1.

Шаг 3. Уплощение конъюнкций и дизъюнкций.

Пусть $\bullet \in \{\vee, \&\}$. Правило уплощения дизъюнкций и конъюнкций *flat*.

Пусть $F = G_1 \bullet \ldots \bullet G_n \bullet C_1 \bullet \ldots \bullet C_m$, где $C_i = (C_{i1} \bullet \ldots \bullet C_{iki})$, тогда $F^{flat} = G_1 \bullet \ldots \bullet G_n\ \&\ C_{11} \bullet \ldots \bullet C_{1k1} \bullet \ldots \bullet C_{m1} \bullet \ldots \bullet C_{mkm}$.

Таблица 1

Правила преобразования *Tr*1

Table 1

***Tr*1 transformation rules**

№	Исходная формула	Результирующая формула
1	$\neg\neg F$	F^{Tr1}
2	$\neg(F_1 \to F_2)$	$F_1{}^{Tr1}\&\neg F_2{}^{Tr1}$
3	$\neg(F_1 \leftrightarrow F_2)$	$(\neg(F_1 \to F_2))^{Tr1}\vee(\neg(F_2 \to F_1))^{Tr1}$
4	$\neg(F_1\&\ldots\&F_n)$	$(\neg F_1)^{Tr1}\vee\ldots\vee(\neg F_n)^{Tr1}$
5	$\neg(F_1\vee\ldots\vee F_n)$	$(\neg F_1)^{Tr1}\&\ldots\&(\neg F_n)^{Tr1}$
6	$\neg\forall_X F$	$\exists_X(\neg F)^{Tr1}$
7	$\neg\exists_X F$	$\forall_X(\neg F)^{Tr1}$
8	$\neg A$	$\neg A$
9	$F_1 \to F_2$	$(\neg F_1)^{Tr1}\vee F_2{}^{Tr1}$
10	$F_1 \leftrightarrow F_2$	$(\neg F_1\vee F_2)^{Tr1}\&(\neg F_2\vee F_1)^{Tr1}$
11	$F_1\&\ldots\&F_n$	$F_1{}^{Tr1}\vee\ldots\vee F_n{}^{Tr1}$
12	$F_1\vee\ldots\vee F_n$	$F_1{}^{Tr1}\vee\ldots\vee F_n{}^{Tr1}$
13	$\forall_X F$	$\forall_X(F)^{Tr1}$
14	$\exists_X F$	$\exists_X(F)^{Tr1}$

Смысл «уплощения» хорошо виден, если формулу представить синтаксическим деревом. Например, следующие требования эквивалентны:

$$\forall x - \& \begin{cases} F_1 \\ \& \begin{cases} G_1 \\ G_2 \end{cases} \\ F_2 \end{cases} \text{ и } \forall x - \& \begin{cases} F_1 \\ G_1 \\ G_2 \\ F_2 \end{cases}.$$

Шаг 4. Трансляция ПЛФ в ПОФ. Пусть G (возможно, с индексами) есть некоторая ПЛФ, а A (возможно, с индексами) – атомарная ПЛФ.

Введем еще одно обозначение. Пусть $P, Q \in \{\forall, \exists\}$, $P \neq Q$, а C – некоторый конъюнкт, тогда:

$$F^Q = \begin{cases} F, \text{если } F = Q_X : C\Phi, \\ Q : \varnothing(F), \text{если } F = P_X : C\Phi. \end{cases}$$

Правила преобразования *Tr*2 можно представить в виде таблицы 2. Правило уплощения позволяет ограничить, насколько это возможно, применение правил 13 и 14, которые приводят к появлению фиктивных кванторов.

Шаг 5. Правило сокращения.

В получаемой формуле возникают узлы с фиктивными кванторами $\forall\varnothing$ и $\exists\varnothing$, кроме этого, могут возникнуть излишние ветвления, плохо влияющие на дальнейший поиск логического вывода. Следующее правило позволяет максимально сократить получаемое дерево.

