АНАЛИТИЧЕСКАЯ ФИЛОСОФИЯ, ЭПИСТЕМОЛОГИЯ И ФИЛОСОФИЯ НАУКИ

УДК 004.055, 004.82, 168.2, 510.62, 510.67

DOI 10.25205/2541-7517-2018-16-4-5-25

С. С. Гончаров, Д. И. Свириденко

*Институт математики им. С. Л. Соболева СО РАН
пр. Акад. Коптюга, 4, Новосибирск, 630090, Россия*

*Новосибирский государственный университет
ул. Пирогова, 1, Новосибирск, 630090, Россия*

gonchar@math.nsc.ru; dsviridenko47@gmail.com

СЕМАНТИЧЕСКОЕ МОДЕЛИРОВАНИЕ И ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ *

Обсуждается проблема решения человеком задач с помощью компьютеров. Анализируются достоинства и недостатки декларативного программирования, в том числе функционального и логического программирования, выступающих как инструментальный искусственного интеллекта. Описывается альтернативный, теоретико-модельный подход к решению задач, носящий название семантического моделирования. Анализируются достоинства предлагаемой концепции, в том числе возможность сочетания аксиоматического и теоретико-модельного подхода к решению задач в рамках единого подхода, а также возможность интеграции на базе семантического моделирования методов непрерывной и дискретной математики. Описывается возможность построения на основе семантического моделирования нового, «объяснительного» искусственного интеллекта, свободного от недостатков, присущих традиционному искусственному интеллекту.

Ключевые слова: искусственный интеллект, машинное обучение, семантическое моделирование, декларативное программирование, вычислимость, теоретико-модельный подход.

Человек в живой природе занимает особое место и одно из основных качеств, выделяющих его в ней, – это умение понимать и ставить задачи, решать их и, что очень важно, при необходимости не только

* Исследование выполнено при финансовой поддержке Российского научного фонда (проект № 17-11-01176).

объяснить найденное решение, но и *обосновать* его. Длительное время человек достаточно успешно пользовался этим умением, не прибегая к помощи каких-либо внешних инструментов, пока не столкнулся с необходимостью научиться решать сложные и масштабные задачи. Это вынудило его искать соответствующий инструментарий, и он придумал компьютер – пожалуй, один из чрезвычайно перспективных и весьма эффективных инструментов.

При решении человеком задач с помощью компьютера стандартная последовательность действий выглядит следующим образом: формулируется задача, ищется алгоритм ее решения, формальное или формализованное описание алгоритма передается программисту для написания кода на том или ином языке программирования, понятного используемому компьютеру, пишется и отлаживается программа, планируются и осуществляются вычисления, порождаемые написанной программой, полученные данные интерпретируются Заказчиком, и если они удовлетворительны с точки зрения некоторого критерия, то принимается решение, что задача решена.

Если же результат неудовлетворителен, то производится анализ предпринятых действий и в них вносятся соответствующие изменения. Заметим, что такие изменения могут коснуться практически всех этапов процесса решения задачи, вплоть до этапа ее постановки. При этом ошибки, нечеткости или неточности, допущенные при формулировке задачи, могут повлечь за собой весьма «дорогостоящие» последствия, что налагает на постановщика задачи особую ответственность.

В связи с таким «экономическим» подходом к проблеме компьютерного решения задач имеет смысл вспомнить известный афоризм: «правильно поставленная задача – наполовину решенная задача». Сразу же возникает вопрос – а существует ли ситуация, когда правильно поставленная задача фактически означает ее полноценное решение? Поскольку речь идет о решении человеком задач с помощью компьютеров, то этот вопрос можно переформулировать иначе – существует ли возможность формулировать задачу таким образом, чтобы из этой формулировки компьютер мог бы автоматически «извлечь» требуемое решение? И не только извлечь решение, но и *объяснить* его?

Заметим, что этот вопрос не оригинален – решением похожей проблемы занимается *искусственный интеллект* (далее – ИИ) [Тейз и др., 1990; Рассел, Норвиг, 2007; Flasiński, 2016]. В частности, такое направление ИИ, как машинное обучение (см., например; [Kubat, 2017]) решает эту проблему, пытаясь с помощью математико-статистических

методов и технологии нейронных сетей путем их обучения (методом «натаскивания») синтезировать решение той или иной массовой задачи. Скажем, по радужной оболочке человеческого глаза или по тепловому полю поставить пациенту диагноз, распознать человеческое лицо в толпе идущих людей и т. д.

Следует отметить, что технология нейронных сетей имеет много удачных и вполне успешных приложений и активно развивается, особенно в последнее время. Но она же обладает и серьезным изъяном – синтезируемые нейронными сетями решения являются плохо или даже вообще необоснованными, поскольку никто не знает, как нейронная сеть приходит к своим решениям (так называемый эффект «черного ящика»). Кроме этого, системы искусственного интеллекта, базирующиеся на применении технологии традиционных нейронных сетей, являются нестабильными, демонстрируя непредсказуемое поведение в непредусмотренных и нестандартных условиях. Оба этих обстоятельства – необоснованность и нестабильность, делают использование традиционных нейронных сетей проблематичным и даже зачастую вообще неприемлемым в таких областях, как медицина, оборона, логистика, управление сложными человеко-машинными системами и т. п. В чем же причина таких негативных эффектов?

На взгляд авторов этой статьи, главной причиной наличия указанных выше недостатков применяемых технологий ИИ является изолированность и обособленность используемых ими приемов и инструментов от исходного *смысла (семантики)* задачи, т. е. от тех знаний о проблемной области, которые используются при ее постановке и формулировке. Такая изолированность и обособленность усугубляется еще и тем, что при написании компьютерных программ используются, как правило, технологии и языки программирования, носящие *императивный* характер, что автоматически влечет за собой разделение данных и операций над ними [Пратт, Зелковиц, 2003]. Напомним, что при описании задач и при их решении в виде алгоритмов мы всегда имеем дело с двумя видами сущностей:

- *данными*, т. е. с тем, что мы хотим обрабатывать, преобразовывать, управлять;
- *операциями*, т. е. с действиями над данными.

