Модел. и анализ информ. систем. Т. **17**, № **4** (2010) 60–69

УДК 517.51+514.17

Об исчислении позитивно-образованных формул для автоматического доказательства теорем

Давыдов А.В.[1], Ларионов А.А.[2], Черкашин Е.А.[3]

Институт динамики систем и теории управления СО РАН
Институт математики, экономики и информатики ИГУ
Институт динамики систем и теории управления СО РАН

e-mail: artem@icc.ru

получена 18 октября 2010

Ключевые слова: автоматическое доказательство теорем, математическая логика, неклассические логики, искусственный интеллект

Рассматривается выразительный логический язык ***LF*** и основанное на нем исчисление. Формулы этого языка состоят из некоторых крупно-блочных структурных элементов, таких как ти́повые кванторы. Язык ***LF*** содержит всего два логических символа — $\forall$ и $\exists$, которые составляют множество логических связок языка. Рассматривается логическое исчисление ***JF*** и полные стратегии автоматического поиска выводов, основанные на единственном унарном правиле вывода. Это исчисление обладает рядом других особенностей, благодаря которым достигается уменьшение комбинаторной сложности при поиске выводов в сравнении с такими известными системами автоматического доказательства теорем (АДТ), как метод резолюций и генценовские исчисления. Рассмотрены вопросы об эффективной реализации нового исчисления в виде программной системы для автоматического доказательства теорем.

1. Определение языка *LF*

Рассматриваемый формализм представления и обработки знаний является дальнейшим развитием логического исчисления ***J*** позитивно-образованных формул (ПО-формул) (см. [1]).

Определение 1. *Пусть Con — множество всех конъюнктов. Будем предполагать, что конъюнкт есть либо конечное множество обычных атомарных формул (атомов) языка первопорядковой логики предикатов, либо* $\mathbf{F}$*, где* $\mathbf{F}$ *удовлетворяет свойству:* $A \subset \mathbf{F}$ *для каждого* $A \in Con$*. Пустой конъюнкт будем обозначать* $\mathbf{T}$*. Атомы любого конъюнкта (кроме* $\mathbf{T}$ *и* $\mathbf{F}$*) могут содержать индивидные переменные, индивидные константы и функциональные символы.*

[1]Работа выполнена при поддержке совместного интеграционного проекта №45 СО РАН.
[2]Работа выполнена при поддержке программы “Университетский кластер”.
[3]Работа поддержана грантами РФФИ 08-07-98005-р_сибирь_а, 0 9-07-12017-офи_м, 0-07-00051-а.

Замечание 1. *Конъюнкт понимается как конъюнкция входящих в него атомов. Поэтому естественно, что пустой конъюнкт логически эквивалентен* ***True****, т.е. тождественен истинному предикату, а конъюнкция бесконечного числа всевозможных атомов (не считающаяся конъюнктом, но допустимая в обычной абстракции актуальной, т.е. завершенной бесконечности) — тождественна ложному предикату* ***False****.*

Определение 2. *Синтаксис языка* ***LF*** *ПО-формул в нотации Бэкуса-Наура можно определить следующим образом:*
<pcf-formula> ::= “∀” “:” “**T**” <e-formula-list>
<t-v-cond> ::= <var-list> “:” <atom-list> | “:” <atom-list>
<e-node> ::= “∃” <t-v-cond>
<a-node> ::= “∀” <t-v-cond>
<e-formula> ::= <e-node> | <e-node> <a-formula> | <e-node> “{” <a-formula-list> “}”
<e-formula-list> ::= <a-formula> | <a-formula> “,” <a-formula-list>
<a-formula> ::= <a-node> <e-formula> | <a-node> “{” <e-formula-list> “}”
<a-formula-list> ::= <e-formula> | <e-formula> “,” <e-formula list>
<atom-list> ::= <atom> | <atom> “,” <atom-list>
<atom> ::= “**T**” | “**F**” | <upperalpha> | <upperalpha> “(” <term-list> “)”
<term-list> ::= <term> | <term> “,” <gterm-list>
<term> ::= <constant> | <variable> | <functor> “(” <term-list> “)”
<constant> ::= <functor>; <variable> ::= <upperalpha>;
<functor> ::= <loweralpha>
<var-list> ::= <variable> | “{” <variable> “}” | “{” <variable-list> “}”
<upperalpha> ::= “A” | “B” | “C” . . . ; <loweralpha> ::= “a” | “b” | “c” . . .