Таблица 2

Правила преобразования *Tr*2

Table 2

***Tr*2 transformation rules**

№	ПЛФ	ПОФ
1	$\exists_X\exists_Y G$	$(\exists_{X,Y} G)^{Tr2}$
2	$\exists_X\forall_Y G$	$\exists_X{:}\varnothing(\forall_Y G)^{Tr2}$
3	$\exists_X G_1\&\ldots\&\exists_X G_n\&A_1\ldots\&A_m$	$\exists_X{:}A_1\&\ldots\&A_m(G_1{}^{Tr2})^{\forall}, \ldots, (G_n{}^{Tr2})^{\forall}$
4	$\exists_X G_1\vee\ldots\vee\exists_X G_n\vee\neg A_1\ldots\vee\neg A_m$	$\exists_X{:}\varnothing\forall{:}A_1\&\ldots\&A_m(G_1{}^{Tr2})^{\exists}, \ldots, (G_n{}^{Tr2})^{\exists}$
5	$\exists_X\neg A$	$\exists_X{:}\varnothing(\forall{:}A)$
6	$\exists_X A$	$\exists_X{:}A$
7	$\forall_X\exists_Y G$	$\forall_X{:}\varnothing(\exists_Y G)^{Tr2}$
8	$\forall_X\forall_Y G$	$(\forall_{X,Y} G)^{Tr2}$
9	$\forall_X G_1\&\ldots\&\forall_X G_n\&A_1\ldots\&A_m$	$\forall_X{:}\varnothing\exists{:}A_1\&\ldots\&A_m(G_1{}^{Tr2})^{\forall}, \ldots, (G_n{}^{Tr2})^{\forall}$
10	$\forall_X G_1\vee\ldots\vee\forall_X G_n\vee\neg A_1\ldots\vee\neg A_m$	$\forall_X{:}A_1\&\ldots\&A_m(G_1{}^{Tr2})^{\exists}, \ldots, (G_n{}^{Tr2})^{\exists}$
11	$\forall_X\neg A$	$\forall_X{:}A$
12	$\forall_X A$	$\forall_X{:}\varnothing(\exists{:}A)$
13	$G_1\&\ldots\&G_n$	$\exists{:}\varnothing(G_1{}^{Tr2})^{\forall}, \ldots, (G_n{}^{Tr2})^{\forall}$
14	$G_1\vee\ldots\vee G_n$	$\forall{:}\varnothing(G_1{}^{Tr2})^{\exists}, \ldots, (G_n{}^{Tr2})^{\exists}$
15	$\neg A$	$\forall{:}A$
16	A	$\exists{:}A$

Если в некоторой ПОФ F

$$Q_X A\begin{cases} P_Y B\begin{cases} QC\begin{cases} P_{Z_1} D_1 - \Phi_1 \\ \cdots \\ P_{Zk} D_k - \Phi_k \end{cases} \\ \Psi \end{cases} \\ \Phi \end{cases}$$

$P, Q \in \{\forall, \exists\}$, $P \neq Q$, C, B – конъюнкты, удовлетворяющие условию $C \subseteq B$, то F эквивалентна F', имеющей следующий вид:

$$Q_X A\begin{cases} P_{Y,Z_1} B, D_1 - \Phi_1 \cup \Psi \\ \cdots \\ P_{Y,Zk} B, D_k - \Phi_k \cup \Psi \\ \Phi \end{cases}$$

Представленное правило нуждается в доказательстве.

Пусть конъюнкт C имеет вид $C^1 \& \ldots \& C^m$. Если $C \subseteq B$, то $B = B' \& C^1 \& \ldots \& C^m$, где B' – некоторый, возможно, пустой конъюнкт. Переведем ПОФ F в язык исчисления предикатов. В случае, если F начинается с квантора $\forall$:

$F = \forall_x A \rightarrow (\Phi \vee \exists_Y (\Psi \& B(\neg C \vee \exists_{z1}(D_1 \& \Phi_1) \vee \ldots \vee (D_k \& \Phi_k))))$.

