ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ И ИХ ЭЛЕМЕНТЫ/COMPUTING SYSTEMS AND THEIR ELEMENTS

НЕПОЛНЫЙ АЛГОРИТМ В КОНСТРУКТИВНОЙ МАТЕМАТИКЕ (ЧАСТЬ 1)

Научная статья

Коновалов В.А.[1, *]
[1] ORCID : 0000-0002-3819-0378;
[1] ООО "Курский мясоперерабатывающий завод", Железногорск, Российская Федерация

* Корреспондирующий автор (vk546[at]yandex.ru)

Аннотация

Представлены результаты исследования проблемы неполноты алгоритмов. Введено новаторское определение «неполный алгоритм», отличающееся от известной в науке неполноты по Тьюрингу. Исследование состоит из двух частей, где в первой части представлены результаты анализа теоретико-типовых причин неполноты алгоритма, во второй части, представлен способ порождения полного алгоритма. Устанавливается одна из коренных причин появления «проблемы неполноты алгоритмов», известная в науке и включенная в список Проблем Стивена Смейла и Проблем тысячелетия, сформулированных Математическим институтом Клэя как проблема: «равенства классов P и NP» (проблема перебора)», показан один из способов определения равенства или неравенства этих классов. Подвергаются критическому анализу теоретико-типовые подходы, а также современные системы типов, применяемые в информатике и приводящие алгоритм к неполноте, а также сопутствующим проблемам синтеза типов для потенциального алгоритма искусственного интеллекта. Рассматриваются типические последовательности, а также известный логический тип (boolean) при их переработке алгоритмом. Подвергаются анализу гетерогенные композиции типов. Прилагается к теории типов тезис Э. Бишопа: «Не спрашивай, истинно ли утверждение, пока не знаешь, что оно означает», который ставит под сомнение идею использования типов, перерабатываемых только дедуктивными методами и рассматривается задача типизации индуктивных наблюдений в алгоритме. Выделены и классифицированы способы, составляющие подхода к формализации неполного алгоритма. Делается вывод о том, что при синтезе алгоритма ИИ, на первом этапе его формализации, целесообразно ввести ограничения на входные данные: таковые должны быть словами и морфизмами нетипическими последовательностями.

Ключевые слова: теория типов, теория алгоритмов, теория категорий, искусственный интеллект, конструктивная математика.

INCOMPLETE ALGORITHM IN CONSTRUCTIVE MATHEMATICS (PART 1)

Research article

Konovalov V.A.[1, *]
[1] ORCID : 0000-0002-3819-0378;
[1] LLC "Kurskiy Myasopererabatyvayushij Zavid", Zheleznogorsk, Russian Federation

* Corresponding author (vk546[at]yandex.ru)

Abstract

The results of the study of the problem of incompleteness of algorithms are presented. An innovative definition of "incomplete algorithm" is introduced, which differs from the Turing incompleteness known in science. The study consists of two parts, where the first part presents the results of the analysis of the type-theoretic causes of the incompleteness of algorithms, and the second part presents a method for generating a complete algorithm. All reasons for the incompleteness of algorithms are of scientific interest. Two main reasons are considered. In both parts of the study, the problem of "equalities of classes P and NP" (enumeration problem), which is well-known in science and included in the list of Stephen Smale Problems and Millennium Problems formulated by the Clay Mathematics Institute, is shown. One of the ways to determine the equality or inequality of these classes is presented. However, it is shown that for this problem it is possible to determine several reasons for its occurrence: type-theoretic, arising from known theories of types and "observer limitations". Therefore, in the first part of the study, type-theoretic approaches are subjected to critical analysis, as well as modern type systems used in computer science and leading the algorithm to incompleteness, as well as related problems of type synthesis for a potential algorithm of artificial intelligence, in the second part of the study, "observer limitations" are considered. Typical sequences are considered, as well as a known logical type (boolean) when they are processed by an algorithm. Heterogeneous compositions of types are analyzed. The theory of types is supplemented with the thesis of E. Bishop: "Do not ask whether a statement is true until you know what it means", which questions the idea of using types processed only by deductive methods, and the problem of typification of inductive observations in the algorithm is considered. The methods that make up the approach to formalizing an incomplete algorithm are identified and classified. It is concluded that when synthesizing an AI algorithm, at the first stage of its formalization, it is advisable to introduce restrictions on the input data: such data should be words and morphisms of atypical sequences.