Теперь определим семантику языка ***LF***.

Определение 3. *Если* $A \in Con, A \notin \{\mathbf{F}, \mathbf{T}\}$*, то* $A^{\&} = \&\{\alpha{:}\alpha \in A\}$, $\mathbf{F}^{\&} = False$, $\mathbf{T}^{\&} = True$ *(пропозициональные константы). Пусть* $X = \{x_1, \ldots, x_m\}$*, тогда*

$$\begin{aligned}(\exists X{:}A\ \Phi)^{ип} &= \exists x_1 \ldots \exists x_m (A^{\&} \& (\Phi)^{ип}),\\ (\forall X{:}A\ \Psi)^{ип} &= \forall x_1 \ldots \forall x_m (A^{\&} \to (\Psi)^{ип}),\\ \textit{где}\ (\Phi)^{ип} &= \&\{(\alpha)^{ип} : \alpha \in \Phi\},\\ (\Psi)^{ип} &= \vee\{(\alpha)^{ип} : \alpha \in \Psi\}.\end{aligned}$$

Таким образом считается, что любая ПО-формула имеет структуру дерева (графа), где ветвление в ∀-узле (соответственно ∃-узле) означает дизъюнкцию (соответственно конъюнкцию). Для удобства и наглядности будем представлять ПО-формулы в виде древовидных структур. Например, формула

$$\forall\colon \mathbf{T}(\exists \bar{x}\colon A(\bar{x})(G_1, \ldots, G_k))$$

будет выглядеть так:

$$\forall\colon \mathbf{T} - \exists \bar{x}\colon A(\bar{x}) - \left\{ \begin{array}{l} G_1 \\ \ldots \\ G_k \end{array} \right.$$

здесь $G_i, i = \overline{1,k}$, – некоторые $\forall$-подформулы (поддеревья), фигурная скобка задает точку ветвления в структуре дерева.

Определение 4. *Любой узел вида $\exists X{:}A$, непосредственно следующий за корнем ПО-формулы, называется базой данной ПО-формулы (А будем называть базой фактов). Подформула, корень которой является базой, называется базовой подформулой. Пусть любой непосредственный потомок $\forall Y : B$ базы $\exists X : A$ называется вопросом к $\exists X{:}A$.*

Замечание 2. *В определении 3 семантика языка **LF** вводится на базе семантики языка исчисления предикатов. Т.е. семантика языка **LF** является семантикой трансформационного типа, задаваемая вышеприведенными правилами переписывания. Иначе говоря, интерпретация ПО-формул определяется через интерпретацию ее образа в ИП. Существуют простые, более или менее экономные, алгоритмы преобразования ПО-формул в язык ИП (см., например, [2]). Под экономностью понимается количество шагов алгоритма преобразования и длина получаемых ПО-формул (под длиной понимается количество кванторов и ветвлений в древовидном представлении). Следует отметить, что преобразование формул с языка исчисления предикатов в язык ПО-формул более трудоемко, чем преобразование по правилам, определенным выше.*

Пример 1. *Рассмотрим формулу языка исчисления предикатов:*

$$F = Man(Socrat)\&\neg Mortal(Socrat)\&\forall x(Man(x) \to Mortal(x))$$

*Образ F в языке **LF** согласно введенной грамматике:*

$$\forall\colon \boldsymbol{T} - \exists\colon MAN(socrat) \begin{cases} \forall X\colon MAN(X) & - \quad \exists\colon MORTAL(X) \\ \forall\colon MORTAL(socrat) & - \quad \exists\colon \boldsymbol{False} \end{cases}$$

Предложение 1. *[2] Язык **LF** ПО-формул является полным относительно выразительности языка логики предикатов первого порядка.*

Последнее предложение означает, что любую формулу языка ИП первого порядка можно представить эквивалентной ей в языке **LF** и наоборот. Следующая теорема устанавливает некоторую сравнительную сложность между представлением формул в языке **LF** и классическим в языке ИП первого порядка.