При раскрытии скобок получаем дизъюнкцию конъюнкций, одна из которых

$$\Psi \& B \& \neg C = \left(\Psi \& B' \& C^1 \& \ldots C^m\right) \&$$

$$\& \left(\neg C^1 \vee \ldots \neg C^m\right) = False,$$ то есть является несущественной.

Поэтому $F=\forall_x A \rightarrow (\Phi \vee \exists_Y \exists_{Y1}(B \& D_1 \& \Psi \& \Phi_1) \vee \ldots \vee (B \& D_k \& \Psi \& \Phi_k))$, что при переводе в ПОФ-представление имеет вид, совпадающий с F'.

Если F начинается с квантора $\exists$, то после перевода в язык исчисления предикатов получаем: $F = \exists_X A \& \Phi \&$

$$\& \left(\forall_Y B \rightarrow \left(\Psi \vee C \& \forall_{Z_1}\left(D_1 \rightarrow \Phi_1\right) \& \ldots \& \forall_{Z_k}\left(D_k \rightarrow \Phi_k\right)\right)\right).$$

Распишем выражение в скобке:

$$\forall_Y B \rightarrow \left(\Psi \vee C \& \forall_{Z_1}(D_1 \rightarrow \Phi_1) \& \ldots \& \forall_{Z_k}(D_k \rightarrow \Phi_k)\right).$$

Устраняем импликацию и группируем дизъюнктивные элементы в отдельную скобку:

$$\forall_Y ((\neg B' \vee \neg C^1 \vee \ldots \vee \neg C^n \vee \Psi) \vee C^1 \& \ldots$$

$$\& C^n \& \forall_{Z_1}(D_1 \rightarrow \Phi_1) \& \ldots \& \forall_{Z_k}(D_k \rightarrow \Phi_k)).$$

Применяя закон дистрибутивности, получаем:

$$\forall_Y \begin{pmatrix} \left(\neg B' \vee \neg C^1 \vee \ldots \vee \neg C^n \vee \Psi \vee C^1\right) \& \ldots \\ \& \left(\neg B' \vee \neg C^1 \vee \ldots \vee \neg C^n \vee \Psi \vee C^n\right) \& \\ \& \left(\neg B' \vee \neg C^1 \vee \ldots \vee \neg C^n \vee \Psi \vee \forall_{Z_1}(D_1 \rightarrow \Phi_1)\right) \& \ldots \\ \& \left(\neg B' \vee \neg C^1 \vee \ldots \vee \neg C^n \vee \Psi \vee \forall_{Z_k}(D_k \rightarrow \Phi_k)\right) \end{pmatrix}.$$

В первых n скобках присутствует тавтология $\neg C_i \vee \neg C_i$, поэтому

$$\forall_Y \begin{pmatrix} True \& \ldots \& True \& \left(\neg B \vee \Psi \vee \forall_{Z_1}(D_1 \rightarrow \Phi_1)\right) \& \ldots \\ \& \left(\neg B \vee \Psi \vee \forall_{Z_k}(D_k \rightarrow \Phi_k)\right) \end{pmatrix} =$$

$$= \forall_Y \forall_{Z_1}\left(\neg(B \& D_1) \vee \Psi \vee \Phi_1\right) \& \ldots$$

$$\& \forall_Y \forall_{Z_k}\left(\neg(B \& D_k) \vee \Psi \vee \Phi_k\right).$$

То есть

$$F = \exists_X A \& \Phi \& \forall_Y \forall_{Z_1}\left((B \& D_1) \rightarrow (\Psi \vee \Phi_1)\right) \& \ldots$$

$$\& \forall_Y \forall_{Z_k}\left((B \& D_k) \rightarrow (\Psi \vee \Phi_k)\right),$$

что при переводе в ПОФ-представление имеет вид, совпадающий с F'.

Доказательство завершено.