Keywords: type theory, theory of algorithms, category theory, artificial Intelligence, constructive mathematics.

Введение

Представлены результаты анализа фундаментальных причин неполноты алгоритмов, вытекающей из теоретико-типовых научных подходов и применяемых в информатике формализаций типов. В исследовании реализуется конструктивный подход в математике ленинградской (советской) школы к теории алгоритмов, дополненный

некоторыми соображениями Э. Бишопа. Дается определение понятию — «неполной алгоритм». Вводится допущение о классах P и NP — классах задач, как об алгоритме полном или неполном, для определенного технологического процесса, в объектах окружающего нас мира, иначе «задаче» в прикладном смысле.

Это допущение сводится к принятию во внимание единственного такого алгоритма из всего множества алгоритмов, которое можно синтезировать для заданного технологического процесса, так как «все возможные алгоритмы» понимаются здесь как неконструктивное допущение, которое игнорируется, т.е. классы задач: «P и NP», перерабатываемые алгоритмами, представляются также алгоритмами.

Это возможно, если принять во внимание допущение о разделении алгоритмов на классы, например, порождающие и перерабатывающие алгоритмы, тогда одним (но не единственным) из порождающих алгоритмов может являться некая «задача», понимаемая в прикладном смысле. Подвергается анализу теоретико-типовая причина появления «проблемы неполноты алгоритмов» — *центральная проблема* теории алгоритмов, известная в науке и включенная в список Проблем Стивена Смейла и Проблем тысячелетия, сформулированных Математическим институтом Клэя, как проблема: «равенства классов P и NP» (проблема перебора)». Показан один из теоретико-алгоритмических способов определения равенства или неравенства этих классов, представленных как порождающие алгоритмы.

Теория типов, в настоящее время, активно применяется совместно с теорией алгоритмов, в прикладных задачах информатики, поэтому дает пищу для новых научных выводов. Анализу подвергается причины «несовершенства» известных формализаций алгоритма, вытекающие из теоретико-типовых подходов. Интерес вызывают все аспекты «негативного» влияния типов на алгоритмы. Осуществляется поиск фундаментальных теоретико-типовых проблем алгоритмов.

В настоящем исследовании принимается во внимание проблема: «вычисления пределов искусственного и человеческого интеллектов» из списка Проблем Стивена Смейла и Проблем тысячелетия, сформулированных Математическим институтом Клэя, где математическая абстракция «искусственный интеллект» (ИИ) понимается через известные работы, посвященные тесту Тьюринга, а также исследования ученых Р.Бенерджи (Ranan Bihari Banerji), Джона Сёрля, Герберта Саймона, Аллана Ньюэлла (данные работы полагаются известными).

Учитываются положения: простой теории типов (Simple Type Theory, STT) Чвистека и Рамси, теории чистых систем типов (Pure type systems, PTS) Берарди и Терлоу, интуиционистской теории типов (Intuitionistic Type Theory, ITT) [1] П. Мартина-Лёфа, гомотопической теории типов (homotopy type theory, HoTT) [2] Воеводского В.А., высшей теории типов (Higher Type Theories, HTT).

Рассматривается негативное влияние утилитарного использования типов в современных языках программирования в информатике, входящего в разрез с теорий типов, предполагающей введение типов абстрактно для: объектов, композиций объектов («классов»), морфизмов, композиций морфизмов, управляющих команд.

Типы в современной информатике представимы и, как правило, представляются словами естественного, искусственного или комбинированного словаря, в каком-либо алфавите. Морфизмы и управляющие команды каналов управлений, также представимы и представляются словами в таких же словарях, по существу являясь такими же типами, часто лежащими за пределами языков программирования в информатике (существующими отдельно и независимо). Поэтому к типам предложено приложить алгоритм, перерабатывающий слова, так как в науке известна теория и алгоритм, способная на такую переработку, то именно они и использованы в настоящем исследовании. Это теория алгоритмов Маркова и его известный алгоритм.