Теорема 1. *[2] $\forall k > 0$ существует последовательность булевских функций $f_1, \ldots, f_n$, такая что*

$$\mathcal{C}(n)_{ПО} \times k^{k^{\frac{n-1}{2}}} < \mathcal{C}(n)_{КНФ},$$

*где $\mathcal{C}(n)_{ПО}$ — сложность представления f_n в языке **LF**, а $\mathcal{C}(n)_{КНФ}$ — сложность представления f_n в конъюнктивной нормальной форме.*

2. Определения исчисления *JF*

В процессе доказательства некоторого утверждения $\mathcal{F}$ часто доказывают противоречивость его отрицания. Мы будет действовать так же.

Определение 5. *Будем говорить, что вопрос $\forall Y{:}B$ к базе $\exists X{:}A$ имеет ответ Θ тогда и только тогда, когда Θ — отображение (подстановка) $Y \to H^\infty$ и $B\Theta \subseteq A$, где H^∞ — эрбрановский универсум, основанный на константах и функциональных символах, встречающихся в $A \cup B$. Если $X = \varnothing$ и в $A \cup B$ нет констант и функциональных символов, то $H^\infty \stackrel{df}{=} \{a\}$, где a — некоторая новая константа.*

Теперь определим единственное правило вывода исчисления ***JF***.

Определение 6. *Если ПО-формула $\mathcal{F}$ имеет структуру $\forall{:}\mathbf{T}\{\Psi,\ \exists X{:}A\ \ \Phi\}$, где Ψ — список других базовых подформул (поддеревьев) формулы $\mathcal{F}$, а Φ содержит подформулу $\forall Y{:}B\ \{\exists Z_i{:}C_i\ \Psi_i\}_{i=\overline{1,k}}$, тогда результатом $\omega\mathcal{F}$ применения унарного правила вывода ω к вопросу $\forall Y{:}B$ с ответом $\Theta : Y \to H^\infty$ является следующая формула:*

$$\omega\mathcal{F} = \forall{:}\ \ \mathbf{T}\ \{\Psi, \{\exists X \cup Z_i{:}A \cup C_i\Theta\ \ \Phi \cup \Psi_i\Theta\}_{i=\overline{1,k}}\}.$$

После последующего переименования некоторых связанных переменных внутри каждой подформулы, выражение $\omega\mathcal{F}$ будет удовлетворять всем требованиям к ПО-формулам. Мы будем подразумевать это переименование каждый раз при применении правила ω. То же самое относится и к следующим упрощающим заменам:

— $\exists X{:}\ \mathbf{F}\ \ \Phi\ /\ \exists{:}\ \mathbf{F}$, т.е. $\exists X{:}\ \mathbf{F}\ \Phi$ заменяется на $\exists{:}\ \mathbf{F}$,

— $\forall{:}\ \mathbf{T}\ \{\Psi, \exists{:}\ \mathbf{F}\}/\ \forall{:}\ \mathbf{T}\ \Psi$ if $\Psi \neq \emptyset$.

Определение 7. *Любая конечная последовательность ПО-формул $\mathcal{F}, \omega\mathcal{F}, \omega^2\mathcal{F}, \ldots, \omega^n\mathcal{F}$, где $\omega^s\mathcal{F} = \omega(\omega^{s-1}\mathcal{F}), \omega^1 = \omega, \omega^n\mathcal{F} = \forall : \mathbf{T}\ \ \exists\ : \mathbf{F}$, называется выводом $\mathcal{F}$ в исчислении $\boldsymbol{JF}$ $=< \forall{:}\mathbf{T}\ \ \exists : \mathbf{F},\ \omega >$ (с аксиомой $\forall{:}\mathbf{T}\ \ \exists : \mathbf{F}$).*

Будем предполагать, что стратегия поиска вывода проверяет вопросы последовательно, без пропусков (повторяя проверку только после того, как закончатся вопросы), не допускает повторного применения ω к вопросу с одной и той же подстановкой (ответом) Θ (т.о., имеем *вопросно-ответную процедуру поиска вывода*).