Очевидно, что любой язык формулирования задачи и любой язык, в терминах которого мы записываем ее решение, должен предоставлять возможность описывать эти сущности. Для этого такие языки обычно содержат, с одной стороны, элементарные выражения, представляющие исходные примитивные данные и операции над ними

(базис языка), а с другой – набор специальных средств конструирования из этих простых объектов более сложных конструкций, которые могли бы рассматриваться как единые сущности, что позволяло бы манипулировать этими конструкциями с помощью средств этого же языка. И то, что при императивном программировании мы вынуждены *разделять* данные и операции, является одной из главных причин разрушения исходной семантики задачи. Помимо всего прочего, разрушение и тем самым неполный учет семантических аспектов задачи приводят к таким проблемам, как неструктурированность больших данных (Big Data), с которой, что интересно, предлагается бороться, используя ... те же традиционные технологии ИИ, пытаясь придать с их помощью некий смысл получаемым решениям и как бы восстановить семантику решаемых задач. Возникает порочный круг – вначале разрушаем исходную семантику задачи с тем, чтобы потом попытаться ее восстановить!

Помимо императивных языков программирования, ИИ активно пользуется так называемыми *декларативными языками* [Зюзысов, 2003], каковыми являются функциональные [Хендерсон, 1983; Хьюенен, Сепянен, 1990; Филд, Харрисон, 1993] и логические [Логическое программирование, 1988; Братко, 1990; Малпасс, 1990; Стерлинг, Шапиро, 1990] языки программирования. Но и это не спасает ситуацию. Остановимся чуть подробнее на этом моменте.

На этапе формулирования человеком задачи, решение которой предусматривает использование компьютера и потому формулировка задачи должна быть свободной от неточностей и двусмысленностей, стараются применять по возможности строгие, формальные конструкции. Человеческий опыт подсказывает, что для человека, формулирующего задачу, предпочтительнее, чтобы используемые языковые конструкции имели вид математических утверждений, носящих повествовательный, декларативный характер. Правда иногда, когда задача изначально формулируется как необходимость осуществить совокупность некоторых действий, используется форма представления задачи и в виде последовательности инструкций, носящих уже повелительный, т. е. императивный характер. Очевидно, что императивными по своей природе являются и описания решений задач в виде алгоритмов. И уж совсем отчетливый императивный характер имеют представляющие эти алгоритмы команды компьютерной программы.

Справедливости ради стоит отметить, что императивные языки программирования высокого уровня (например, объектно-ориентированные языки), как правило, имеют достаточно развитые средства

написания повествовательных, декларативных конструкций. Используя такие средства, можно в отдельных случаях, когда уже заранее имеется некая компьютерная программа решения целого класса задач (шаблон), формулировать конкретную задачу из данного класса задач не как последовательность инструкций, а как совокупность декларативных условий, позволяющих параметрически настроить этот шаблон на решение именно этой конкретной задачи и автоматически получать это решение.

Понятно, что такой автоматизированный способ решения задачи выглядит намного привлекательнее, поскольку вместо указаний о том, *как* решать задачу мы формулируем лишь описание того, *что* следует решать. Поэтому вполне закономерным является вопрос: а можно ли создать декларативный язык постановки задач, который бы:

- опирался на общепризнанную и развитую математическую базу;
- максимально был приближен к постановщику задачи, был удобен и комфортабелен для формулировок задач и способен был адекватно отражать в постановке исходную семантику используемых знаний;
- был пригоден для формулировки по возможности максимально широкого класса практических задач;
- допускал бы эффективную интерпретацию своих конструкций как набор команд для компьютера?

Заметим, что в предельном случае этот вопрос звучит как вопрос о возможности создания некоего универсального декларативного языка с добротной математической базой и пригодного для декларативного описания любой(!) задачи, имеющей алгоритмическое решение – такие языки носят название Тьюрингово полных.

Поскольку в формулировке всех этих вопросов речь идет о задачах, допускающих решение с помощью компьютера в виде набора неких вычислительных конструкций, то поиск ответов на поставленные вопросы естественно начать с поиска математической основы такого языка, т. е. с поиска и анализа существующих в дискретной математике моделей, формализующих понятие «вычислимость».

Напомним, что наиболее распространенными в математике являются *язык множеств* и *язык функций*. Поэтому вполне естественно, что изучение понятия вычислимости в основном осуществлялось и продолжает осуществляться в терминах именно этих языков – «вычислимая функция», «вычислимое множество». Однако не все созданные математиками модели вычислимости для данных понятий оказались приемлемыми и перспективными при компьютерном решении задач. Скажем, в парадигме функционального программирования

(далее – ФП) используются в основном только три модели, формализующие понятие «вычислимость» в терминах вычислимых функций:

- классическая теория рекурсивных функций;
- лямбда-исчисление (типовое и бестиповое);
- комбинаторная логика.

Классическая *модель рекурсивных функций* составляет основу языка РЕФАЛ (РЕкурсивных Функций АЛгоритмический язык), придуманного еще в Советском Союзе в начале 60-х гг. прошлого столетия. Другая функциональная модель вычислимости – *лямбда-исчисление*, или *исчисление лямбда-конверсий*, наиболее популярная в ФП модель, была создана в 1934 г. А. Черчем. Предложенный им *лямбда-язык*, несмотря на его кажущуюся простоту, оказался весьма мощным по своей выразительности, поскольку, как выяснилось, позволяет (согласно знаменитому *Тезису Черча*) описывать все(!) интуитивно вычислимые функции (отметим, что справедливость Тезиса Черча была подкреплена теми математическими фактами, что в лямбда-языке оказались выразимы все функции, определяемые в других альтернативных моделях вычислимости, и наоборот).

