УДК 510.649
doi: 10.18101/2304-5728-2016-2-55-76

© Л. Н. Федорченко

Теория типов, компьютерная алгебра, поддержка доказательств и грамматики

Данная статья посвящена краткому обзору и рассмотрению основных направлений исследований, связанных с теорией типов и её приложениями. Даётся характеристика универсальных систем аналитических вычислений (систем компьютерной алгебры), а также систем поддержки аналитических вычислений и возможных доказательств истинности выводов (Proof Assistants). Рассмотрено применение типов в современной информатике, прежде всего, в *функциональном и объектно-ориентированном программировании.* Перечислены основные научные коллективы, ведущие исследования и разработки программных систем в данном направлении.

Ключевые слова: типы, подтипы, формальные системы, системы поддержки доказательств, трансформации на графах, грамматики.

© L. N. Fedorchenko

Type Theory, Computer Algebra, Proof Assistants and Grammars

This article provides a brief overview and review of the main areas of research related to type theory and their applications. A description of universal systems of analytical calculations (Computer Algebra), as well as support systems of analytical calculations (Proof Assistants) are given. The application types in Computer Science, above all, functional and object-oriented programming has been discussed. Specialized the main research teams which are working out research and development of software systems in this area.

Keywords: types, subtypes, formal systems, proof checking, graph transformations, grammars.

Введение

Попытки использовать ЭВМ для реализации символьных алгоритмов делались практически с самых первых шагов развития вычислительной техники [1]. За последние три десятилетия символьные вычисления или аналитические вычисления (в зарубежной литературе – Computer Algebra (CA)) стали одной из наиболее значительных областей информатики. Бурное развитие компьютерной алгебры объясняется двумя основными причинами.

Во-первых, это широкий интерес, который проявляется к символьным вычислениям со стороны специалистов самых разных областей знания.

Системы компьютерной алгебры предоставляют богатый инструментарий, который используется как в теоретических исследованиях по физике, химии, генетике, медицине, так и в чисто технических приложениях.

Во-вторых, движущим моментом развития СА является тот факт, что исследования нечисловых алгоритмов алгебры и математического анализа для их реализации на ЭВМ представляют самостоятельный интерес и приводят к развитию некоторых областей математики в самых неожиданных направлениях.

Наиболее актуальным с практической точки зрения, вероятно, следует признать направление развития СА, связанное с разработкой и совершенствованием *универсальных систем аналитических вычислений*. Среди них назовем наиболее известные – MACSYMA, REDUCE, Maple, Mathematica и другие. Их использование позволило в последние годы решить ряд ранее нерешенных теоретических и практических задач. Среди областей, где применение систем СА оказалось наиболее эффективным, следует отметить дифференциальные уравнения, общую теорию относительности, физику высоких энергий, небесную механику и др. Однако, отсутствие "встроенного" механизма доказательства, подтверждающего правильность аналитических преобразований в СА, осуществляемых системой вычислений, является существенным недостатком этих систем, особенно если принять во внимание распространенность ошибок в алгоритмах символьных вычислений, о которой можно судить как по публикациям в научной литературе [6–9], так и по многочисленным дискуссиям пользователей, доступным в Internet. Поэтому в последние годы различные математические коллективы направили свои усилия на создание программного обеспечения – *систем поддержки аналитических вычислений* и возможных доказательств истинности выводов (Proof Assistants – PA). Такие системы либо встраиваются в уже существующие СА, либо представляют собой автономные программы со своими методами представления данных весьма специальной структуры, языками, позволяющие манипулировать с ними, и разнообразными библиотеками эффективных функций для выполнения базисных операций.

Таким образом, в настоящее время параллельно с системами СА активно развиваются системы компьютерной поддержки доказательства (PA), основанных на системах формальной, в основном, конструктивной или интуиционистской логики. Последнее связано с тем фактом, что только для этих систем, в отличие от систем классической логики, существуют естественные алгоритмы извлечения программ из доказательства (т. е. нахождения его "вычислительного содержания"). Именно благодаря активному развитию PA в математической логике получило большое развитие направление – теория типов (Type Theory), которое выразилось в новой логической систематизации математики и в новых приложениях логики, системах поддержки доказательства теорем и в системах аналитических вычислений.

Высокий уровень развития, достигнутый теорией типов, существенно

помогает процессу формализации математики, развитию теории символьных вычислений, расширяет возможности применения компьютерной алгебры за счет сочетания систем CA и PA.

В настоящее время, исследования в этом направлении ведутся в ряде научных центров. Прежде всего, это многочисленные центры, входящие в проект "Types" в рамках Европейского Сообщества: INRIA (Франция), Даремский университет (Великобритания), университет Чалмерс (Chalmers) (Швеция) и ряд других. Значительную роль играет европейская сеть CALCULEMUS [6, 8, 9]. К этому же направлению относятся исследования в IRIT (Тулуза, Франция) и СПИИРАН (Санкт-Петербург), осуществляемые при поддержке Франко-Русского Центра им. А.М.Ляпунова, а также CNRS и РФФИ (проекты TTCA, ISOT и SynGT [42]). Данная статья посвящена краткому обзору и рассмотрению основных направлений этих исследований, связанных с теорией типов.

1. Что такое «теория типов»?

Прежде чем рассматривать понятие теории типов, напомним основные этапы развития аксиоматики в математике вообще и в формальной логике в частности. Как известно, в аксиоматическом построении математической теории предварительно выбирается некоторая система неопределяемых понятий и отношения между ними. Эти понятия и отношения называются основными. Далее без доказательства принимаются основные положения рассматриваемой теории – аксиомы. Все дальнейшее содержание теории выводится логически из аксиом. Впервые аксиоматическое построение математической теории было предпринято Евклидом в «Началах» в построении геометрии. И хотя истинность всех положений этой теории не вызывала сомнений, в доказательстве теорем используются нигде явно не сформулированные положения, которые считаются очевидными. Такой подход к аксиоматике в теории оставался единственным до XIX века, когда Н.И. Лобачевский впервые в явном виде высказал убеждение в невозможности доказательства пятого постулата Евклида. Так возникли и были решены в работах Лобачевского и позже в работах немецкого математика Ф.Клейна (1849–1925), (доказавшего непротиворечивость геометрии Лобачевского), впервые в истории математики проблемы невозможности доказательства и непротиворечивости в аксиоматической теории. Непротиворечивость аксиоматической теории означает, что из данной системы аксиом нельзя логическим путем вывести два противоречащих друг другу утверждения. Доказательство непротиворечивости аксиоматических теорий можно осуществить различными методами. Одним из них является метод моделирования (интерпретаций). Здесь в качестве основных понятий и отношений выбираются элементы некоторого множества и отношения между ними, а затем проверяется, будут ли выполняться для выбранных понятий и отношений аксиомы данной теории, то есть, строится модель для данной теории. Ясно, что метод моделирования сводит вопрос о непротиворечивости одной теории к проблеме непроти-

воречивости другой теории. Большинство интерпретаций для математических теорий строится на базе теории множеств, в связи с этим важно доказать непротиворечивость теории множеств.