Замечание 3. *Заметим, что поиск ответов Θ является трудоемкой и часто встречающейся задачей при поиске вывода. Как сказано выше, ответ Θ к вопросу $\forall Y{:}B$ существует тогда и только тогда, когда имеет место включение $B\Theta \subseteq A$, где A, B — некоторые множества атомов, A — база фактов. Таким образом, необходимо найти все удовлетворяющие данному включению подстановки Θ. Следовательно, мы имеем дело с классической задачей поиска поглощающих подстановок. Для того чтобы ее решить, существуют различные алгоритмы поглощения.*

Теорема 2. *[8] Правило вывода ω является эквивалентным преобразованием ПО-формул, т.е. если F – ПО-формула, то*

$$\vdash F^{ИП} \leftrightarrow (\omega F)^{ИП}.$$

Теорема 3. *[8] ПО-формула F противоречива, тогда и только тогда, когда она выводима в ***JF**.*

3. Особенности языка *LF* и исчисления *JF*

1. Хотя семантика ПО-формул очень прозрачная и определяется как обычная (классическая, теоретико-модельная) семантика соответствующих образов этих формул в исчислении предикатов, ПО-формулы довольно необычны по своей форме, а ПО-формализм в целом имеет ряд важных особенностей, расширяющих потенциал логического подхода к представлению и обработке знаний и определяющих его эффективность в приложениях. Рассмотрим некоторые особенности.

a) Любая формула, записанная в языке ***LF***, имеет *крупноблочную структуру*, все элементы которой суть только *позитивные кванторы* всеобщности и существования.

b) ПО-формула имеет *простую* и *регулярную структуру*, т.е. при считывании текста формулы она обладает в определенном смысле предсказуемостью структуры, ввиду поочередного появления в каждой ветви $\exists$- и $\forall$-узлов.

c) ПО-представление для первопорядковых формул *более компактно*, чем в языке дизъюнктов. Это же верно и для пропозициональных формул — в сравнении с представлениями в стандартных нормальных формах (дизъюнктивных и конъюнктивных).

d) Нет необходимости предварительно в исходной первопорядковой формуле удалять кванторы существования процедурой скулемизации (как это требует применение метода резолюций [3]), увеличивающей сложность термов и в целом формулы.

e) В языке ***LF*** исходная эвристическая структура знания *сохраняется лучше*, благодаря тому, что естественно-языковое представление знаний никогда не использует "теоретических" кванторов $\forall x$, $\exists x$ и, наоборот, широко задействует типовые кванторы $\forall x(A \to \sqcup)$, $\exists x(A\&\sqcup)$. Использование в ПО-формулах позитивных кванторов как частного случая типовых кванторов означает, что выражения A имеют очень простой (и эффективный для обработки ПО-формул) вид (конъюнкции атомов). ПО-структура особенно близка к исходной структуре знания, если оно записано как выражение, образованное из литералов (атомов и/или их отрицаний) с помощью позитивных кванторов и логических связок $\&, \vee$, т.к. в этом случае ПО-структура может отличаться от исходной просто появлением кванторов $\mathbf{K} : \mathbf{T}$ (вместо упомянутых связок $\&, \vee$) и/или заменой отрицаний атомов $\neg A$ подформулой $\forall : A \;\; \exists : \mathbf{F}$.