Близкую к лямбда-исчислению модель вычислимых функций, носящую название *комбинаторной логики*, в 1940 г. создал Х. Карри. Обе эти модели вычислимости некоторое время развивались как чисто математические теории, и только в начале 1960-х гг. Д. Маккарти создал на основе языка лямбда-исчисления язык функционального программирования ЛИСП (LISP – LISt Processing), а Д. Тернер на базе комбинаторной логики реализовал низкоуровневый язык для описания трансляторов языков функционального программирования. Впоследствии на идеях лямбда-исчисления и комбинаторной логики было создано еще несколько языков, в том числе и функциональные системы программирования (FP-системы), позволяющие работать с функциями высших порядков.

Многообразие созданных к 90-м гг. прошлого столетия языков функционального программирования послужило причиной реализации проекта создания функционального языка, в котором были объединены наиболее интересные идеи ФП. Так, в начале 90-х гг. появился функциональный язык программирования, названный HASKELL в честь Хаскелла Карри. Тем не менее следует признать, что основной вклад в развитие парадигмы ФП внес язык ЛИСП, который на протяжении многих лет оставался наиболее популярным функциональным языком программирования. Этот язык был использован в большом числе исследований и разработок в области ИИ при решении таких задач, как обработка естественного языка, создание экспертных систем,

распознавание образов и многих других. Однако заметим, что, строго говоря, и ЛИСП, и HASKELL не являются чисто декларативными языками, поскольку в них разрешается использовать императивные конструкции, оправдывая это тем, что они позволяют создавать весьма эффективные практические приложения. На практике же это приводит к излишней сложности используемых конструкций, поскольку исчезает их первичная декларативная ясность. Кстати, аналогичный эффект имеет место и при добавлении в императивные языки функциональных средств. Заметим, что попытки создания языка, совмещающего в себе сильные стороны функционального и императивного программирования, продолжаются до сих пор.

Итак, что же предлагает нам современное ФП с позиций декларативного описания задач и процессов их решения. В ФП действия представляются в виде функций, которые одни данные (аргументы) преобразуют в другие данные (значения). При этом, в отличие от императивных языков, значения функций однозначно определяются их аргументами и не зависят как от истории вычислительного процесса, так и от внешних по отношению к вычислительному процессу сигналов и состояний. В то же время имеется и важное отличие программно реализованной функции от ее математического аналога – «программистская» функция должна быть определена *явно*, т. е. таким образом, чтобы всегда была возможность извлечь из ее описания способ ее вычисления. Отсюда, в ФП исходные условия задачи описываются в виде конечного набора явных определений функций, (база данных), а цель задачи ставится в виде запроса на вычисление значений так называемого целевого выражения с заданными параметрами и, возможно, содержащего обращения как к базовым функциям, так и к функциям, определенным в базе данных.

Таким образом, общая схема формулирования и решения задачи в ФП сводится к:

- функциональной спецификации задачи, т. е. к написанию ее исходных условий в виде конечного набора определений функций;
- написанию запроса и заданию его параметров;
- вычислению значений данного запроса.

При этом процесс вычисления с математической точки зрения представляет собой процесс исполнения так называемых правил переписывания, который можно интерпретировать как логический вывод искомого результата из аксиоматической теории, в роли которой выступает набор определений функций.

Похоже, но несколько иначе, осуществляются постановка и решение задач в другой парадигме декларативного программирования –

в так называемом логическом программировании (далее – ЛП). Его математическую основу составляет аксиоматическая модель вычислимости, базирующаяся на понятии вычислимого множества. Логическая программа представляет собой формульное описание вычислимого отношения между параметрами в терминах формального логического языка. При этом разница между параметрами, объявленными входными, и параметрами, рассматриваемыми как выходные, в логической программе достаточно условна, что и отличает парадигму логического программирования от функционального. Дело в том, что вычисление функции всегда является строго направленным процессом – мы подаем на вход значения аргументов функции и на выходе должны получить результат, т. е. вычисленные значения функции. В логических же языках допускаются ситуации, когда вначале указывается значение выходного параметра и ищется такое значение входного, при котором имеет место заданное отношение. Другое важное отличие логического программирования от функционального – это неоднозначность логических вычислений, поскольку удовлетворяющих исходному запросу значений переменных у логической программы может оказаться несколько.

В логическом программировании используется так называемое формульное описание вычисляемых отношений с применением достаточно узкого подмножества языка исчисления предикатов первого порядка. Заметим, что для логической программы, как и для функциональной, также имеют место понятия «база данных» и «запрос». При этом база данных (т. е. исходные условия задачи) в ЛП задается конечным набором логических правил и фактов, записанных в виде логических формул узкого подмножества языка исчисления предикатов, которые рассматриваются как аксиомы предметной области. Запрос также представляет собой логическую формулу этого языка, которая может содержать переменные, а целью решения задачи, специфицированной в виде логической базы данных и формулы-запроса, является попытка либо доказать, что запрос есть *логическое следствие* аксиом, т. е. является *теоремой* данной системы аксиом (с определением желательно всех вариантов означивания переменных в запросе), либо убедиться в том, что запрос не является логическим следствием этой аксиоматической теории.

Первые попытки реализовать отдельные идеи логического программирования делались еще в начале 60-х гг. прошлого столетия. Значительный вклад в становление ЛП внес Дж. Робинсон, который для языка исчисления предикатов первого порядка в 1965 г. разработал алгоритм унификации и на его основе создал метод резолюций.

Используя этот метод, в 1971 г. А. Колмероз и Ф. Руссел создали первую версию языка ПРОЛОГ (Prolog – PROgrammation en LOGique), написанную на ФОРТРАНе. Сразу же отметим, что ПРОЛОГ, как и ЛИСП, является квазидекларативным языком, поскольку в нем также достаточно активно используются императивные конструкции. Основопологающей для становления ЛП как новой парадигмы декларативного программирования стала публикация Р. Ковальским в 1974 г. статьи, в которой он показал, что для целей ЛП достаточно ограничиться использованием лишь *хорновских дизъюнктов*, т. е. формул вида $P_1 \& \dots \& P_n \rightarrow Q$.