Колмогоров А.Н. в работе [3, С. 3] указывал на переработку только слов алгоритмом Маркова, как на «недостаток» его подхода. Этот «недостаток» преодолим. В частности, в работе [4] показана схема алгоритма Маркова, которая способна перерабатывать: слова, морфизмы и команды управления. В этих исследованиях используется специальный термин «тип морфизма» и переменная $type_K$, способ переработки, которой известен каналу управления, для схемы алгоритма. В ходе исследований обнаружены и разделены три новаторских варианта схем алгоритма Маркова: *N*, *cN*, *cNnet*, которые отличаются от *y*-схемы нормального алгоритма Маркова выводом, который нельзя считать нормальным по Маркову.

Синтез новаторских схем: *N, cN, cNnet* алгоритма Маркова стало возможным путем переопределения марковской стрелки «$\rightarrow$», называемой *y*-знак препинания, нормального вывода в дополнительном служебном алфавите *M*, введенном вместо известного служебного алфавита Маркова: $B = \varepsilon\eta\gamma$, где ε — обычные, η — заключительные.

Теоретически предполагается, что выразительная сила *N, cN, cNnet*–схем алгоритма Маркова потенциально обеспечивает возможность переработки всех известных типов. *Но так ли это?* Безусловно типы и типы морфизмов могут быть приложены и к алгоритму ИИ, однако возникает вопрос: *«На сколько таковые применимы в нем?»*. Сам алгоритм ИИ находится в настоящее время в процессе формализации, идет активный сбор научных данных о путях и походах к его синтезу. Вместе с тем нет его должной теории, все теоретические и практические вопросы пока открыты, ведется дискуссия по всему спектру, сопровождающих его, научно-технический решений.

В настоящем исследовании, типы и типы морфизмов дополнительно доопределены в рамках подходов известных теорий: алгоритмов Маркова [5], категорий и топосов, что позволяет рассматривать их, с одной стороны, как алгоритмы, с другой стороны, как категории. Категории могут быть кодированы аристотелевыми категориями, поэтому, если такое кодирование применить, то можно выдвинуть некоторые гипотезы и об алгоритме ИИ.

Рассматриваются исследовательские задачи:

1) проверки возможности *N, cN, cNnet*–схем алгоритма Маркова переработать известные типы, выполнив при этом все предписания алгоритма;

2) проверки известных типов на беззаконность по Л.Э.Ю. Брауэру;

3) обнаружения творческих последовательностей, образовавшихся из композиций типов, в результате переработки алгоритмом неоднородных (гетерогенных), входных выборочных совокупностей данных множества источников;

4) научного обоснования необходимости и целесообразности новаторства в математическом определении «типа», в частности, «типа морфизма»;

5) проверки топосных классификаторов и канала управления *N, cN, cNnet*–схем алгоритма Маркова на пригодность к полному раскодированию известных типов.

Целями исследования: проверка типов на возможность переработки алгоритмом и поиск научно-технических путей формализации алгоритма ИИ. Объектом исследования положим «типы морфизмов».

Перечисленные научные работы и теории необходимы для достижения прикладной цели исследования: поиска путей синтеза алгоритма ИИ. Главной прикладной задачей является поиск формы алфавита Маркова для алгоритма ИИ. Частными прикладными задачами, при этом, являются синтез алфавитов Маркова:

– слов и служебного;

– полного перечня морфизмов композиции алгоритмов ИИ;

– определение способа синтеза типов этой же композиции.

Поэтому, в настоящем исследовании, известные проблемы теорий: алгоритмов и типов, рассматриваются прежде всего, как препятствующие синтезу алгоритма ИИ, поэтому изыскиваются научно-технические пути их разрешения через анализ коренных причин их возникновения.

Исследование представлено в двух частях. В первой части представлены результаты анализа теоретико-типовых причин неполноты алгоритма. Во второй части, представлен способ порождения полного алгоритма.