2. Теперь обратим внимание на важные преимущества исчисления ***JF***, которые появляются не только благодаря особенным свойствам нашего необычного языка, а также благодаря особым мощным свойствам механизма логического вывода. Рассмотрим эти преимущества.

a) В отличие от многих других известных логических исчислений, например, генценевского типа, наше исчисление ***JF*** имеет единственное правило вывода, благодаря чему размерность комбинаторного пространства поиска выводов уменьшается. Однако это не главное преимущество исчисления. В множество важных достоинств ***JF*** включается не только единственность правила вывода, но также его унарность (например, в сравнении с методом резолюций). Другое преимущество исчисления ***JF*** — очевидное с точки зрения уменьшения комбинаторики — это то, что правило вывода ω является крупно-блочным, так же как и сам язык ***LF***. Такое правило

позволяет дополнительно уменьшить сложность комбинаторного пространства (в сравнении с правилом резольвирования в методе резолюций, которое хотя тоже является единственным правилом, но бинарное и мелко-блочное).

b) Описанная техника вывода (опровержения) анализирует ближайшую окрестность корня ПО-формулы, т.е. только корень (базу) и непосредственно следующие узлы структуры (вопросы к базе). Это оказывается возможным, благодаря особенности (1a), и позволяет *сфокусировать внимание* на локальном фрагменте ПО-формулы без потери свойства полноты вывода.

c) Техника вывода может формулироваться в содержательных терминах вопросно-ответной процедуры вместо технических терминов формальной выводимости (т.е. в терминах логических связок, атомов и т.п.).

d) Благодаря особенностям (1a), (2b), (2c) техника вывода хорошо совместима с эвристиками конкретных приложений, а также с общими эвристиками управления выводом. Благодаря особенности (2a), процесс вывода состоит из *крупно-блочных шагов*, хорошо *наблюдаем* и *управляем*.

e) Техника вывода допускает естественный ИЛИ-параллелизм, т.к. опровержения по ветвям ПО-формулы осуществляются независимо друг от друга.

f) Базовое исчисление ***JF*** обладает свойством простой и сильной модифицируемости: простым ограничением применения правила вывода и без изменения аксиомы $\exists : \mathbf{F}$ может сильно варьироваться семантика логики ***JF*** (конструктивность, немонотонность и др.), что вместе с другими особенностями (и, в первую очередь, (2d)) весьма важно для приложений.

4. Об эффективной реализации исчисления *JF* в виде программной системы для АДТ

В данном разделе предлагается методика повышения эффективности обработки ПО-формул, базирующаяся на использовании специализированных структур данных, представляющих ПО-формулу в памяти компьютера и учитывающих перечисленные выше особенности ПО-формул.

При построении программной системы АДТ нами использовались следующие подходы.

1. Индексирование термов

Количество шагов логического вывода (ЛВ) может быть сколь угодно большим, а значит, объем формулы (общее количество атомов в формуле) может значительно увеличиваться за счёт добавления новых атомов в базу, расщепления формулы из-за дизъюнктивного ветвления, добавления новых вопросов. Такое увеличение формулы затрудняет поиск ответных подстановок, в связи с тем что необходимо учитывать всё большее количество атомов при поиске ответа.

Данная проблема присуща и другим методам АДТ, и на сегодняшний день существует множество хорошо известных методов работы с большими множествами

термов (атомов, формул, подстановок), например: индексирование путями [5], дерево сравнения (discrimination tree) [7], дерево подстановок (substitution tree indexing) [4] и т.п.

Каждый из методов имеет свои преимущества. Например, дерево подстановок требует меньше времени для поиска заданного атома, однако вставка нового элемента в индекс намного сложнее и медленнее. Нами выбран метод индексирования путями, который довольно быстро позволяет добавлять и извлекать заданные атомы, однако при этом затрачивает больше памяти, чем другие методы.

Кратко суть метода заключается в следующем. Каждый атом (или терм) представляется как список путей до каждого подтерма. Например, атом A(a,f(x,b)) представляется так: A, A1a, A2f, A2f1x, A2f2b. В каждом пути хранится указатель на соответствующий атом. Поиск примеров в множестве атомов для заданного атома заключается в пересечении всех множеств атомов, на которые указывает каждый из путей заданного атома. Пути выбираются с точностью до переменной.