В библиотеке TPTP, кроме задач в классическом первопорядковом представлении, достаточно большое количество задач представлено в виде множеств дизъюнктов, задачи приведены в скулемовской стандартной форме и восстановить их первоначальную формализацию проблематично. Для перевода задач, представленных в виде множеств дизъюнктов, в которых отсутствуют переменные, связанные кванторами существования, достаточно воспользоваться приведенной ниже формулой. Объединим дизъюнкты некоторого множества S в классы C^1, C^2, C^3, C^4, которые представляются следующим образом:

- $C^1 = \{C_1^1, \ldots, C_{n_1}^1\}$ – множество единичных основных (не содержащих переменных) положительных дизъюнктов;
- $C^2 = \{C_1^2, \ldots, C_{n_2}^2\}$ – множество положительных дизъюнктов, то есть $C_j^2 = \bigvee_{i=1}^{k_j} L_i^{2_j}$, $\forall j = \overline{1, n_2}$, где $L_i^{2_j}$ – положительные литеры, $k_j > 1$ – количество литер в соответствующих дизъюнктах, обозначим через X_j^2 множество переменных, встречающихся в дизъюнкте C_j^2;
- $C^3 = \{C_1^3, \ldots, C_{n_3}^3\}$ – множество отрицательных дизъюнктов, то есть $C_j^3 = \bigvee_{i=1}^{k_j} \neg L_i^{3_j}$, $\forall j = \overline{1, n_3}$, где $L_i^{3_j}$ – положительные литеры, $k_j > 1$ – количество литер в соответствующих дизъюнктах, обозначим через X_j^3 множество переменных, встречающихся в дизъюнкте C_j^3;
- $C^4 = \{C_1^4, \ldots, C_{n_4}^4\}$ – множество дизъюнктов, содержащих и положительные, и отрицательные литеры, то есть $C_j^4 = \bigvee_{i=1}^{m_j} \neg L_i^{4_j} \vee \bigvee_{i=m_{j+1}}^{k_j} L_i^{4_j}$,

$\forall j = \overline{1, n_4}$, где $L_i^{4_j}$ – положительные литеры, $m_j > 0$, $k_j > 0$ – количество отрицательных и положительных литер в соответствующих дизъюнктах, обозначим через X_j^4 множество переменных, встречающихся в дизъюнкте C_j^4.

ПОФ, соответствующая упорядоченному таким образом множеству дизъюнктов, будет иметь вид:

$$\exists C_1^1, \ldots, C_{n_1}^1 \begin{cases} \forall_{X_r^3} : L_1^{3_r}, \ldots, L_{k_r}^{3_r} \\ \forall_{X_s^4} : L_1^{4_s}, \ldots, L_{m_s}^{4_s} - \exists L_{m_s+1}^{4_s}, \ldots, L_{k_s}^{4_s} \\ \forall_{X_p^2} : \varnothing \begin{cases} \exists L_1^{2_p} \\ \cdots \\ \exists L_{k_p}^{2_p} \end{cases} \end{cases}$$

$\forall p = \overline{1, n_2}, \forall r = \overline{1, n_3}, \forall s = \overline{1, n_4}$. То есть итоговая формула будет содержать n_2 подформул, соответствующих дизъюнктам класса C^2, n_3 подформул класса C^3, n_4 подформул класса C^4.

Тестирование. Для систем АДТ важно время, затраченное на решение задачи. Поскольку разработанный транслятор (включая синтаксический анализатор) планируется использовать в системе АДТ для ПОФ-исчисления, эффективность трансляции также важна.

Эффективность работы программы в целом может зависеть от двух факторов: качества генерируемого кода компилятором (либо скорости исполнения интерпретатором) и качества используемых алгоритмов.

Важными характеристиками полученной в результате трансляции ПОФ являются количество узлов в древовидном представлении ПОФ и количество фиктивных кванторов. Поэтому процедура трансляции ПЛФ в ПОФ состоит из нескольких шагов, часть из которых направлены на уменьшение количества узлов и фиктивных кванторов. Исходная эвристическая структура ПЛФ сохраняется благодаря тому, что сохраняются кванторы ∀ и ∃ и их исходный порядок следования (в отличие от процедуры скулемизации, используемой для получения КНФ, устраняющей кванторы ∃ и вносящей в формулы дополнительные термы).