Постановка задачи

2.1. Разъяснения

В исследовании используется результат, полученный в работе [4], а именно, введенный в научный оборот — «тип морфизма». Понятие — «тип морфизма» математически определено в дополнении к известным положениям теории алгоритмов Маркова. Это понятие используется совместно с «альтернативным» способом определения — «типа» *объекта*.

Поэтому в подходах, принятых в известных теориях типов и исследовании [4], реализуются различные «базовые» допущения о том, что такое «тип», точнее к чему именно прилагать это понятие к *объекту* или морфизму. Такие допущения и сама такая постановка вопроса не приводят к противоречиям теорий алгоритмов и типов, поэтому допустимы. Однако очевидно, что подходы достаточно далеко отстоят друг от друга, так как сам *объект* определен в них по-разному. Наличие и принятие таких двух подходов проходят «красной линией» настоящего исследования. Поэтому, когда речь здесь заходит о типе, то всегда понимается теоретико-типовая конструкция, принятая в одной из теорий типов, когда необходимо отдельно выделить какое-либо отличие, вытекающее из введенного понятия «тип морфизма», то это показывается отдельно.

Предложение о пересмотре теорий типов здесь не вводится. Переход к подходу с «типом морфизма» в теоретических рассуждениях осуществляется только в контексте какого-либо замечания.

2.2. Теоретико-типовая проблема алгоритма

Говоря о теории типов, первым делом, необходимо определить, что такое «типичность». Сделаем это через известные в науке «типические последовательности», это важный термин, так как позволяет понять, что есть «нетипичность», до того, как наделить какой-либо объект, например, конкретизированную последовательность символов (алфавита) такими свойствами.

Примем определение Колмогорова А.Н. и Успенского В.А., показанное как «Типические последовательности: определение» в работе [6, С. 429]: «В сущности, типичность означает принадлежность к каждому разумному большинству» и [6, С. 429]: «Правильным уточнением будет служить некоторый естественный алгоритмический аналог понятия множества меры 1. Формулировку такого аналога предложил Мартин-Лёф».

Примечание: речь идет о работе Мартин-Лёф П. О понятии случайной последовательности. — Теория вероятн. и ее примен., 1966, т. X I, в. 1, с. 198–200. Далее, авторы: Колмогоров и Успенский, в той же работе [6, С. 430] дают более точное определение: «Теорема Мартин-Лёфа гласит: пересечение всех множеств эффективной меры 1 не только не пусто, но и само имеет эффективную меру 1. Таким образом, это пересечение является наименьшим множеством эффективной меры 1. Оно называется конструктивным носителем (constructive support) меры. Это множество мы и провозглашаем в качестве класса T всех типических последовательностей».

Примечание: в этом определении речь идет о другой работе П. Мартина-Лёфа: Martin-Lof P. The definition of random sequences / P. Martin-Lof // Inform. Control. — 1966. — Vol. 9, № 6. — P. 602–619.

Типические последовательности воспринимаются здесь как конструкционные материалы для специализированных последовательностей, например, типов. Здесь же вопрос о том могут ли сами эти типические последовательности являться типами — специализированными последовательностями, не поднимается, вероятно, ввиду его очевидности.

Однако, далее имеет место быть известный «закон Мёрфи» — «Anything that can go wrong will go wrong» — типические последовательности сами стали типами, но не только, они также стали алфавитами в Марковском смысле, что стало «проблемой». Причиной этого явилось бурное развитие шифрования, точнее сама цель – скрыть содержание сообщения.

2.3. Как быстро выйти на источник проблемы

Так как имеет место применение математических объектов — типических последовательностей, как специализированных, т.е. против их же природы и фундаментальной математической основы с целью использовать её для затруднения декодирования (дешифрования), то здесь не обойтись «легким» обзором и перечислением исторических этапов развития такой научной и инженерной мысли, а необходимо поискать более простой способ выйти на источник проблемы теорий типов от типов — типических последовательностей.

Предложим другой путь: рассмотрим один из известных типов, понимаемых «примерно» одинаково во всех известных теориях типов. В качестве такового выберем логический тип: «Boolean» и подвергнем его анализу в рамках теоретико-алгоритмического подхода.