Покажем данную процедуру на примере. Пусть дан атом A(a,f(x,b)) и множество атомов {A(a,f(c,b)), A(a,f(b,e)),A(a,f(k,b)), A(b,f(e,b))}. Любой атом, являющийся основным примером для заданного, содержит в своём списке путей следующие пути: A1a, A2f2b. Таким образом, для поиска основных примеров заданного атома необходимо найти пересечение множеств атомов, на которые указывают пути. В частности путь A1a указывает на множество {A(a,f(c,b)), A(a,f(b,e)),A(a,f(k,b))}, а путь A2f2b на {A(a,f(c,b)), A(a,f(k,b)), A(b,f(e,b))}. Пересечение этих множеств есть множество {A(a,f(c,b)),A(a,f(k,b))}, это и есть множество примеров для атома A(a,f(x,b)). Подробную информацию о методе индексирования путями можно найти, например, в [7].

Ответные подстановки находятся следующим образом. Последовательно просматриваются атомы конъюнкта вопроса. Для текущего атома находятся все примеры из соответствующей базы. По данным примерам восстанавливаются подстановки. Далее подстановки применяются к ещё непросмотренным атомам конъюнкта вопроса. И процедура повторяется. При этом на каждом шаге найденные подстановки соединяются с предыдущими. Если удалось пройти все атомы конъюнкта вопроса, то текущее соединение подстановок и есть ответ. Неудача происходит в случае, если на определенном шаге в базе не найдется ни одного примера для атома.

Наша реализация метода идейно не отличается от авторского, однако имеются некоторые изменения, связанные с особенностями ЛВ в исчислении ПО-формул.

2. Разделение данных

Формула может содержать много повторяющихся данных, копирование и хранение которых отнимает лишние ресурсы памяти и процессора.

Нами выделено два класса разделения данных: агрессивное и мягкое.

Агрессивное разделение заключается в том, что разделяется большинство общих подтермов атомов. Например, атомы A(b,f(e)), B(f(e),c), C(g(m,f(e)), k) содержат одинаковые подтермы f(e). Агрессивное разделение памяти заключается в том, что в каждом из перечисленных атомов используется один и тот же участок памяти

для обозначения терма f(e).

Мягкое разделение заключается в следующем: такое явление, как дизъюнктивное ветвление, приводит к расщеплению формулы и копированию бóльшей её части. Скопированная информация является избыточной, и копирование требует некоторого процессорного времени. Размер формулы, в общем случае, растет в ходе ЛВ. Разумно не копировать соответствующие поддеревья, а разделять эти данные, т.е. в расщепленной формуле должно быть указано, что данная часть является общей для всех подформул. Более подробно см. [6].

Мягкое разделение позволяет экономить как процессорное время, так и значительные объемы памяти (при наличии дизъюнктивного ветвления). Агрессивное разделение позволяет экономить ещё больше памяти, однако при этом затрачивает больше процессорного времени. Поэтому использование агрессивного разделения зависит от выбранной стратегии вывода.

3. Дерево состояний вывода

Введём понятие дерева состояний вывода, корнем которого является исходная формула, а каждые непосредственные наследники являются формулами, представляющими собой новую добавляемую информацию (результаты расщепления, добавление новых вопросов, фактов).

В данном случае появляется проблема для индексирования путями, т.к. всё множество базовых атомов имеет древовидную структуру, и это ограничивает возможность нахождения некоторых атомов в одной базе, т.е. при выборке атомов (например, поиск примеров) из множества могут появиться атомы из другой базы, что неправильно. Это решается следующим образом: при обработке текущего состояния формулы принимаются во внимание только те атомы, основные пути (путь в дереве от заданного узла до корня) которых в дереве состояний являются подпутями основного пути, определяющего нужную базу. На рисунке 1 представлен пример дерева состояний.

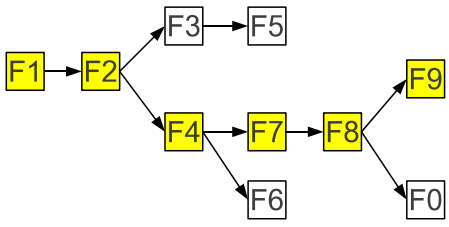

Рис. 1.