Пик популярности идей ЛП пришелся на 80-е гг. прошлого столетия, свидетельством чему явился японский проект создания ЭВМ 5-го поколения. В предшествующие этому проекту и последующие годы намечилось несколько направлений развития ЛП, среди которых выделим следующие:

- *совершенствование и модификация* языка ПРОЛОГ (расширение базы языка, использование более мощных логических средств, постулирование модульности логических программ и т. п.);
- *параллельное* логическое программирование (заметим, что логические программы по своей природе являются параллельными);
- *комбинирование* логического стиля программирования с функциональным.

Остановимся на последнем направлении. Следует отметить, что уже с момента появления логического и функционального языков программирования неоднократно предпринимались попытки объединить эти две декларативные парадигмы. Как правило, главной целью всех предлагаемых вариантов объединения являлась возможность использования более широкого набора стратегий управления вычислениями, например, эффективной стратегией ленивых вычислений, характерной для функционального программирования, или стратегией параллельных вычислений, изначально свойственных логическим программам. Однако серьезной проблемой при подобных объединениях оказалась проблема единообразия используемых структур данных, а также решение проблемы их типизации. Именно по этой причине в объединенных декларативных языках наиболее распространенными структурами данных оказались списки и деревья. Что же касается проблемы типизации данных, то в декларативном программировании реализуются оба возможных варианта ее решения – либо полное игнорирование понятия типа данных, либо, наоборот, следование строгой и жесткой дисциплине их определения и использования. Кстати, заметим, что отсутствие в декларативном

языке понятия типа данных (так называемые *бестиповые языки*) имеет свое преимущество, поскольку в таких языках на множество всех его объектов (данные, действия, программы) можно смотреть как на *единое пространство*. Яркими примерами таких языков являются ЛИСП и ПРОЛОГ, у которых единственной формой представления объектов являются символьные выражения, что и позволяет программе обрабатывать и преобразовывать другие декларативные программы и даже саму себя – знаменитая *проблема самоприменимости*. Отметим, что эта особенность декларативных бестиповых языков активно используется в ИИ и именно поэтому эти языки так популярны в ИИ. Кстати, из-за бестиповости декларативных языков серьезной математической задачей в свое время оказалась задача построения строгой математической семантики для таких языков. Эта проблема была практически одновременно решена в 70-е гг. известными математиками – в России акад. Ю. Л. Ершовым (теория А-пространств Ершова), в Англии – Даном Скоттом (теория информационных систем).

Итак, подведем итог вышесказанному. Ключевая идея декларативного программирования – то обстоятельство, что программа в нем является *теорией*, а вычисления представляют собой *вывод* в этой теории. Понятно, что чем меньше ограничений накладывается на допустимые способы построения таких теорий, тем проще пользователю специфицировать задачу. Но, к сожалению, имеет место и другое наблюдение – чем меньше ограничений, тем менее эффективным будет алгоритм интерпретации теорий и, следовательно, алгоритм организации поиска вывода. Данное противоречие заставляет сторонников и апологетов декларативного программирования искать разумный компромисс между удобством и выразительностью существующих декларативных языков и их эффективностью. Такой поиск может проходить либо в рамках существующих парадигм функционального и логического программирования, либо в их критическом осмыслении и, возможно, создании новой парадигмы.

Как известно, фундаментальным изучением понятия вычислимости в математике занимается математическая логика в рамках двух основных подходов: *аксиоматическом* и *теоретико-модельном*. Как было замечено выше, именно аксиоматический подход составляет математическую основу декларативного программирования. Но ведь в качестве концептуальной основы технологии компьютерного решения задач можно выбрать и теоретико-модельный подход, взяв, скажем, в качестве математической базы возможность *формульной определимости вычислимости на конструктивных моделях*,

а сам процесс вычислимости мыслить как процесс проверки *истинности формулы* на конструктивной модели. Эта идея и была положена в основу концепции *семантического моделирования*, выдвинутой в 80-х гг. прошлого столетия сотрудниками ИМ СО РАН академиками РАН С. С. Гончаровым, Ю. Л. Ершовым и д-ром физ.-мат. наук Д. И. Свириденко [Goncharov, Sviridenko, 1986; 1989; Ershov et al., 1986; 1987; Гончаров, Свириденко, 1986; Ershov, 1983a; 1983b; 1996]. Отметим, что авторы концепции в то время использовали термин «семантическое программирование», поскольку выдвинутая концепция была ориентирована в основном на решение задач управления сложными системами. Однако впоследствии концепция постепенно стала приобретать черты модельного подхода и все больше и больше ориентироваться на решение задач моделирования сложных процессов, а также на задачи обработки больших и потоковых данных. По этой причине было предложено концепцию переименовать в семантическое моделирование. Суть этой концепции заключается в следующем.

1. Предполагается наличие исходной базовой модели, выступающей как ядро системы, средствами которой будут специфицироваться и решаться задачи. При этом предполагается, что все объекты данной модели (элементы, функции, предикаты) и операции над ними являются *конструктивными*, т. е. вычислимыми [Ershov, Goncharov et al., 1998; Ambos-Spies et al., 2011; Goncharov, 2007]. Заметим, что в семантическом моделировании базовая модель рассматривается вместе со своей *списочной надстройкой*, состоящей из наследственно-конечных списков, порожденных элементами этой модели. Допускается интерпретация некоторых предикатов базовой модели как *оракулов*.