Значения: «1/0», «истина»/«ложь», «true»/»false», единица»/«ноль», «24»/«0» этого типа «Boolean» интуитивно восприняться как типичные, иначе — «типические», причем здесь же приведен пример пяти алфавитов, включающих только показанные пары. Причем слова пусть не вводят в заблуждение – это также буквы.

Известны алгоритмы, которые получают на вход последовательность таких значений — буквы этих алфавитов и перерабатываю их.

Поэтому можно обобщить и сформулировать шире: типические последовательности, поступают на вход алгоритма.

Обобщим также наблюдения за ними: некоторые последовательности, интуитивно воспринимаемые как специализированные, оказываются «такими же как» (в ловеровском смысле) и типические, например, «0»/«24» (вольта), только это сложно заметить, опираясь только на интуицию. Здесь типические последовательности, перерабатываемые в *«определенном порядке»*, составляют алгоритм (например, широко применяемый в промышленной автоматике).

Дополнительно рассмотрим и другой случай, когда типические последовательности образуют буквы, а буквы слова, причем таковые могут быть намеренно перемешаны, ровно для того, чтобы не понять их, тогда *«определенный порядок»* отождествления (отнесения к классу) типических последовательностей – буквам, букв – словам и их перемешивания, составляют алгоритм.

Расширим опыт наблюдений, например, перемешенные типические последовательности здесь интуитивно воспринимаются как беззаконные в брауэровом смысле, так как мы о них ничего не можем сказать, но можем их наблюдать, одновременно таковые, также подходят под определение «случайных по Колмогорову» (в смысле, показанном в [6]), но способных утратить случайность (вероятно и беззаконность), в случае обнаружения в них — абсолютно неслучайных объектов [6], в результате творчества, например, подбора или дешифрования фрагмента – переработки творческих последовательностей по Брауэру [7] — «подбора».

Необходимо проверить алгоритм, перерабатывающий слова (например, Маркова), на предмет все ли предписания алгоритма — *«определенный порядок»*, синтезирующего эти слова, перерабатываются, т.е. на сколько алгоритм-переработчик полон относительно алгоритма-синтезатора слов. Также насколько оправдано и обосновано новаторство в математическом определении «типа» слов через их взаимодействия — «типы морфизмов».

Очевидно, что возможно некое новаторство в способе задания *«определенного порядка»*. Однако такое новаторство проводится здесь не будет, так как цель данного исследования не разработать и обосновать новую теорию типов, а выявить «несовершенства» (слабые места) алгоритмов, вытекающие из смежной теории типов, в частности, на примере алгоритмов, перерабатывающих слова, морфизмы и их значения: *N, cN, cNnet*–схем алгоритма Маркова.

2.4. Кому может быть интересно исследование

Исследование предполагается как вклад в развитие положений теории алгоритмов и может быть интересно для специальностей ВАК: «1.1.5. Математическая логика…», «1.2.1. Искусственный интеллект…», «1.2.3. Теоретическая информатика…», «2.3.2. Вычислительные системы…», «2.3.3. Автоматизация и управление…».

В настоящее время теории: множеств, категорий и категорий, алгоритмов, стоят крепко, несмотря на отдельные ограничения известных их формализаций, чего нельзя сказать о теориях типов. Предполагалось, что последние дадут некое обобщение над первыми, возможно заместив их, путем разрешения известных проблем этих теорий, однако оказалось, что это не так.

Тем не менее теория типов вошла в информатику как вполне годное техническое решение для синтеза языков программирования, в современном их понимании, поэтому от опыта, накопленного от её применения в информатике, так просто не отмахнуться.

Заметим, что Колмогоров А.Н. и Успенский В.А. в работе [6] «подсветили» проблему будущих теорий: типов и алгоритмов, через задачу обнаружения абсолютно неслучайных объектов, если о наличии таковых в данных на входе не известно алгоритму, в том числе, то, что это могут быть типы, в частности, типы морфизмов, то: «*у всякого алгоритма возникает постоянная (неотделимая) задача обнаружения этих объектов и дополнительная логика вывода*»: «известны», «не известны», «обнаружены», «не обнаружены», «обнаружены с какой-либо мерой», «обнаружены «такие же как» и т.д.