Как математический термин «теория типов» возник в конце XIX века в работе Бертрана Рассела ("Principia Mathematica" [2]) в связи с открытием так называемого «парадокса Рассела», когда в теории множеств были обнаружены противоречия. Ярким примером такого противоречия является парадокс Б.Рассела. Разобьем все мыслимые множества на два класса. Назовем множество «нормальным», если оно не содержит себя в качестве своего элемента и «ненормальным» в противном случае. Пусть L – множество всех «нормальных» множеств. К какому классу относится множество L? Если L – «нормальное» множество, то $L \in L$, то есть содержится в классе «нормальных» множеств, но тогда оно содержит себя в качестве своего элемента, и поэтому «ненормально». Если L – «ненормальное» множество, то $L \notin L$, то есть не содержится среди «нормальных» множеств, но тогда L не содержит себя в качестве своего элемента, и поэтому оно «нормально». Таким образом, понятие «нормального» множества приводит к противоречию.

Защищая логицизм в математике (точка зрения, что математика сводима к формальной логике), Рассел предложил ввести понятие *тип математического объекта*. Чтобы исправить ситуацию с Парадоксом, он вводит иерархию типов математических объектов (в его варианте – «базовые объекты» имеют уровень 0, множества базовых объектов – уровень 1, множества множеств – уровень 2 и т.д.). Это – один из источников теории типов [2].

Дальнейшее развитие теория типов приобрела в работе Гёделя (Gődel) о функционалах конечных типов, в которой также вводится иерархия объектов и типов (базовый уровень 0 – это все натуральные числа, уровень 1 – это функции $0 \to 0$ и т.д.) [3].

Другие методы обоснования математики были развиты Д. Гильбертом (1862–1943) и его учениками. Они основываются на построении математических теорий как синтаксических теорий, в которых аксиомы записываются формулами в некотором алфавите и точно указываются правила вывода одних формул из других, то есть в теорию как составная часть входит математическая логика. Таким образом, математическая теория, непротиворечивость которой требовалось доказать, стала предметом другой математической теории, которую Гильберт назвал метаматематикой, или теорией доказательств. В связи с этим возникает задача построения синтаксической, то есть формализованной аксиоматической теории самой математической логики. Выбирая по-разному системы аксиом и правила вывода одних формул из других, получают различные синтаксические логические теории. Каждую из них называют логическим исчислением. Самым простым является исчисление высказываний, моделью для которого служит алгебра логики высказываний. В этой логике не рассматривается структура и тем более содержание высказываний (здесь объекты – выска-

зывания, принимают значение истина и ложь). В то же время и в науке, и в практике используются заключения, зависящие как от структуры, так и от содержания используемых высказываний. Возникает необходимость в расширении логики высказываний до такой логики, где можно исследовать структуру высказываний. Такой логической системой является логика предикатов, содержащая всю логику высказываний в качестве своей части. Предикаты, входящие в систему, могут быть первого порядка и высших порядков в зависимости от области их определения. Типизацию математических объектов вводят в основном в исчисление предикатов высших порядков.

2. Для чего нужна типизация?

В известной книге Гради Буча (Booch) [15] тип определяется следующим образом:

«…*тип (type)* – это определение области допустимых значений, которые может принимать объект, и множества операций, которые могут выполняться над объектом, а *типизация (typing)* – это механизм, препятствующий замене объектов одного типа на другой или, в крайнем случае, жестко ограничивающий такую замену».

Введенная типизация нужна для того, чтобы избежать парадоксов. Без типовое λ-исчисление позволяет моделировать любые частично-рекурсивные функции. Процесс вычислений моделируется процедурой нормализации λ-термов, однако, если нет типов, то по терму невозможно проверить, заканчивается ли этот процесс (алгоритмическая неразрешимость). В программировании типы нужны, чтобы избавиться от наиболее часто возникающих, базовых ошибок, например, применение функции к аргументу, не принадлежащему области определения.

С философской точки зрения можно сказать, что введение типов усиливает различие между множеством (как абстрактной совокупностью) и его элементом. Тип может иметь элементы, но не равняется их совокупности. Например, типы без элементов (пустые типы) необязательно равны между собой. Это все позволяет избежать грубых ошибок в программировании, так как позволяет учесть динамический элемент (т.е. тип как бы соотносится с его «потенциальными» элементами). Такое понятие о типе ближе к древнему «наивному» пониманию типов (видов, родов), какое мы находим у Аристотеля.

Понятие теории типов, рассматриваемое ниже, является более узким, чем типизация в программировании. Теория типов представляет в этом смысле современное развитие подхода, близкого к логицизму Бертрана Рассела. В качестве основных промежуточных этапов можно указать на результаты, содержащиеся в работе Гёделя (K. Gödel) [3] и работе Мартин-Лёфа (Per Martin-Löf) [4] по интуиционистской теории типов, в которой он ввел понятие логической структуры (Logical Framework, LF), играющее большую роль в современной теории типов.

Прежде чем рассматривать современную теорию типов и её примене-

ния, остановимся на направлениях в современной информатике, где дисциплина работы с типами достигла высокого уровня: это *функциональное и объектно-ориентированное программирование.*

Функциональным программированием принято называть стиль в программировании, который делает акцент скорее на вычислении выражений, чем на исполнении команд. Выражения в этих языках получаются применением функции к базовым значениям и их комбинацией. Функциональный язык поддерживает программирование в этом стиле.

Следуя определению Хенка (Henk)1, λ-исчисление – это формальный язык для выражения произвольных вычислимых функций. В комбинации с типами он позволяет с одной стороны описывать функциональные программы, а с другой – дает средства для описания математических доказательств. Функциональные программы состоят

- из модулей, которые, по сути, являются математическими функциями, (это следует из определения λ-исчисления Чёрчем (Church)). Примерами являются программы на языке ЛИСП и его более поздней модификации, языке Scheme;
- и из типов для этих модулей. Так что только допустимые комбинации модулей имеют вычислимое значение, (это также основано на концепции типа для λ-исчисления Чёрча). Впервые такая концепция типа была реализована в языке ML.

Остаётся добавить, что функции характеризуются типами, состоящими из области определения и области значений, которые в свою очередь сами могут быть сложными типами. В связи с этим, в компиляторах для функциональных языков большая роль отводится проверке корректности типов. В большинстве случаев имеются также механизмы решения уравнений между типами, содержащими переменные, и вывода типа составного выражения.

Концепция типа и объекта последовательно развивается и успешно используется в современных языках программирования, таких как C++ и Java, принимающих стиль объектно-ориентированного программирования. Этот стиль характеризуется следующими основными принципами:

данные и процедуры объединяются в программные объекты;

- для связи объектов используется механизм посылки сообщения;
- объекты с похожими свойствами объединяются в типы (классы);
- объекты наследуют свойства (типы) других объектов через иерархию типов.