2. Исходным условиям задачи сопоставляется смешанное формульно-термальное, т. е. фактически логико-функциональное определение новой модели той предметной области, к которой относится решаемая задача, с использованием так называемых Σ -*формул* и Σ -*термов* языка исчисления предикатов первого порядка сигнатуры исходной базовой модели. При этом отметим, что в отличие от классического подхода термальная часть используемого Σ -языка значительно расширена путем допущения *условных* и *рекурсивных* термов [Гончаров, 2017; Гончаров, Свириденко, 2018]. Допускается также рекурсивное описание не только новых функций, но и определяемых предикатов.

3. В качестве ответа на формулу-запрос или терм-запрос, записанные либо как Σ -формула, либо как Σ -терм, алгоритмически ищутся в первом случае такие конкретизации свободных переменных фор-

мулы, для которых данная Σ -формула является *истиной*, или вычисляется значение Σ -терма-запроса.

Сразу же отметим, что на практике предлагается ограничиться классом так называемых Δ_0 -*формул* и Δ_0 -*термов*. Напомним, что отличие Δ_0 -формул от Σ -формул заключается в том, что при написании Δ_0 -формул разрешается использовать только *ограниченные кванторы* существования и всеобщности. Причина такого ограничения заключается в желании с самого начала ограничиться эффективными вычислениями, т. е. вычислениями заданной, например, полиномиальной или автоматной сложности [Arora, Barak, 2009; Papadimitriou, 1994; Ospichev, Ponomarev, 2018].

Итак, в предлагаемом подходе главными выступают процесс формульно-термального описания некой проблемной области в виде конструктивной модели и процедура либо проверки истинности Σ -формулы-запроса на этой модели, либо вычисления Σ -терма в этой модели. Практика показала, что такой модельный подход позволяет вполне адекватно отразить и полностью сохранить исходную семантику задачи. Именно поэтому он и называется *семантическим моделированием*.

Концепция семантического моделирования получила свое воплощение в целой серии теоретических и прикладных работ. В настоящее время продолжается дальнейшее развитие ее теоретических и прикладных аспектов, в том числе в направлении создания практических технологий семантического моделирования и их адаптации применительно к особенностям различных конкретных проблемных областей – ритейл, финтех, медицина, техника, умные контракты и кошельки, криптоэкономика, теория решения инновационных задач (далее – ТРИЗ) и др. [Малых, Манцивода, 2017; Martynovich, Vityaev, 2016; Vityaev, Martinovich, 2015; Витяев, Неупокоев, 2014; Витяев, Мартынович, 2015; 2017; Свириденко, Сибирияков, 2017а; 2017б].

Одним из главных направлений развития семантического моделирования в настоящее время является его применение к проблемам ИИ. В результате исследований в этом направлении возникла альтернативная традиционному искусственному интеллекту концепция «объясняющего» *искусственного интеллекта* (*Explainable Artificial Intelligence*), целью которого является создание таких технологий ИИ, в том числе и технологий обучения компьютеров, которые базировались бы на исходной семантике решаемой задачи и позволяли бы автоматически синтезировать семантически обусловленное и понятное человеку ее решение, при этом автоматически адаптирующиеся к изменению условий решения этой задачи. И такие технологии, основан-

ные на оригинальных математических результатах, обобщающих логико-математический и вероятностный анализ формальных понятий, были созданы [Martynovich, Vityaev, 2016; Vityaev, Martinovich, 2015; Витяев, Неупокоев, 2014; Витяев, Мартынович, 2015; 2017]. При этом полученные математические результаты позволили не только создавать эффективные методы решения задач ИИ, свободные от указанных ранее недостатков традиционных технологий ИИ, но и, что очень важно для дальнейшего развития ИИ, разработать уникальный подход к решению проблемы *самообучения* систем ИИ.

Что касается практического применения концепции «объясняющего» ИИ, то успешными примерами здесь могут служить работы по автоматизации и глубокой роботизации бизнес-отношений, а также работы в области умных контрактов и умных кошельков нового поколения, осуществляемых ИМ СО РАН совместно с иркутскими коллегами из Иркутского государственного университета и группой компаний АЙЛАЙН (см., например, проекты Bsystem¹ и Kirik²). С помощью семантического моделирования и методов «объяснительного» искусственного интеллекта в настоящее время решается также широкий круг и других теоретических и прикладных проблем, например, решение совместно с Новосибирским государственным университетом проблем анализа сложных и масштабных коммуникационных сетей и платежных систем.

Важно отметить, что семантический подход позволяет, в отличие от аксиоматического варианта, осуществлять эффективный синтез решений задач заранее оговоренной сложности, например, полиномиальной или автоматной. Более того, в рамках семантического моделирования можно ставить и решать задачу создания «почти везде» эффективных алгоритмических решений, скажем, алгоритмов автоматной или полиномиальной сложности на «почти всех» исходных данных. Эти исследования, как и многие другие, носят уникальный характер.

Значительным преимуществом семантического моделирования перед другими концепциями математического моделирования является возможность в рамках единого подхода объединять идеи, концепции, модели и алгоритмы *непрерывной* и *дискретной* математики. Такой эффект достигается путем использования в качестве сигнатурных сущностей базовой конструктивной модели алгоритмов непрерывной математики, которые могут выступать как своеобраз-

¹ <http://www.inbox.io>

² www.kirik.io

ные *оракулы*, над которыми выстраивается логико-математическая надстройка, описывающая логику решения задач из той или иной предметной области. Такой подход позволяет, например, по-новому подойти к решению такой сложной проблемы, как управление гибридными системами.