2.5. Что можно выяснить об алгоритме искусственного интеллекта?

Что есть алгоритм искусственного интеллекта — это открытый вопрос. Насколько известные теоретико-алгоритмические и теоретико-типовые конструкции применимы в нем? Это алгоритм обязан быть полным или нет?

Решение задачи

3.1. Известный тип «Boolean»

В теоретико-типовых подходах известен и нашел широкое практическое применение тип «Boolean». Это тип используется, например, в промышленной автоматизации, причем сигналы этого типа формируют различные устройства (объекты окружающего нас мира), в том числе, объединенные в композиции. Как правило, такие сигналы, передаются по электрическим линиям в виде электрического напряжения 0/24 В или 0/220 В.

Поэтому рассмотрим пример таких систем промышленной автоматики, рисунок 1.

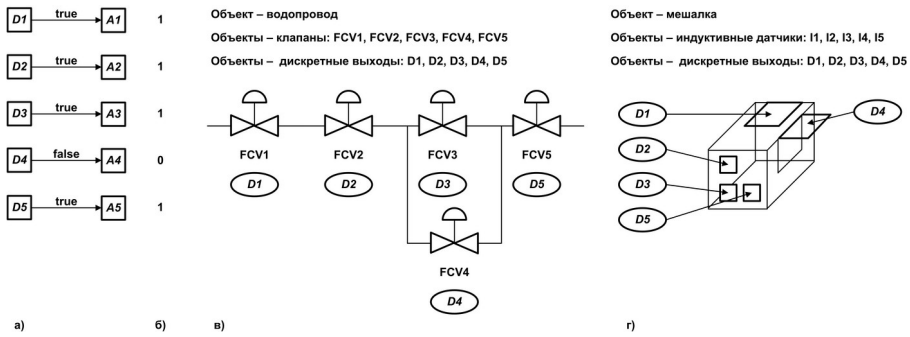

Рисунок 1 - Пример применения логического типа «Boolean»:
а — морфизмы, поступающие на вход некоторого алгоритма; *б* — «вычислимые функции», состоящие из, например, нулей и единиц, или ответов «да» и «нет», или значений морфизмов «false» и «true»; в — пример объекта с композицией морфизмов как на а); г — аналогичный объект, который может иметь ту же композицию морфизмов

Примечание: D1,D2,D3,D4,D5,A1,A2,A3,A4,A5 — слова; «→» – стрелка — обозначение морфизма; true,false — значение морфизма

На рисунке 1а) показаны морфизмы, поступающие на вход некоторого алгоритма.

Возникает вопрос: *«Какому объекту окружающего нас мира может принадлежать композиция слов и морфизмов, показанная на рисунке 1а)?»*

На рисунке 1в) показан пример такого объекта, а на рисунке 1г) показан аналогичный объект, который вполне может иметь ту же композицию морфизмов.

Из приведенных примеров видно, что тип «Boolean» используется для передачи сигналов от некоторого объекта окружающего нас мира, здесь, например, эти сигналы означают ответы на вопросы об этом объекте. Причем неизвестно какие вопросы будут заданы об этом объекте и насколько ответы способны показать о каком объекте идет речь. При этом заданных вопросов может быть вполне достаточно для решения какой-либо практической задачи.

Из ответов не вполне понятно состояние объектов, если рассмотреть композицию ответов, что кодируется как: «да» или «нет», где эти ответы исключает или наоборот определяет взаимное их отношение в композиции.

Человек умеет интерпретировать эти вопросы и ответы, так как ему известно о каком объекте идет речь и именно человек позаботился о том, что эти вопросы и ответы содержательно и адекватно определяют состояние объекта. Хотя на практике это не всегда так, часто упускаются важные вопросы или возможные ответы, тогда человек совершенствует их.

Этот человек может показать, как перерабатывать такие «вычислимые функции», как показаны на рисунке 1б), состоящие из, например, нулей и единиц, или ответов «да» и «нет», или значений морфизмов «false» и «true», алгоритму, синтезировав таковой.