Появляются объектно-ориентированные системы, обладающие свойствами, в основе которых лежит понятие об объекте определенного типа и его области действия (применения). Такими свойствами являются:

[1] 1. http://www.cs.kun.nl/~henk/lc.html

- абстрактные типы данных – объединение данных и операций для описания новых типов, позволяющие использовать новые типы наравне с уже существующими;
- инкапсуляция (сокрытие реализации) – данные и процедуры объекта скрываются от пользователя, связь с объектом ограничивается набором сообщений, которые объект «понимает»;
- полиморфизм (многозначность сообщений) – одинаковые сообщения по-разному понимаются разными объектами, в зависимости от принадлежности объекта тому или иному типу (классу);
- динамическое связывание – значение имени (область памяти для данных или текст программы для процедур) становится известным только во время выполнения программы;
- наследование – позволяет при создании новых объектов использовать свойства уже существующих объектов, описывая только новые добавляемые свойства.

Все эти свойства, напрямую связанные с основополагающими понятиями типа и объекта. Они усилили выразительные способности как универсальных языков программирования (линии C++), так и языков, используемых в системах поддержки доказательств и системах аналитических вычислений.

3. Системы поддержки доказательства и теория типов

Происхождение систем поддержки доказательств (Proof Assistants, PA), которое привело к интерактивным системам поддержки в современном смысле, обычно ведется от проекта "AUTOMATH" De Bruijna [16]. Хотя ленинградская школа математиков вела работы по логическому поиску вывода еще в начале 60-х годов. Первым сообщением перед публикой о постановках задачи поиска вывода был доклад Н.А. Шанина на IV Всесоюзном математическом съезде в июне 1961 [1]. В августе этого же года в ЛОМИ начала работать группа математической логики (группа ТРЭПЛО). Основная задача группы – построение алгоритма поиска естественного логического вывода. Была разработана компьютерная программа поиска вывода в исчислении высказываний. Теоретические основы программы и некоторые результаты экспериментов докладывались на I Всесоюзном симпозиуме по машинному поиску логического вывода (Тракай, июль 1964 г.) [31] и опубликованы в брошюре [32]. Вторая программа была посвящена поиску в исчислении предикатов. При ее написании за основу был взят обратный метод С.Ю. Маслова [33]. Схема программы и основные идеи впервые были доложены на II Всесоюзном симпозиуме по машинному поиску логического вывода (Тракай, август 1969 г.) и опубликованы в статье [34].

Идея интерактивного (человеко-машинный диалог) доказательства была впервые рассмотрена Милнером в [17]. Нововведением было использование механизма типизации в функциональном языке ML для построения доказательств. Эти идеи были развиты Паулсоном [18] для LCF и Петер-

соном [19] для теории типов Мартин Лёфа. Затем Кокан и Юйет (Coquand & Huet) [20] добавили средства для непредикативных определений к языку AUTOMATH и построили интерактивную систему проверки доказательств. На базе этих исследований была разработана система NuPRL [21] – первая крупномасштабная система для построения доказательств на базе теории типов.

Имеется большое количество логик, представляющих интерес для информатики. Разработка программной системы для интерактивной проверки доказательств в любой из этих логик – непростая задача. На уровне абстрактного синтаксиса должна поддерживаться *схема формул*, *термов и правил*, операторы связывания переменных и операция подстановки. На уровне формальных доказательств должно обеспечиваться представление доказательств, а также механизмы, связанные с проверкой корректности и построением доказательств. Сюда следует добавить получение конкретных применений из схем и проверку условий применимости, если таковые имеются. Наконец, необходимы механизмы поддержки этапов автоматического поиска вывода (тактики поиска, унификации и т.д.).

Необходимость экономии усилий привела к появлению понятия логической структуры (Logical Framework, LF). Цель использования LF состоит в унифицированном представлении широких классов логик и логико-математической теории. LF объединяет в себе формальную систему, дающую формальные средства для представления теории, а также неформальный метод для нахождения такого представления. Существующие в настоящее время LF используют понятие *зависимых типов* (допускается параметризация типов объектными выражениями). Обычно имеется три уровня выражений: *объекты*, *типы*, классифицирующие объектные выражения, и *роды* (kind), классифицирующие семейства типов. Используется также понятие “равенства по определению” между синтаксическими выражениями (учитывающее, например, бета-конверсию, известную из λ-исчисления).

Конкретная логико-математическая система определяется в рамках LF путем декларации конечного множества констант (которым присваиваются типы и/или роды). Эти константы представляют собой синтаксис этой конкретной системы, её базовые аксиомы и схему правил. Сама LF таким образом играет роль метаязыка. Так, например, декларации могут иметь вид

Prop:Type (логические утверждения конкретной системы образуют тип),

Prf: Prop -->Type (*Prf* – от слова “доказательство” – это функция, определяющая семейство типов, параметризованных элементами типа *Prop*. Здесь *Prf(A):Type* является типом доказательств *A*, где *A: Prop* – утверждение),

Nat: Type, 0*: Nat, succ: Nat-->Nat* (определения типа натуральных чисел),

n: Nat |- *vect(n):Type* (определение типа n-мерных векторов для каждого *n*).

Можно отметить, что сами доказательства включаются в формализм теории типов как объекты некоторого типа. Этот подход естественным образом связан с идеей соответствия доказательств и программ: существуют алгоритмы для перевода доказательств в теории типов в соответствующую программу. В этом смысле, большинство ныне существующих систем компьютерной поддержки доказательства (Proof Assistant) основано на теории типов.

Сами LF (их имеется несколько) обычно являются довольно слабыми логическими системами (для того, чтобы облегчить синтаксический анализ и построение доказательств в метатеории). Однако использование констант, объявляемых в LF (например, кванторы логики высших порядков которые могут декларироваться как константы LF), позволяет представлять выводы сколь угодно мощных логических и вычислительных систем. Ряд современных PA основан на использовании LF (см. ниже табл. 1, 2).