И еще одно важное преимущество предлагаемого семантического подхода – он не отрицает использования в нем элементов аксиоматического подхода. Более того, в теории и практике семантического моделирования активно используется возможность эффективного синтеза обоих подходов – теоретико-модельного и аксиоматического. Так, например, при проведении исследований по объяснительному ИИ выяснилось, что результаты, полученные в рамках данного синтетического подхода, дают нам значительное преимущество перед чисто математико-статистическим или чисто аксиоматическим подходами, поскольку позволяют вместо традиционного машинного (глубокого) обучения типа «черный ящик» предложить принципиально другой подход к проблеме машинного обучения, основанный на сочетании математико-статистического и логического подходов (см., например: [Martynovich, Vityaev, 2016; Vityaev, Martinovich, 2015; Витяев, Непокоев, 2014; Витяев, Мартынович, 2015; 2017]). Такое сочетание позволяет строить иерархию логико-вероятностных классов, которые, с одной стороны, легко *понимаемы* человеком как совокупность логико-вероятностных утверждений, а с другой – более точно, на основе самообучения, формируют иерархию «естественных» классов, тем самым теоретически, по своим последствиям, намного превосходящих глубокое обучение, основанное на традиционных нейронных сетях. Проведенные предварительные эксперименты полностью подтвердили этот теоретический вывод. Более того, эти эксперименты показали также, что такое семантически-«прозрачное» машинное (глубокое) обучение адекватно формализует не только «естественную» классификацию, но и даже процесс реального восприятия, описанный в психологической литературе.

Последнее обстоятельство позволяет вплотную подойти к решению труднейшей проблемы ИИ – проблемы *саморефлексии* систем ИИ. Заметим, что успешное решение этой проблемы даст возможность говорить о достижении искусственным интеллектом следующего уровня развития, который можно условно назвать *искусственным интеллектом 3.0* (если считать, что развиваемый нами «объясняющий» ИИ – это *искусственный интеллект уровня 2.0*). Именно на уровне 3.0 развития технологий ИИ появится возможность создавать системы ИИ, умеющие не только ставить перед собой

задачи и осуществлять поиск их решения, но и, что принципиально важно, *объяснять* и *обосновывать* найденные решения. Отдельные элементы ИИ 3.0 в рамках семантического подхода создаются уже сейчас. Есть уверенность в том, что уже в течение ближайших 10–15 лет будет получено решение таких важнейших проблем, как проблема *самоидентификации* и *самоприменимости*, включая решение проблемы *целеполагания*, ведущих к удовлетворительному решению проблемы *саморефлексии*.

В заключение отметим, что активное развитие концепции семантического моделирования не только повлекло за собой решение как новых, так и давно стоящих перед ИИ проблем, о которых шла речь выше, но и оказало значительное влияние на такие смежные научные и практические области, как поисковые системы, автоматизация бизнес-процессов, смарт-контракты и смарт-кошельки, робототехника, гибридные системы, ТРИЗ, онтологическое моделирование и многие другие, что говорит о ее значительном научном и прикладном потенциале, особенно в связи с предстоящей цифровизацией экономики.

Список литературы

Братко И. Программирование на языке ПРОЛОГ для искусственного интеллекта. М.: Мир, 1990.

Витяев Е. Е., Мартынович В. В. Прозрачное глубокое обучение на основе вероятностных формальных понятий в задаче обработки естественного языка // *Изв. ИГУ. Серия: Математика*. 2017. Т. 22. С. 31–49.

Витяев Е. Е., Мартынович В. В. Формализация «естественной» классификации и систематики через неподвижные точки предсказаний // *Сибирские электронные математические известия (Siberian Electronic Mathematical Reports)*. Новосибирск, 2015. Т. 12. С. 1006–1031.

Витяев Е. Е., Неупокоев Н. В. Формальная модель восприятия и образа как неподвижной точки предвосхищений // *Подходы к моделированию мышления: Сб. / Под ред. В. Г. Редько*. М.: УРСС Эдиториал, 2014. С. 155–172.

Гончаров С. С. Условные термы в семантическом программировании // *Сиб. мат. журн.* 2017. Т. 58, № 5. С. 1026–1034.

Гончаров С. С., Свириденко Д. И. Математические основы семантического программирования // *Докл. АН СССР*. 1986. Т. 289, № 6. С. 1324–1328.

Гончаров С. С., Свириденко Д. И. Рекурсивные термы в семантическом программировании // Сиб. мат. журн. Ноябрь-декабрь, 2018. Т. 59, № 6. С. 1279–1290.

Зюзысов В. М. Математическое введение в декларативное программирование: Учеб. пособие. Томск: Изд-во ТГУ, 2003. 83 с.

Логическое программирование: Сб. ст. / Под ред. В. Н. Агафонова. М.: Мир, 1988. 368 с.

Малпасс Д. Р. Реляционный язык ПРОЛОГ и его применение. М.: Наука, 1990.

Малых А. А., Манцивода А. В. Документное моделирование // Изв. ИГУ. Серия: Математика, 2017.

Прайт Т., Зелковиц М. Языки программирования: разработка и реализация. 4-е изд. СПб.: Питер, 2003.

Рассел С., Норвиг П. Искусственный интеллект. Современный подход. 2-е изд. / Пер. с англ. М.: Изд. дом «Вильямс», 2007. 1408 с.

Свириденко Д. И., Сибиряков В. Г. ТРИЗ-теория решения инновационных задач. Ч. 1: Как решать инновационные задачи: разработка концепции инновации // Сибирская финансовая школа. Инновации. 2017а. № 3/122. С. 26–35.

Свириденко Д. И., Сибиряков В. Г. ТРИЗ-теория решения инновационных задач. Ч. 2: Что такое инновационная задача // Сибирская финансовая школа. Менеджмент и инновации. 2017б. № 4/123. С. 21–37.

Стерлинг Л., Шапиро Э. Искусство программирования на языке ПРОЛОГ. М.: Наука, 1990.

Тейз А., Грибомон П., Луи Ж. и др. Логический подход к искусственному интеллекту: от классической логики к логическому программированию: Пер. с фр. М.: Мир, 1990. 432 с.

Филд А., Харрисон П. Функциональное программирование: Пер. с англ. М.: Мир, 1993.

Хендерсон П. Функциональное программирование. Применение и реализация: Пер. с англ. М.: Мир, 1983. 349 с.

Хювенен Э., Себянен И. Мир ЛИСПа: В 2 т. М.: Мир, 1990.