Причем очевидно, что такие алгоритмы не универсальны для всех объектов, они подойдут, например, для множества объектов, показанных на рисунке 1в), но не подойдет для объектов, показанных на рисунке 1г) или наоборот, но может так совпасть, что подойдут для обоих множеств. Эти «вычислимые функции» то же, что и «специализированные последовательности», показанные выше.

Примечание: этот промежуточный вывод известен в науке и получен давно А.Н. Колмогоровым в работе [6, С. 429]: «При этом — для фиксированной последовательности — возможно много таких алгоритмов, и каждый из этих алгоритмов, в свою очередь, может быть осуществлен различными программами, представляющими собою тексты».

Алгоритмы, которые при этом удастся синтезировать, вполне соответствуют известным «классическим» формализациям А. Тьюринга и А.А. Маркова, введенным именно для «вычислимых функций», где таковые, как здесь, одновременно являются «специализированными последовательностями».

Здесь логический тип «Boolean» корректно передает тип именно морфизма, между двумя словами, хотя часто не очевидно, что «истина» или «ложь» не могут быть «просто подвешены в воздухе», но при этом знание об этом типе не дает алгоритму никакого знания об объекте, формирующим морфизмы этого типа.

Вызывает интерес, что «абсолютно неслучайные объекты» (или единственный такой объект) в «специализированной последовательности», алгоритм также обнаружить не может.

Тип «Boolean» безусловно важен как теоретико-типовая конструкция, но нужен ли таковой алгоритму, например, ИИ, в таком виде, точнее математически — таком способе определения? «Сколько в теории типов таких типов, которые теоретико-алгоритмически определены не вполне корректно или удачно?»

Здесь нельзя говорить, что тип «Boolean» математически неполон, но можно говорить, что способ его определения вероятно не завершен, поэтому возможно неполон.

В *N, cN, cNnet*–схемах алгоритма Маркова имеется механизм порождения морфизма такого типа, кодирования этого морфизма в алфавите *M*, переработки, как самого значения морфизма, так и слов, для которых таковой определен, но фактически алгоритм выполняется не полностью, для полного его выполнения необходимо каким-либо образом сообщить алгоритму как перерабатывать всё множество морфизмов типа «Boolean», что эквивалентно введению в его состав *дополнительного* алгоритма.

На практике так и происходит, в рассмотрение вводится *дополнительный* алгоритм, перерабатывающий «вычислимые функции» в смысле Тьюринга («специализированные последовательности», вероятно при этом и «абсолютно неслучайные объекты») и для каждой такой функции, синтезируется практически (почти) «уникальный» (*дополнительный*) алгоритм.

Таким образом, приложение теоретико-типовых положений к теоретико-алгоритмическим нельзя полагать завершенными, так строго не определен этот *дополнительный* алгоритм. Здесь же возникает теоретико-алгоритмическая неопределенность самого объекта, формирующего сигналы логического типа, которую нельзя никак преодолеть без внешнего вмешательства творца этого алгоритма и его помощника, который поможет творцу присоединить источники, так как это хотел творец.

Кроме того, для двух электрических сигналов «0» и «24» В, где В тип — «Вольт», алгоритму совершенно непонятно, что есть «0», действительно значение или пустота. П. Мартин-Лёф в своей интуиционистской теории типов, конечно, учитывал, для типа «Boolean» пустоту, однако не понятно, почему не пошел дальше, например, рассмотрев другие способы формализации этого типа. Например, в промышленной автоматике с 60-х годов 20-века известна токовая петля (current loop), которая вполне подходит как контр-модель типа «Boolean» и содержит уровни, которые можно понимать, как: пусто, ложь, истина, причем между ложью и истиной имеется шкала, которая не обязательно линейна.

Интересно, если приводить значения на этой шкале к общему числу значений, не удастся ли получить аналог формулы Байеса, преодолевающей парадокс Гемпеля? Может это и есть более адекватная модель типической последовательности и типа «Boolean»?

3.2. Что такое гетерогенные композиции типов?