Таблица 1

Характеристика систем компьютерной поддержки доказательств

Название (Proof-Assistants)	**Характеристика**	**Авторы** (страна, язык, реализации)	**Ссылка в Internet**
Lego	Система, допускающая “настройку” на использование в качестве логической основы Эдинбургской LF, Calculus of Constructions, UTT, (это – теории типов с зависимыми типами, произведениями, суммами, индуктивными типами, подтипами).	R. Pollack и Zhaohui Luo (Великобритания); реализована в языке ML New Jersey	www.durham.ac.uk/~dcs0zl/
Plastik	Логическая основа: LF, предложенная Zhaohui Luo (теории типов с зависимыми типами, произведениями, семействами индуктивных типов, подтипами)	P. Callaghan (Великобритания); реализована в языке Haskell	www.durham.ac.uk/~dcs0zl/
Coq	Система на базе Calculus of Constractions (исчисления, включающего зависимые типы, индуктивные типы, произведения, суммы, подтипы).	G. Huet, G. Kahn, C.Paulin-Mohring (Франция), реализована в языке Objective Caml	coq.inria.fr/

PCoq	На базе системы Coq с графическим интерфейсом	Yves Bertot (Франция), реализована в языке Ocaml и Java	www-sop.inria.fr/lemme/
PVS	Программа, использующая типы, но основанная на классической логике высших порядков и μ-исчислении	SRI International Cambridge Computing Lab. (Англия)	pvs.csl.sri.com/
HOL	Используются логики высших порядков (Higher Order Logic)	R. Milner, M. Gordon (Великобритания), реализована в языке ML	www.cl.cam.ac.uk/Research/HVG/HOL/
Isabelle	Используются логики высших порядков (в сочетании с HOL)	L. Paulson, (Cambridge) Tobias Nipkow(TU, Мюнхен)	www.cl.cam.ac.uk/Research/HVG/ isabelle
Elf	LF. разработанная Мартин-Лёфом (Стокгольм)	F. Pfenning (Швеция, США), реализована в языке ML	www-2.cs.cmu.edu/~fp/elf.html
Alf	Хорновская логика	M. Hanus, A. Schwab (Германия)	www.informatik.uni-kiel.de/~mh/
Prolog	Хорновская логика (логическое программирование, не теория типов)	Создан в 1987 г., существует в объектном коде. Исходных текстов нет	www.swi-prolog.org/

Основные направления современных исследований по теории типов могут быть охарактеризованы следующим образом:

– разработка и развитие систем поддержки доказательства (PA);
– исследование моделей для теории типов;
– разработка гибридных методов (например, проверка на моделях + доказательство);
– определение “унифицирующего подхода” (например, понятие прототипов);
– создание конкретных прикладных программ, встраиваемых в библиотеку модулей (например, проверка корректности программ, анализ естественного языка интерфейс для неспециалистов и т.д.).

В настоящее время большая часть работ по этим направлениям осуществляется в Западной Европе. Ряд групп (в странах Европейского союза и ассоциированных с ним) объединен в группу "Types".

Таблица 2

Краткая характеристика систем компьютерной алгебры

Название	Характеристика	Авторы (страна, язык реализации)	Ссылка в Internet (http, home page)
Axiom	Программа для расчетов с анализом логических правил. Объектно-ориентированная. Связь с библ. NAG на Фортране.	NAG Ltd., Wilkinson House, Jordan Hill Road (Англия, Оксфорд, реализована в языке Си)	www.nag.co.uk/Symbolic/AX.html home.earthlink.net/~jgg964/axiom.html
Derive	Математический помощник в аналитических вычислениях (развитие системы muMATH)	Soft Warehouse Inc., MathWare Ltd. (США, реализована в языке Си)	www.mathware.com/Software/Derive www.derive.com/drvset.htm www.cms.livjm.ac.uk/deriveprogramming
Maple	Наиболее популярный универсальный математический пакет СА с дополнительной библиотекой подпрограмм	Waterloo Maple Inc., Gonnet G., Gentleman V. (Канада, реализована в языке Си)	www.maplesoft.com www.risc.uni-linz.ac.at/software/distmaple/ www.softline.ru www.exponenta.ru www.math.rsu.ru/mexmat/kvm/MME
MUMath	Математические вычисления для микрокомпьютеров по аналогии с REDUCE и MACSYMA	Stoutemyer D.R., Rich A. (США, реализована в языке muSIMP)	Разработчики системы прекратили работу по ее развитию, переключившись на систему Derive
MACSYMA	Одна из первых математических символьных программ, (первоначально развивалась только на компьютерах VAX, на Лисп-машинах фирм Symbolics, LMI и на рабочих станциях Sun)	C.Engelman, W. Martin, J.Moses, Symbolics Inc., Macsyma Inc. (США, Массачусетский технологический инст-т при участии ARPA, NASA, DOE, реализ в Common Lisp)	www.scientek.com/macsyma/ www.macsyma.com/ distance-ed.math.tamu.edu/Math696/Maxima.html
Mathematica	Самая мощная универсальная математическая система СА	S. Wolfram Research Inc. (США, реализована в языке Си)	www.Wolfram.com library.Wolfram.com/documents
MathCAD	Гибкая интегрированная математическая САПР (Mathmatica + Computer Aided Design), дополнена средствами СА от Maple	MathSoft Inc. (США, реализована в языке-Си++)	www.mathsoft.com www.exponenta.ru
MATLAB	Диалоговая интегрированная среда для матричных вычислений (MATrix LABoratory), дополнена средствами СА из Maple	C.Engelman, MathWorks Inc. (США, реализована в языках Лисп, Фортран, C/C++)	www.mathworks.com www.softline.ru www.exponenta.ru
REDUCE	Интерактивная система аналитических вычислений (ранее была самой распространенной)	A.C. Hearn, Univ. Of Utah, Rand Corp. (США, реализована в языке Лисп)	www.uni-koeln.de/REDUCE www.rrz.uni-koeln.de/REDUCE/ www.codemist.tc/reduce/index.html www.zib.de/Symbolik/reduce/

Дадим несколько более подробную характеристику по центрам, темы работ которых можно соотнести с перечисленными рубриками.

В настоящее время координатором проекта по теории типов является Z. Luo, руководитель группы математиков и логиков в университете Дарема (Durham), Великобритания. Тематика группы включает теорию логических структур, таких как UTT (Универсальная Теория Типов) [22], PAL+ (теория типов без λ-абстракции) [23], работы над системами поддержки доказательств Lego и Plastic [24, 25], работы, связанные с понятием подтипа [26], функциональное программирование с использованием зависимых типов [27], типы в естественном языке [28]. Подробную информацию об этих исследованиях можно найти также на страницах Internet.

К наиболее крупным центрам проекта относится также центр в Техническом Университете Чалмерс (Chalmers) (Götheborg), который был координатором проекта на предыдущем этапе. Здесь ведутся исследования по следующим направлениям:

- алгебраические типы данных в теории типов Мартин-Лёфа [4];
- система Alfa для редактирования доказательств и программ, основанная на теории типов; эта система используется в процессе обучения студентов с интерфейсом на естественном языке;
- работы по представлению рекурсии в теории типов;
- модели теории зависимых типов [14];
- работа по конструктивизации математики с использованием теории типов. В этих работах подчеркивается, что теория типов относится к особой ветви конструктивной математики [5,20];
- сочетание построения доказательства с проверкой на моделях [Dubjer, Takeyama];
- применение теории типов к работе с полуформализованными текстами [10–13].

Крупнейшим центром является ИНРИА (Франция) – (INRIA Rocquencourt и INRIA– Sofia–Antipolis). Другими важными центрами во Франции являются Университет Париж-7, Университет Орсей и связанные с ними исследовательские группы.