Цветом выделен основной путь для состояния F9 в дереве состояний. Каждый путь представляет собой полную базовую подформулу. Каждая базовая подформула определяется основным путем от узла к корню в дереве состояний, таким образом, количество листьев есть количество баз в формуле на текущий момент. На данном рисунке изображена формула с четырьмя базами, которые определяются путями: F5, F3, F2, F1; F9,F8,F7,F4,F2,F1; F0,F8,F7,F4,F2,F1; F6,F4,F2,F1. Кроме

того, по данному дереву состояний можно определить, что в ходе ЛВ трижды происходило расщепление формулы — после состояния F2, после состояния F4 и после состояния F8. F1 - исходная формула. Разделение данных определяется общими подпутями основных путей для заданных узлов, например, базовые подформулы, представляемые F3 и F4, разделяют общие данные – F2 и F1.

4. Дальнейшие исследования

Дальнейшие исследования связаны с разработкой методики управления процессом поиска логического вывода (модификации базовой стратегии). Как уже говорилось, одной из сильных сторон метода ПО-формул является свойство простой и сильной модифицируемости (способность к простой модификации базовой стратегии ЛВ). Разрабатываются достаточно общие типы эвристик, которые в виде вспомогательных алгоритмов встраиваются в алгоритм поиска ЛВ. В частности, применением более специализированных эвристик для конкретных задач можно значительно повышать эффективность ЛВ. Для описания таких эвристик предполагается разработка специального языка, который позволяет преобразовывать элементы ПО-формулы, интерпретируя их как содержательную информацию о задаче.

Список литературы

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

2. *Жерлов А.К., Васильев С.Н. Исчисления позитивно-образованных формул // Актуальные проблемы информатики, прикладной математики и механики: Сб. научных трудов. Красноярск, 1996. Ч. 1. С. 44–56.*

3. *J.A. Robinson. A Machine-Oriented Logic Based on Resolution Principle // Journal of ACM. 1965. January. P. 23–41.*

4. *Peter Graf. Substitution Tree Indexing // Proceedings of the 6th International Conference on Rewriting Techniques and Applications. 1995. P. 117–131.*

5. *Stickel M. The path-indexing method for indexing terms // Technical Note 473, Artificial Intelligence Center, SRI International, RAVENSWOOD AVE., MENLO PARK, CA 94025*

6. *Черкашин Е.А. Разделяемые структуры данных в системе автоматического доказательства теорем КВАНТ/3 // Вычислительные технологии. 2008. Т. 13. С. 102-107.*

7. *McCune W.W. Experiments with Discrimination-Tree indexing and Path Indexing for Term Retrieval // Journal of Automated Reasoning. 1992. V. 9(2). P. 147–167.*

8. *Давыдов А.В. Исчисление позитивно-образованных формул с функциональными символами // Прикладные алгоритмы в дискретном анализе: сб. науч. тр.*

/ Под ред. д-ра физ.-мат. наук, проф. Ю.Д. Королькова. Иркутск : Изд-во Иркут. гос. Ун-та, 2008. 157 с. (Дискретный анализ и информатика, Вып. 2). С. 23–49.

On the calculus of positively constructed formulas for authomated theorem proving

Davydov A.V., Larionov A.A., Cherkashin E.A.

Keywords: automated theorem proving, mathematical logic, nonclassical logics, artificial intelligence

The paper deals with an expressive logic language ***LF*** and its calculus. Formulas of this language consist of some large-block structural elements, such as type quantifiers. The language ***LF*** contains only two logic symbols — $\forall$ and $\exists$, which form the set of logic connectives of the language. A logic calculus ***JF*** and complete strategy for automated proof search based on a single unary rule of inference are considered. This calculus has a number of other features that lead to the reduction of combinatorial complexity of finding the deductions in comparison with the known systems for automated theorem proving as the resolution method and Genzen calculus. Problems of effective implementation of ***JF*** as a program system for automated theorem proving are considered.

Сведения об авторах:
Давыдов Артем Васильевич,
Институт динамики систем и теории управления СО РАН, г. Иркутск,
научный сотрудник;
Ларионов Александр Александрович,
Институт математики, экономики и информатики ИГУ, г. Иркутск, аспирант;
Черкашин Евгений Александрович,
Институт динамики систем и теории управления СО РАН, г. Иркутск,
канд. техн. наук, зав. лаб.