Алгоритм трансляции написан на языке Rust [10]. Тестирование разработанных алгоритмов производилось на задачах из библиотеки TPTP. Использовался компьютер MacBookPro с процессором 2,3 GHz Intel Core i7 и оперативной памятью 8 GB 1 600 MHz DDR3.

Всего было отобрано 6 817 задач, по которым собиралась статистика. Количество задач, для которых ПЛФ-представление содержит меньше узлов, чем ПОФ-представление, равно 1 169. В таблице 3 представлены 10 таких задач. Количество задач, для которых ПОФ-представление содержит меньше узлов, чем ПЛФ-представление, равно 5 648, 10 из них приведены в таблице 4. В таблице 5 показаны 5 наиболее трудоемких для трансляции задач.

Кроме того, не обнаружено никакой зависимости между рейтингом задачи и наличием в ней неограниченных переменных в ПОФ-представлении данной задачи. Этот факт говорит о том, что для систем АДТ, основанных на методе резолюций, не имеет значения, содержит ли формализация задачи неограниченные переменные. Однако для метода АДТ, основанного на ПОФ-исчислении, отсутствие неограниченных переменных заметно улучшает эффективность поиска логического вывода. Поэтому с помощью ПОФ-исчисления естественным об-

Таблица 3

Задачи, в которых узлов в ПОФ больше чем в 2 раза по сравнению с ПЛФ

Table 3

The tasks with twice more nodes in positively constructed formulas comparing with first-order logic formulas

Задача TPTP	Узлов в ПЛФ	Связок в ПЛФ	Узлов в ПОФ	Время трансляции (сек.)
HWV097+1.p	66 290	57 069	136 382	0.5405
LCL181+1.p	6	5	13	0.0014
SET047+1.p	8	4	23	0.0003
SET194+3.p	20	11	41	0.0004
SET577+3.p	28	17	57	0.0005
SET580+3.p	25	16	55	0.0004
SET624+3.p	23	14	47	0.0004
SET788+1.p	8	4	22	0.0003
SEU142+1.p	26	16	54	0.0005
SEU149+1.p	22	14	46	0.0001

Таблица 4

Задачи, когда узлов в ПЛФ больше чем в 5 раз по сравнению с ПОФ

Table 4

The tasks with five times more nodes in first-order logic formulas comparing with positively constructed formulas

Задача TPTP	Узлов в ПЛФ	Связок в ПЛФ	Узлов в ПОФ	Время трансляции (сек.)
AGT024+2.p	1105	1071	182	0.0147
AGT025+2.p	1105	1071	182	0.0125
GEO301+1.p	2603	2077	516	0.0149
GEO302+1.p	2604	2078	516	0.015
GRA026+1.p	66	60	12	0.0006
LCL648+1.001.p	16	12	2	0.0007
LCL649+1.001.p	17	13	3	0.0016

Таблица 5

Наиболее трудоемкие для трансляции задачи

Table 5

The most time-consuming translation tasks

Задача TPTP	Узлов в ПЛФ	Связок в ПЛФ	Узлов в ПОФ	Время трансляции (сек.)
HWV067+1.p	356879	356875	209006	25.6
HWV061+1.p	492486	492482	251387	20.6
HWV133+1.p	2141861	2000646	4174817	19.6
SEU418+4.p	831581	655928	391496	6.78
SEU419+4.p	831605	655943	391510	6.33

разом выделяется такой особый класс формул, вывод которых можно построить быстрее, чем методом резолюций.

Заключение

В работе приводится краткое описание языка ПОФ и рассматриваются вопросы их обработки перед запуском автоматических алгоритмов поиска вывода. Представлен метод преобразования ПЛФ в язык ПОФ в виде эффективных алгоритмов обработки первопорядковых формул.

Разработанные алгоритмы сохраняют исходную эвристическую структуру знаний, представленных формулами языка исчисления предикатов. Рассмотрены вопросы преобразования формул языка дизъюнктов в язык ПОФ, и предложен упрощенный алгоритм для трансляции таких задач.