Основная система поддержки доказательства, с которой ведется работа во Франции – это система Coq с языком, который называется также Coq. В его основе лежит так называемое исчисление конструкции ("Calculus of Constructions") [29]. Страница Internet, посвященная Coq https://coq.inria.fr/ содержит загрузочные версии системы Coq 8.5. Coq занимает одно из рекордных мест среди систем PA по числу конкретных разработок по компьютерной формализации математических теорий. Стандартная библиотека включает следующие разделы: классическая логика и зависимое равенство, основы арифметики Пеано, бинарное представление целых чисел, формализацию вещественных чисел, булевы функции, классическую и конструктивную теорию множеств. Разработки, добавленные пользователями, включают Конструктивную Теорию Категорий [30], элементы

конструктивной геометрии, теории групп и областей [G. Kahn], парадоксы теории множеств и теории типов (B. Barras, T. Coquand, B. Werner).

Центр в университете Nijmegen (Нидерланды) входит одновременно в группу “Types” и в проект “Calculemus”. Здесь работает руководитель проекта Calculemus” профессор H. Barendregt [6, 8], который вместе с соавторами подготовил фундаментальную монографию по *типовому λ-исчислению*. Велась работа по разработке понятия “setoid”, множества с отношением эквивалентности на элементах, которое, в отличие от классического понятия множества, допускает формализацию в конструктивной математике и играет большую роль в PA. Велись работы с языком Coq по формализации конструктивных вещественных чисел для доказательства простоты больших чисел [35]. Системы компьютерной алгебры использовались для вычисления «тестовых значений», используемых в формальном доказательстве [36]. Из работ по формализации других областей математики следует упомянуть проверку корректности быстрого преобразования Фурье [37].

Два крупных центра находятся в Мюнхене (Университет Людвига Максимилиана и Технический Университет).

Ограниченный объем данной статьи не позволяет подробно охарактеризовать работу, осуществляемую в других центрах. Центры, перечисленные выше, являются весьма представительными в отношении исследований, осуществляемыми в группе “Types”. Приведем общий список центров и соответствующих Internet-адресов.

Великобритания

http://www.dur.ac.uk/TYPES/ (Durham University) (проект закончен)
http://www.cl.cam.ac.uk/research/pls/ (Cambridge University)
http://www.lfcs.informatics.ed.ac.uk/ (University of Edinburgh)
http://www.maths.man.ac.uk/DeptWeb/Groups/Logic/LogicResearch.html
http://www.maths.manchester.ac.uk/study/postgraduate/projects/mathematical-logic/
(University of Manchester).
http://www.dcs.qmul.ac.uk/ (Queen Mary and Westfield College)
http://www.dcs.shef.ac.uk/ (University of Sheffield).

Франция

http://pauillac.inria.fr/ (INRIA-Rocquencourt)
http://www.logique.jussieu.fr/index-english.html (Университет Paris-7)
http://www-sop.inria.fr/certilab/certilab-eng.html (INRIA Sofia-Antipolis)
http://www.imj-prg.fr/lm/
http://www.lri.fr/ (Paris–Sud).

Германия

http://www.mathematik.uni-muenchen.de/~logik/ (LM Munchen)
http://www4.informatik.tu-muenchen.de/proj/types/ (TU Munchen)

Италия

http://www.di.unito.it/ (Университет Турина)
http://www.dimi.uniud.it/SLP/gruppo.html (Университет Удине).

Нидерланды

http://www.cs.kun.nl/ (Университет Nijmegen).

Польша
http://www.uwb.edu.pl/index-en.html (Белостокский университет)
http://zls.mimuw.edu.pl/~urzy/home.html (Варшавский университет).
Португалия
http://www.math.uminho.pt/site/frontoffice/dmum/index.html
(Университет Minho)
Швеция
http://www.cs.chalmers.se/ComputingScience/Research/Logic/
(Университет Chalmers)

Проект “Calculemus” был основан в 1999 году. Целью проекта является разработка нового поколения программных систем, основанных на соединении возможностей по формализации доказательств, предоставляемых РА, и вычислительной силы систем СА. Один из основных центров, участвующих в проекте – это RISC (Research Institute for Symbolic Computation) в Линце (Австрия).

Для полноты картины следует упомянуть и о ряде исследовательских центров вне Европы, где ведутся работы в рассмотренных выше направлениях. Это – университет Карнеги Мелон и Йельский университет в США,– проект Twelf – логические и металогические структуры [38]; университет Квинсленд (Quieensland) в Австралии (центр верификации программ); JAIST (Институт Инновационных Исследований в Науке и Технологии, Япония).

4. Использование расширенных КС-грамматик в регулярной форме для представления вычислительных и логических правил

В связи с огромным разнообразием логических систем и вычислительных правил, представляющих интерес для информатики, большую ценность представляют языки описания логических систем. Одним из перспективных, но недостаточно исследованных подходов здесь является использование формальных грамматик. Приведем пример грамматического представления схемы логического правила (в секвенциальном исчислении).

Рассмотрим правило:

$$\frac{\Gamma \mapsto A; B\Gamma' \mapsto C}{\Gamma(A \Rightarrow B)\Gamma' \mapsto C}$$ (введена импликация слева)

С грамматической точки зрения терминалами здесь являются символы $;$, $\mapsto$, $\Rightarrow$, $(,)$, Γ , Γ' – нетерминалы для списка формул, *A, B, C* – нетерминалы для формул. Сами по себе схемы правил являются регулярными выражениями над терминалами и нетерминалами, причем, на этапе вывода до терминальных выражений используется “почти КС”-грамматика (используются параллельные продукции). Конкретное применение правила получается применением грамматических продукций к нетерминалам, в результате чего нетерминалы Γ , Γ' превращаются в конкретные списки формул (параметрические формулы), *A, B, C* – в конкретные формулы.

Построение схемы вывода требует решения грамматических уравнений (обычно несложных) для проверки возможности отождествления заключения одних правил с посылками других. Это позволяет использовать уже существующие системы грамматического анализа, такие как система SynGT, разрабатываемая в СПИИРАН. Применения этого подхода лежат в области построения прототипов доказательств (схем), преобразовании доказательств, осуществляемых на уровне схем и многое другое [42-45].

5. Изоморфизм типов и его применения

Исследования по изоморфизму типов, начатые более 20 лет назад [39], см. также монографию [40] сильно активизировались в последнее время благодаря новым применениям в теории типов и РА, в задаче поиска данных, в библиотеках функций (функциональных языков) и в Интернете.