Ambos-Spies K., Badaev S., Goncharov S. Inductive inference and computable numberings // Theoretical Computer Science. 2011. Vol. 412, No. 18. P. 1652–1668.

Arora S., Barak B. Computational Complexity: a Modern Approach. Cambridge Univ. Press, 2009.

Ershov Yu. L. Definability and Computability. Siberian School of Algebra and Logic. New York: Plenum, 1996.

Ershov Yu. L. Dynamic logic over admissible sets // Soviet. Math. Dokl. 1983b. Vol. 28. P. 739–742.

Ershov Yu. L. The principle of Σ -enumeration // Soviet. Math. Dokl. 1983a. Vol. 27. P. 670–672.

Ershov Yu. L., Goncharov S. S., Nerode A., Remmel J. Introduction to the Handbook of Recursive Mathematics, Handbook of Recursive Mathematics, Vol. 1–2, Amsterdam ets.: Elsevier, 1998. Vol. 138, Pt 1, vii–xlv, 40 p.

Ershov Yu. L., Goncharov S. S., Sviridenko D. I. Semantic foundations of programming // Lecture Notes in Computer Science. 1987. Vol. 278. P. 116–122.

Ershov Yu. L., Goncharov S. S., Sviridenko D. I. Semantic programming // Information processing, Proc. IFIP 10-th World Comput. Congress. Dublin, 1986. Vol. 10. P. 1113–1120.

Flasinski M. Introduction to Artificial Intelligence. Switzerland: Springer International Publ., 2016. 321 p.

Goncharov S. S. Computability and Computable Models // Mathematical Problems from Applied Logic II International Mathematical Series / Eds. D. M. Gabbay, M. Zakharyashev, S. S. Goncharov. Springer, 2007. Vol. 5. P. 99–216.

Goncharov S. S., Sviridenko D. I. Theoretical aspects of Σ -programming // Lecture Notes in Computer Science. 1986. Vol. 215. P. 169–179.

Goncharov S. S., Sviridenko D. I. Σ -programming, Transl., II. Ser. // Am. Math. Soc. 1989. Vol. 142. P. 101–121.

Kubat M. An Introduction to Machine Learning. 2 Ed.; Springer Int. Publ. AG, 2017. 348 p.

Martynovich V. V., Vityaev E. E. Recovering Noisy Contexts with Probabilistic Formal Concepts // Proceedings of the 2nd International Workshop on Soft Computing Applications and Knowledge Discovery (SCAKD 2016). Moscow, Russia, July 18, 2016. CEUR Workshop Proceedings. Vol. 1687. P. 24–35.

Ospichev S., Ponomarev D. On the Complexity of Formulas in Semantic Programming // Сибирские электронные математические известия. 2018. № 15. P. 987–995.

Papadimitriou C. H. Computational complexity. Addison-Wesley, 1994.

Vityaev E. E., Martynovich V. V. Probabilistic Formal Concepts with Negation / Eds. A. Voronkov, I. Virbitskaite. PCI 2014, LNCS 8974, 2015. P. 385–399.

S. S. Goncharov, D. I. Sviridenko

*Sobolev Institute of Mathematics SB RAS
4 Koptyug Ave., Novosibirsk, 630090, Russian Federation*

*Novosibirsk State University
1 Pirogov Str., Novosibirsk, 630090, Russian Federation*

gonchar@math.nsc.ru; dsviridenko47@gmail.com

SEMANTIC MODELING AND ARTIFICIAL INTELLIGENCE

The paper discusses the problem of solving human problems using computers. Analyzed are the advantages and disadvantages of declarative programming, including functional and logical programming, acting as an artificial intelligence tool. An alternative, model-theoretic approach to solving problems, called the semantic modeling, is described. The advantages of the proposed concept are analyzed, including the possibility of combining the axiomatic and model-theoretic approach to solving problems in a single approach, as well as the possibility of integrating on the basis of semantic modeling methods of continuous and discrete mathematics. The possibility of constructing a new, «explanatory artificial intelligence free from the drawbacks inherent in the traditional artificial intelligence, based on semantic modeling, is described.

Keywords: artificial intelligence, machine learning, semantic modeling, declarative programming, computability, model-theoretic approach.

References

Ambos-Spies K., Badaev S., Goncharov S. Inductive inference and computable numberings. *Theoretical Computer Science*, 2011, vol. 412, no. 18, p. 1652–1668.

Arora S., Barak B. *Computational Complexity: a Modern Approach*. Cambridge Univ. Press, 2009.

Bratko I. *Programming in the PROLOG language for artificial intelligence*. Moscow, Mir Publ., 1990. (In Russ.)

Ershov Yu. L. *Definability and Computability. Siberian School of Algebra and Logic*. New York, Plenum, 1996.

Ershov Yu. L. Dynamic logic over admissible sets. *Soviet. Math. Dokl.*, 1983b, vol. 28, p. 739–742.

Ershov Yu. L. The principle of Σ -enumeration. *Soviet. Math. Dokl.*, 1983a, vol. 27, p. 670–672.

Ershov Yu. L., Goncharov S. S., Nerode A., Remmel J. Introduction to the Handbook of Recursive Mathematics, Handbook of Recursive Mathematics, Vol. 1–2, Amsterdam et. al.: Elsevier. Vol. 138, Part 1, 1998, p. vii–xlvi, 40 ff.

Ershov Yu. L., Goncharov S. S., Sviridenko D. I. Semantic foundations of programming. *Lecture Notes in Computer Science*, 1987, vol. 278, p. 116–122.

Ershov Yu. L., Goncharov S. S., Sviridenko D. I. Semantic programming. *Information processing*, Proc. IFIP 10-th World Comput. Congress. Dublin, 1986, vol. 10, p. 1113–1120.

Field A., Harrison P. *Functional programming*: Trans. with English. Moscow, Mir Publ., 1993. (In Russ.)