Работа выполнялась при поддержке РФФИ, проект № 16-29-04238офи_м.

Литература

1. Vassiliev S.N. Machine synthesis of mathematical theorems. J. Logic Program., 1990, vol. 9, no. 2–3, pp. 235–266.

2. Васильев С.Н., Жерлов А.К., Федунов Е.А., Федосов Б.Е. Интеллектное управление динамическими системами. М.: Физматлит, 2000. 352 с.

3. Cherkashin E.A., Davydov A.V., Larionov A.A. Calculus of positively-constructed formulas, its features strategies and implementation. Proc. 36-th Intern. Convent. MIPRO 2013/CIS, Chroatia, Opatija, 2013, pp. 1289–1295.

4. Sutcliffe G. The TPTP problem library and associated infrastructure. The FOF and CNF parts, v3.5.0. J. Autom. Reason., 2009, vol. 43, no. 4, pp. 337–362. DOI: 10.1007/s10817-009-9143-8.

5. Kovács L., Voronkov A. First-order theorem proving and Vampire. Proc. 25th Conf. CAV, LNCS, 2013, vol. 8044, pp. 1–35. DOI: 10.1007/978-3-642-39799-8_1.

6. Itegulov D., Slaney J., Woltzenlogel Paleo B. Scavenger 0.1: A theorem prover based on conflict resolution. In: de Moura L. (eds.). Proc. 26th Conf. CADE, LNCS, 2017, vol. 10395, pp. 344–356. DOI: 10.1007/978-3-319-63046-5_21.

7. Baumgartner P. SMTtoTPTP – a converter for theorem proving formats. Proc. 25th Intern. Conf. CADE, LNCS, 2015, vol. 9195, pp. 285–294.

8. Черкашин Е.А. Программная система «КВАНТ/1» для автоматического доказательства теорем. Иркутск: Изд-во ИДСТУ СО РАН, 1999. 16 с.

9. Nagashima Y., Kumar R. A proof strategy language and proof script generation for Isabelle/HOL. Proc. 26th Conf. CADE, LNCS, 2017, vol. 10395, pp. 528–545. DOI: 10.1007/978-3-319-63046-5_32.

10. Rust Programming Language. URL: www.rust.org (дата обращения: 25.03.2019).

Software & Systems
DOI: 10.15827/0236-235X.128.556-564

Received 02.04.19
2019, vol. 32, no. 4, pp. 556–564

The method for translating first-order logic formulas into positively constructed formulas

A.V. Davydov [1], *Research Associate, artem@icc.ru*
A.A. Larionov [1], *Programmer, bootfrost@zoho.com*
E.A. Cherkashin [1], *Senior Researcher, eugeneai@icc.ru*

[1] *Matrosov Institute for System Dynamics and Control Theory of Siberian Branch of Russian Academy of Sciences, Irkutsk, 664033, Russia*

Abstract. The paper considers the logic calculus of positively constructed formulas (PCF calculus) and based on it automated theorem proving (ATP) method. The PCF calculus was developed and described as a first-order logic formalism in works of S.N. Vassilyev and A.K. Zherlov as a result of formalizing and solving problems of control theory. There are examples of describing and solving some control theory problems, effectively (from the point of view of the language expressiveness and the theorem proving means efficiency) solved using PCF calculus, for example, controlling a group of lifts; directing a telescope at the planet center, which is in an incomplete phase, and mobile robot control.

Comparing to the capabilities of other logical means for subject domain formalization and logic conclusion search, the PCF calculus have the advantage of the expressiveness combined with the compactness of knowledge representation, the natural parallelism of their processing, large block size and lower combinatorial complexity of conclusions, high compatibility with heuristics, and great capabilities for interactive proof. The selected class of formulas makes it possible to build constructive proofs. This class of formulas is much wider than the class of Horn clauses used in the Prolog. There are no restrictions in the logical formalization of the axiomatic base of the subject domain, and the target statement is a conjunction of queries (in terms of the Prolog).