В рамках данного проекта ведется работа по введению новых правил редукции термов в теории типов, основанных на использовании изоморфизмов []. Одна из проблем, затрудняющих создание практически эффективного интерфейса между РА и системами СА, состоит в том, что основным отношением равенства в РА является равенство, основанное на редукции термов. Например, в случае функции (элементов функционального типа *A --> B*), такие функции считаются равными, если они задаются термами, имеющими одну и ту же нормальную форму (так называемое “интенсиональное равенство”). Это отношение значительно уже обычного экстенсионального равенства, где требуется лишь равенство значений для всех значений аргументов. В отличие от систем СА, в которых введение правил преобразования выражений пользователем (если это возможно) осуществляется без дополнительного контроля корректности, подход, основанный на введении редукции, связанных с изоморфизмами, включает в себя проверку корректности (доказательство существования и единственности нормальных форм) [41].

Заключение

В качестве перспективной формальной базы для совместного использования систем компьютерной алгебры и систем компьютерной поддержки доказательства теорем начата разработка подхода, основанного на использовании формальных грамматик для представления схем логических выводов и вычислительных схем.

Наиболее современные системы компьютерной поддержки доказательства основаны на использовании расширяемых систем правил, содержащих некоторое постоянное логическое ядро (Logical Framework) и расширяемую часть, состоящую из правил, задаваемых пользователем.

Вычислительные алгебраические правила, используемые системой компьютерной алгебры, при надлежащей их формализации будут относиться к этой расширяемой части. Формализация эта является необходимым этапом, предшествующим совместному использованию системы

компьютерной поддержки доказательства и систем компьютерной алгебры.

Вопрос о форме задаваемых пользователем правил не является достаточно проясненным. Это может быть заранее заданный создателями системы запас схем, где от пользователя зависит только выбор параметров. В других случаях формализация полностью оставляется на усмотрение пользователя. Поход, основанный на использовании грамматик, находится между этими двумя крайностями.

К основным преимуществам этого подхода относятся гибкость в представлении схем (схема правила рассматривается как нетерминальное выражение в грамматике) и наличие большого запаса программных средств поддержки. Одним из таких средств является система SynGT (Syntax Graph Transformations), разрабатываемая участниками проекта с российской стороны.

Литература

1. Шанин Н.А. Доклад на IV Всесоюзном математическом съезде в июне 1961 г.

2. Whitehead, Alfred North, and Bertrand Russell (1910, 1912, 1913) Principia Mathematica, 3 vols, Cambridge: Cambridge University Press. Second edition, 1925 (Vol. 1), 1927 (Vols 2, 3). Abridged as Principia Mathematica to *56, Cambridge: Cambridge University Press, 1962.

3. Godel K. On intuitionistic arithmetic and number theory. In M. Davis (Ed.) The Undecidable. Raven Press. N.Y., 1965.

4. Martin-Lof P. Intuitionistic type theory. Bibliopolis, 1984.

5. Coquand T., Huet G. Constructions: a higher order proof system for machanized mathematics. Proc. 1985 European Conference on Computer Algebra, Lecture Notes in Computer Science. V. 203. Springer, 1985.

6. Barendregt H., Barendsen E. “Autarkic computations in formal proofs“. Journal of Automated Reasoning, 28(3). 2002. P. 321 – 336.

7. Bronstein M. “Computer Algebra Algorithms for Linear Ordinary Differential and Difference Equations”. Preprint INRIA. 2001.

8. Barendregt H., (with Cohen A.) “Electronic Communication of Mathematics and the Interaction of Computer Algebras Systems and Proof Assistants” J. Symbolic Computation, 32 (2001). P. 3 – 22.

9. Barendregt H. (with Geuvers H.) Proof-checking using Dependent Type Systems. Handbook of Artificial Reasoning. Vol.II. Ch.18 (2001). 1149–1240.

10. Ranta A. Grammatical Framework: A Type-Theoretical Grammar Formalism. Manuscript, September 2002. Journal of Functional Programming, to appear.

11. Ranta A. Type-theoretical interpretation and generalization of phrase structure grammar. Bulletin of the IGPL 3. 1995. P. 319 – 342.

12. Hallgren T. and Ranta A. "An Extensible Proof Text Editor". M. Parigot & A. Voronkov (eds), Logic for Programming and Automated Reason-

ing (LPAR'2000), LNCS/LNAI. 1955. P. 70 – 84, Springer Verlag, Heidelberg, 2000.

13. Ranta A. A Multilingual Natural-Language Interface to Regular Expressions, L. Karttunen and K. Oflazer (eds). Proceedings of the International Workshop on Finite State Methods in Natural Language Processing, Bilkent University, Ankara, 1998. P. 79 – 90.

14. Coquand T. Domain Models in Type Theory. URL: (Slides: http://www.cs.chalmers.se/~coquand/type.html).

15. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на C++. — Изд-во Бином, 1998. — 560 с.

16. De Bruijn, N.G. A survey of the project AUTOMATH. In: Seldin, J.P. and Hindley, J.R. Eds. To H.B. Curry:Essays in Combinatory Logic, Lambda Calculus and Formalism. (1980). Academic Press. P. 589–606.

17. Gordon M., Milner R. and Wadsworth C. Edinburgh LCF: A Mechanized Logic of Computation. Lect. Notes in Comp. Sci, V. 78. 1979.

18. Paulson L. Interactive Theorem Proving with Cambridge LCF. Tech. rep. 80, Computer Laboratory, Univ. of Cambridge, Nov. 1985.

19. Petersson K. A programming system for type theory. Tech. Rep.21, Programming Methodology Group, Univ. of Gotheborg / Chalmers Institute of Technology. Mar. 1982.

20. Coquand T., Huet G. Constructions: A higher-order proof system for machanized mathematics. In: EUROCAl'85: European Conference on Computer Algebra (1985), B. Buchberger, Ed., Vol. 203 of Lecture Notes in Computer Science, Spinger. P.151 – 184.

21. Constable R. L. et al. Implementing Mathematics with the NuPRL Proof Development System. Prentice Hall, 1986.

22. Luo Z. Computation and Reasoning: A Type Theory for Computer Science. Oxford University Press, Oxford, 1994.

23. Luo Z. PAL+: A lambda-free logical framework. J. Functional Programming, 2001.

24. Luo Z., Pollack R. Lego Proof Developmnet System: user's manual, LFCS Report ECS–LFCS–92–211, department of computer science, University of Edinburgh.

25. Callaghan P. and Luo Z. Plastic: an implementation of typed LF with coercive subtyping and universes. Submitted to Proc. LFM'99, 2000.

26. Soloviev S., Luo Z. Coercion Completion and Conservativity in Coercive Subtyping. Annals of Pure and Applied Logic. 113 (2002). 297 – 322.

27. McBride C. Faking It: Simulating Dependent Types in Haskell. J.

28. Callaghan P. C. An Evaluation of LOLITA and Related Natural Language Processing Systems. PhD thesis, University of Durham, Department of Computer Science, Laboratory for Natural Language Engineering, 1997.

29. Coquand T. and Huet G. The calculus of constructions. Information and Computation , 76(2/3):95–120. February/March 1988.