Flasinski M. *Introduction to Artificial Intelligence*. Springer International Publ., Switzerland, 2016.

Goncharov S. S. Computability and Computable Models. *Mathematical Problems from Applied Logic II International Mathematical Series*. Dov M. Gabbay, M. Zakharyashev, S. S. Goncharov (eds.), Springer, 2007, vol. 5, p. 99–216.

Goncharov S. S. Uslovnnye termny v semanticheskom programmirovanii [Conditional terms in semantic programming]. *Siberian Mathematical Journal*, 2017, vol. 58, no. 5, p. 1026–1034. (In Russ.)

Goncharov S. S., Sviridenko D. I. Mathematical principles of semantic programming. *Doklady Akademii Nauk SSSR [Reports of the USSR Academy of Sciences]*, 1986, vol. 289, no. 6, p. 1324–1328). (In Russ.)

Goncharov S. S., Sviridenko D. I. Rekursivnye termny v semanticheskom programmirovanii [Recursive terms in semantic programming]. *CSF*, 2018, vol. 59, no. 6, p. 1279–1290. (In Russ.)

Goncharov S. S., Sviridenko D. I. Theoretical aspects of Σ -programming. *Lecture Notes in Computer Science*, 1986, vol. 215, p. 169–179.

Goncharov S. S., Sviridenko D. I. Σ -programming. Transl., II. Ser., *Am. Math. Soc.*, 1989, vol. 142, p. 101–121.

Henderson P. *Functional programming. Application and implementation*: Trans. with English. Moscow, Mir Publ., 1983. (In Russ.)

Huvenen E., Sepyanen I. *The World of LISP*: In 2 vols. Moscow, Mir Publ., 1990. (In Russ.)

Kubat M. *An Introduction to Machine Learning*. 2 Ed.; Springer International Publ. AG. 2017.

Logical programming. Digest of articles. V. N. Agafonov (ed.). Moscow, Mir Publ., 1988. (In Russ.)

Malpass D. R. *Relational language PROLOG and its application*. Moscow, Nauka, 1990. (In Russ.)

Malykh A. A., Mantsivoda A. V. Document modeling. *Izvestiya ISU [News of ISU. Series: Mathematics]*, 2017. (In Russ.)

Martynovich V. V., Vityaev E. E. Recovering Noisy Contexts with Probabilistic Formal Concepts. *Proceedings of the 2nd International Workshop on Soft Computing Applications and Knowledge Discovery (SCAKD 2016)*, Moscow, Russia, July 18, 2016. CEUR Workshop Proceedings, vol. 1687, p. 24–35.

Ospichev S., Ponomarev D. A Note on the Complexity of Formulas in Semantic Programming. *Sibirskie elektronnye matematicheskie izvestiya [Siberian Electronic Mathematical Reports]*, 2018, no. 15, p. 987–995. (In Russ.)

Papadimitriou C. H. *Computational complexity*. Addison-Wesley, 1994.

Pratt T., Zelkovitz M. *Programming languages: development and implementation*. 4 th ed. St. Petersburg, Peter Publ., 2003. (In Russ.)

Russell S., Norvig P. *Artificial Intelligence. A Modern Approach*. Second ed.: Prentice Hall, Upper Saddle River New Jersey, 2003. 1408 p. (In Russ.)

Sterling L., Shapiro E. *The Art of Programming in the language PROLOG*. Moscow, Nauka, 1990. (In Russ.)

Sviridenko D. I., Sibiryakov V. G. TRIZ-teoriya resheniya innovatsionnykh zadach. Chast 1: [TRIZ-Theory of Innovative Problems Solution: Part 1: How to solve innovative problems: the development of the concept of innovation]. *Siberian financial school. Innovations*, 2017a, no. 3/122, p. 26–35. (In Russ.)

Sviridenko D. I., Sibiryakov V. G. TRIZ-teoriya resheniya innovatsionnykh zadach. Chast 2: Chto takoe innovatsionnaya zadacha [TRIZ-Theory of Innovation Solutions: Part 1. What is an Innovative Task]. *Siberian financial school. Management and Innovation*, 2017b, no. 4/123, p. 21–37. (In Russ.)

Teiz A., Gribomon P., Louis J. et al. *Logicheskii podhod k iskusstvennomu intellektu: ot klassicheskoi logiki k logicheskomu programmirovaniyu [A Logical Approach to Artificial Intelligence: from Classical Logic to Logical Programming]*. Moscow, Mir Publ., 1990. (In Russ.)

Vityaev E. E., Martynovich V. V. Probabilistic Formal Concepts with Negation. A. Voronkov, I. Virbitskaite (eds.). PCI 2014, LNCS 8974, 2015, p. 385–399.

Vityaev E. E., Martynovich V. V. Formalization of the «natural» classification and taxonomy through fixed points of predictions. *Siberian Electronic Mathematical Reports*. S. L. Sobolev Institute of Mathematics of the SB RAS, 2015, vol. 12, p. 1006–1031. (In Russ.)

Vityaev E. E., Neupokoev N. V. Formalnaya model vospriyatiya i obraza kak nepodvizhnoi tochki predvoskhishchenii [Formal model of perception

and image as a fixed point of anticipation]. *Podkhody k modelirovaniyu myshleniya* [Approaches to the Modeling of Thinking] (a collection edited by Doctor of Physical and Mathematical Sciences VG Redko). Moscow, URSS Editorial, 2014, p. 155–172. (In Russ.)

Vityaev V. V., Martynovich V. V. Transparent profound learning based on probabilistic formal concepts in the task of processing natural language. *Izvestiya ISU* [News of ISU. Series: Mathematics]. 2017, vol. 22, p. 31–49. (In Russ.)

Zyuzysov V. M. *Matematicheskoe vvedenie v deklarativnoe programmirovaniye*: Uchebnoe posobie [Mathematical Introduction to Declarative Programming: a Tutorial]. Tomsk, TSU Publ., 2003. (In Russ.)