To test the ATP software system (prover) based on the PCF calculus the authors used the TPTP (Thousands of Problems for Theorem Provers) library. The TPTP format has become a standard in the community that studies automated reasoning. There is a natural need for the developed prover to accept problems in this format as input. Thus, the problem of translating the first-order predicate logic formulas presented in the TPTP format to the POF format arises. This problem is nontrivial due to the special structure of the PCF calculus formulas.

The paper proposes a more efficient translation method (compared to the previously developed algorithm in the first implementation of the prover based on the PCF calculus) for the first-order predicate calculus language preserving the original heuristic knowledge structure, and its simplified version for the problems presented in language of clauses. The efficiency is a number of steps and the length of the obtained formulas. The proposed method was implemented as a software system – a language translator of first-order TPTP logic formulas to the PCF calculus language. The paper presents test results of the developed method, which imply that there is a certain class of first-order formulas that are not taken into account as special by existing ATP systems, while the PCF calculus has special strategies that increase the efficiency of the inference search for such class of formulas.

Keywords: mathematical logic, automated theorem proving, translation algorithms.

Acknowledgements. *The work has been supported by the RFBR, project no. 16-29-04238офи_м.*

References

1. Vassiliev S.N. Machine synthesis of mathematical theorems. *J. Logic Program*. 1990, vol. 9, no. 2–3, pp. 235–266.

2. Vassilyev S.N., Zherlov A.K., Fedunov E.A., Fedosov B.E. *Intelligent Control of Dynamic Systems*. Moscow, Fizmatlit Publ., 2000, 352 p.

3. Cherkashin E.A., Davydov A.V., Larionov A.A. Calculus of positively constructed formulas, its features: strategies and implementation. *36th Intern. Convent. MIPRO 2013/CIS*. 2013, Chroatia, Opatija, 2013, pp. 1289–1295.

4. Sutcliffe G. The TPTP problem library and associated infrastructure. The FOF and CNF parts, v3.5.0. *J. of Automated Reasoning*. 2009, vol. 43, no. 4, pp. 337–362. DOI: 10.1007/s10817-009-9143-8.

5. Kovács L., Voronkov A. First-order theorem proving and Vampire. *Proc. 25th Conf. CAV, LNCS*, 2013, vol. 8044, pp. 1–35. DOI: 10.1007/978-3-642-39799-8_1.

6. Itegulov D., Slaney J., Woltzenlogel Paleo B. Scavenger 0.1: A theorem prover based on conflict resolution. *Proc. 26th Conf. CADE, LNCS*. 2017, vol. 10395, pp. 344–356. DOI: 10.1007/978-3-319-63046-5_21.

7. Baumgartner P. SMTtoTPTP – a converter for theorem proving formats. *Proc. 25th Intern. Conf. CADE, LNCS*. 2015, vol. 9195, pp. 285–294.

8. Cherkashin E.A. *KVANT/1 Programm System for Automated Theorem Proving*. PhD Thesis. IDSTU SO RAN Publ., Irkutsk, 1999, 16 p.

9. Nagashima Y., Kumar R. A proof strategy language and proof script generation for Isabelle/HOL. *Proc. 26th Conf. CADE, LNCS*, 2017, vol. 10395, pp. 528–545. DOI: 10.1007/978-3-319-63046-5_32.

10. *Rust Programming Language*. Availavle at: www.rust.org (accessed March 25, 2019).

Для цитирования

Давыдов А.В., Ларионов А.А., Черкашин Е.А. Метод трансляции первопорядковых логических формул в позитивно-образованные формулы // Программные продукты и системы. 2019. Т. 32. № 4. С. 556–564. DOI: 10.15827/0236-235X.128.556-564.

For citation

Davydov A.V., Larionov A.A., Cherkashin E.A. The method for translating first-order logic formulas into positively constructed formulas. *Software & Systems*. 2019, vol. 32, no. 4, pp. 556–564 (in Russ.). DOI: 10.15827/0236-235X.128.556-564.