30. Saibi A. and Huet G. "Constructive Category Theory" In Proceedings of the joint CLICS–TYPES Workshop on Categories and Type Theory, Goteborg

(Sweden), January 95.

31. Матулис В. А. Первый всесоюзный симпозиум по проблеме машинного поиска логического вывода. — Успехи мат. наук. — Т. XIX, вып. 6. — 1964. — С. 239 – 241.

32. Шанин Н. А., Давыдов Г. В., Маслов С. Ю., Минц Г.Е., Оревков В.П., Слисенко А. О. Алгорифм машинного поиска естественного логического вывода в исчислении высказываний. — «Наука», 1965. — 39 с. [in: Automation of reasoning, Springer-Verlag, Berlin. Vol.1. P. 424 – 483, 1983].

33. Маслов С. Ю. Обратный метод установления выводимости в классическом исчислении предикатов. — Докл. АН СССР. — Т.159, №.1. — 1964. — С. 17 – 20.

34. Давыдов Г. В., Маслов С. Ю., Минц Г.Е., Оревков В.П., Слисенко А. О. Машинный алгорифм установления выводимости на основе о обратного метода. — Зап. научн. семинаров ЛОМИ. — 1969. — Т .16. — С. 8 – 19.

35. Niqui M. Constructive Reals in Coq: Axioms and First Order Equivalence, Types 2000 Workshop, December 7–12 2000. Univ. of Durham.

36. Capprotti O., Oostdijk M. Formal and effective primality proofs by use of Computer Algebra Oracles. JSC, Special Issue on Computer Algebra and Mechanized Reasoning, 2001.

37. Carpetta V. Certifying Fast Fourier Transform with Coq. Manuscript.

38. Frank Pfenning and Carsten Schermann. System description: Twelf - a meta-logical framework for deductive systems. In H. Ganzinger, editor, Proceedings of the 16th International Conference on Automated Deduction (CADE-16), Trento, Italy, July 1999. Springer-Verlag LNAI 1632. P. 202 – 206.

39. Соловьев С.В. Категория конечных множеств и декартово замкнутые категории. — Зап. Научных семинаров ЛОМИ. — 1981. — Т. 105. — С. 174 – 194.

40. Di Cosmo R. Isomorphism of Types. Burkhauser, 1995.

41. Chemouil D., Soloviev S. Extensional Isomorphisms of Inductive Types in Simply Typed Lambda-calculus. — 15 p. Submitted.

42. Федорченко Л. Н., Соловьев С. В. Синтаксические преобразования в системе SynGT и их новые приложения // Материалы VIII Санкт-Петербургской международной конференции «Региональная информатика–2002». — Ч. 2. — СПб: СПОИСУ, 2002. — С. 50.

43. Fedorchenko L. N., Soloviev S. V., Naumov I. N., Mehats L. Syntax Graph Transformations in the System SynGT and their Applications. — Rapport IRIT/2003-06-R. UMR 5505 CNRS-INP-UPS. 15 pp. (English).

44. Fedorchenko L. and Baranov S. Equivalent Transformations and Regularization in Context–Free Grammars. Bulgarian Academy of Sciences. Cybernetics and Information Technologies (CIT). Vol. 14. No. 4. P. 11–28. Sofia 2015 (Scopus).

45. Федорченко Л. Н. SynGT: применение атрибутов // Региональная информатика и информационная безопасность. — Сб. трудов. —

Вып. 1. — СПб.: СПОИСУ, 2015. — С. 68 – 73.

46. Boisvert B., Feraud L., Soloviev S. Typed lambda terms in categorical graph rewriting // In The International Conference Polynomial Computer Algebra, April 18-22, 2011. Saint-Petersburg, Russia, Euler International Mathematical Institute.VVM Publishing. P. 9 – 16.

47. Баранов С. Н., Буавер Б., Соловьев С. В., Феро Л. Некоторые приложения лямбда-исчислений с типами к атрибутным вычислениям в системах категорных преобразований графов // Труды СПИИРАН. — 2012. — Вып. 23. — С. 296 – 323.

References

1. Shanin N.A. Doklad na IV Vsesojuznom matematicheskom s#ezde v ijune 1961 g.

2. Whitehead, Alfred North, and Bertrand Russell (1910, 1912, 1913) Prin-cipia Mathematica, 3 vols, Cambridge: Cambridge University Press. Second edition, 1925 (Vol. 1), 1927 (Vols 2, 3). Abridged as Principia Mathematica to *56, Cambridge: Cambridge University Press, 1962.

3. Godel K. On intuitionistic arithmetic and number theory. In M. Davis (Ed.) The Undecidable. Raven Press. N.Y., 1965.

4. Martin-Lof P. Intuitionistic type theory. Bibliopolis, 1984.

5. Coquand T., Huet G. Constructions: a higher order proof system for machanized mathematics. Proc. 1985 European Conference on Computer Algebra, Lecture Notes in Computer Science. V. 203. Springer, 1985.

6. Barendregt H., Barendsen E. “Autarkic computations in formal proofs“. Journal of Automated Reasoning, 28(3). 2002. P. 321 – 336.

7. Bronstein M. “Computer Algebra Algorithms for Linear Ordinary Differential and Difference Equations”. Preprint INRIA. 2001.

8. Barendregt H., (with Cohen A.) “Electronic Communication of Mathematics and the Interaction of Computer Algebras Systems and Proof Assistants” J. Symbolic Computation, 32 (2001). P. 3 – 22.

9. Barendregt H. (with Geuvers H.) Proof-checking using Dependent Type Systems. Handbook of Artificial Reasoning. Vol.II. Ch.18 (2001). 1149–1240.

10. Ranta A. Grammatical Framework: A Type-Theoretical Grammar Formalism. Manuscript, September 2002. Journal of Functional Programming, to appear.

11. Ranta A. Type-theoretical interpretation and generalization of phrase structure grammar. Bulletin of the IGPL 3. 1995. P. 319 – 342.

12. Hallgren T. and Ranta A. "An Extensible Proof Text Editor". M. Parigot & A. Voronkov (eds), Logic for Programming and Automated Reasoning (LPAR'2000), LNCS/LNAI. 1955. P. 70 – 84, Springer Verlag, Heidelberg, 2000.

13. Ranta A. A Multilingual Natural-Language Interface to Regular Expres-sions, L. Karttunen and K. Oflazer (eds). Proceedings of the International

Workshop on Finite State Methods in Natural Language Processing, Bilkent University, Ankara, 1998. P. 79 – 90.

14. Coquand T. Domain Models in Type Theory. URL: (Slides: http://www.cs.chalmers.se/~coquand/type.html).

15. Buch G. Ob#ektno-orientirovannyj analiz i proektirovanie s pri-merami prilozhenij na S++. — Izd-vo Binom, 1998. — 560 s.

16. De Bruijn, N.G. A survey of the project AUTOMATH. In: Seldin, J.P. and Hindley, J.R. Eds. To H.B. Curry:Essays in Combinatory Logic, Lambda Calculus and Formalism. (1980). Academic Press. P. 589–606.

17. Gordon M., Milner R. and Wadsworth C. Edinburgh LCF: A Mechanized Logic of Computation. Lect. Notes in Comp. Sci, V. 78. 1979.

18. Paulson L. Interactive Theorem Proving with Cambridge LCF. Tech. rep. 80, Computer Laboratory, Univ. of Cambridge, Nov. 1985.

19. Petersson K. A programming system for type theory. Tech. Rep.21, Programming Methodology Group, Univ. of Gotheborg / Chalmers Institute of Technology. Mar. 1982.

20. Coquand T., Huet G. Constructions: A higher-order proof system for machanized mathematics. In: EUROCAl'85: European Conference on Computer Algebra (1985), B. Buchberger, Ed., Vol. 203 of Lecture Notes in Computer Science, Spinger. P.151 – 184.

21. Constable R. L. et al. Implementing Mathematics with the NuPRL Proof Development System. Prentice Hall, 1986.

22. Luo Z. Computation and Reasoning: A Type Theory for Computer Science. Oxford University Press, Oxford, 1994.

23. Luo Z. PAL+: A lambda-free logical framework. J. Functional Programming, 2001.

24. Luo Z., Pollack R. Lego Proof Developmnet System: user's manual, LFCS Report ECS–LFCS–92–211, department of computer science, University of Edinburgh.

25. Callaghan P. and Luo Z. Plastic: an implementation of typed LF with coercive subtyping and universes. Submitted to Proc. LFM'99, 2000.

26. Soloviev S., Luo Z. Coercion Completion and Conservativity in Coercive Subtyping. Annals of Pure and Applied Logic. 113 (2002). 297 – 322.

27. McBride C. Faking It: Simulating Dependent Types in Haskell. J.

28. Callaghan P. C. An Evaluation of LOLITA and Related Natural Language Processing Systems. PhD thesis, University of Durham, Department of Computer Science, Laboratory for Natural Language Engineering, 1997.

29. Coquand T. and Huet G. The calculus of constructions. Information and Computation , 76(2/3):95–120. February/March 1988.

30. Saibi A. and Huet G. "Constructive Category Theory" In Proceedings of the joint CLICS–TYPES Workshop on Categories and Type Theory, Goteborg (Sweden), January 95.

31. Matulis V. A. Pervyj vsesojuznyj simpozium po probleme mashinnogo poiska logicheskogo vyvoda. — Uspehi mat. nauk. — T. XIX, vyp. 6. — 1964. — S. 239 – 241.

32. Shanin N. A., Davydov G. V., Maslov S. Ju., Minc G.E., Orevkov V.P., Slisenko A. O. Algorifm mashinnogo poiska estestvennogo logicheskogo vyvoda v ischislenii vyskazyvanij. — «Nauka», 1965. — 39 s. [in: Automation of reasoning, Springer-Verlag, Berlin. Vol.1. P. 424 – 483, 1983].

33. Maslov S. Ju. Obratnyj metod ustanovlenija vyvodimosti v klassicheskom ischislenii predikatov. — Dokl. AN SSSR. — T.159, №.1. — 1964. — S. 17 – 20.

34. Davydov G. V., Maslov S. Ju., Minc G.E., Orevkov V.P., Slisenko A. O. Mashinnyj algorifm ustanovlenija vyvodimosti na osnove o ob-ratnogo metoda. — Zap. nauchn. seminarov LOMI. — 1969. — T .16. — S. 8 – 19.

35. Niqui M. Constructive Reals in Coq: Axioms and First Order Equivalence, Types 2000 Workshop, December 7–12 2000. Univ. of Durham.

36. Capprotti O., Oostdijk M. Formal and effective primality proofs by use of Computer Algebra Oracles. JSC, Special Issue on Computer Algebra and Mechanized Reasoning, 2001.

37. Carpetta V. Certifying Fast Fourier Transform with Coq. Manuscript.

38. Frank Pfenning and Carsten Schermann. System description: Twelf - a meta-logical framework for deductive systems. In H. Ganzinger, editor, Proceedings of the 16th International Conference on Automated Deduction (CADE-16), Trento, Italy, July 1999. Springer-Verlag LNAI 1632. P. 202 – 206.

39. Solov'ev S.V. Kategorija konechnyh mnozhestv i dekartovo zamknutye kategorii. — Zap. Nauchnyh seminarov LOMI. — 1981. — T. 105. — S. 174 – 194.

40. Di Cosmo R. Isomorphism of Types. Burkhauser, 1995.

41. Chemouil D., Soloviev S. Extensional Isomorphisms of Inductive Types in Simply Typed Lambda-calculus. — 15 p. Submitted.

42. Fedorchenko L. N., Solov'ev S. V. Cintaksicheskie preobrazovanija v sisteme SynGT i ih novye prilozhenija // Materialy VIII Sankt-Peterburgskoj mezhdunarodnoj konferencii «Regional'naja informati-ka–2002». — Ch. 2. — SPb: SPOISU, 2002. — S. 50.

43. Fedorchenko L. N., Soloviev S. V., Naumov I. N., Mehats L. Syntax Graph Transformations in the System SynGT and their Applications. — Rapport IRIT/2003-06-R. UMR 5505 CNRS-INP-UPS. 15 pp. (English).

44. Fedorchenko L. and Baranov S. Equivalent Transformations and Regularization in Context–Free Grammars. Bulgarian Academy of Sciences. Cybernetics and Information Technologies (CIT). Vol. 14. No. 4. P. 11–28. Sofia 2015 (Scopus).

45. Fedorchenko L. N. SynGT: primenenie atributov // Regional'naja informatika i informacionnaja bezopasnost'. — Sb. trudov. — Vyp. 1. — SPb.: SPOISU, 2015. — S. 68 – 73.

46. Boisvert B., Feraud L., Soloviev S. Typed lambda terms in categorical graph rewriting // In The International Conference Polynomial Computer Algebra, April 18-22, 2011. Saint-Petersburg, Russia, Euler International Mathematical Institute.VVM Publishing. P. 9 – 16.

47. Baranov S. N., Buaver B., Solov'ev S. V., Fero L. Nekotorye prilozhenija ljambda-ischislenij s tipami k atributnym vychislenijam v sistemah kategornyh preobrazovanij grafov // Trudy SPIIRAN. — 2012. — Vyp. 23. — S. 296 – 323.

Федорченко Людмила Николаевна, кандидат технических наук, старший научный сотрудник лаборатории прикладной информатики Санкт-Петербургского института информатики и автоматизации Российской академии наук (СПИИРАН), e-mail: LNF@iias.spb.su.

Fedorchenko Ludmila Nikolaevna, PhD in Engineering, Senior Researcher, St. Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences (SPIIRAS).