Для ответа на вопрос: «Сколько в теории типов таких типов, которые теоретико-алгоритмически определены не вполне корректно или удачно?», потребуется рассмотреть несколько примеров.

Говоря о типах морфизмов в работе [4] понимались все возможные такие типы из известных в науке и практике. Такой подход был выбран по прагматичным соображениям: «принять во внимание все известные типы, не формализуя их, в том виде, в котором таковые стали общепризнаны, общезначимы и на данный момент формализованы».

Этот подход не плохо работает в конструктивных рассуждениях — «прими то, что осмыслено и конструктивно реализовано на сегодняшний день, а потом будешь делать выводы».

Если рассмотреть такие типы внимательно, то можно заметить, что, например, известные типы: *.doc, *.pdf, char, int, real, struct, ампер, вольт, килограмм, метр, по существу, должны, с теоретической точки зрения, быть беззаконны в смысле Л.Э.Я. Брауэра, но на практике это не совсем так, в частности, типы *.doc, *.pdf, struct – небеззаконны в силу наличия внутренней структуры.

Тогда, показанный в [4] способ их переработки для *N*, *cN*, *cNnet*с-хемах алгоритма Маркова, может быть применим к множеству из них, но не ко всем, но не по причине, показанной выше — незнания объекта, их формирующего, а по другим причинам, покажем это примером.

Обозначим такие типы $type_K$, объекты как *X,Y,Z,W*, морфизмы к которым приложены эти типы морфизмов традиционно — стрелками.

Приведем пример композиции объектов и их типов морфизмов:

$$X \xrightarrow{\text{type}_\text{K}=\text{"Ампер"}} Y \xrightarrow{\text{type}_\text{K}=\text{"*doc"}} Z \xrightarrow{\text{type}_\text{K}=\text{"print}f\text{"}} W \qquad (1)$$

Рассмотрим эти типы отдельно без объектов:

$$\text{type}_\text{K} = \text{"Ампер"} \rightarrow \text{type}_\text{K} = \text{"*doc"} \rightarrow \text{type}_\text{K} = \text{"print}f\text{"} \qquad (2)$$

Проанализируем выражения (1), (2), заметив, что если принимать во внимание объекты *X,Y,Z,W* из формулы (1) то, можно понять, что такое преобразование можно реализовать, хотя сразу и не скажешь как, иначе дело обстоит с формулой (2), где таких объектов нет, а сама формула математически бессмысленна, или как, назвал это математическое «явление» Л.Э.Я. Брауэр — беззаконна. При этом типы морфизмов: *"*doc"* и *"printf"* сами по себе небеззаконны, так как содержат структуру, предписания по дешифрованию и управление переработкой.

Примечания:

1) пример для *X,Y,Z,W* (последовательно): измеритель тока, программный регистратор на базе компьютера, принтер, бумага;

2) слово «явление» здесь употреблено для того, чтобы не использовать слово «свойство», которое может окрасить фразу в теоретико-множественный смысл, что делать не хотелось бы;

3) тип, например, — это строго говоря имя конкретизированной, известной беззаконной последовательности, сама последовательность, конечно, состоит из значений морфизмов, хотя в теоретико-типовом смысле — это известный тип.

3.3. Интуитивное представление о композиции типов в алгоритме ИИ

Введем теоретическое допущение, а именно, известную в конструктивной математике абстракцию потенциальной осуществимости для композиции типов, которая интуитивно «кажется» пригодной для алгоритма ИИ. Синтезируем пример для этой абстракции, положив потенциальную осуществимость высказыванием. В качестве такого высказывания выберем слово из словаря русского языка, построенного из букв русского алфавита — это слово *«Могу»*.

Для того чтобы завершить потенциальную осуществимость, необходимо недвусмысленно указать, что именно потенциально осуществимо для такого высказывания, как *«Могу»* — это, например, высказывание, в том же словаре и алфавите — *«Синтезировать»*, продолжая его словом — *«Печать»*. Зачем это делать? Выбранная конкретизированная форма потенциальной осуществимости — это эксперимент, т.е. так сделать разрешено, сделаем и посмотрим на полученный результат.

Используем приведенные формулы (1) и (2) для конструирования такого примера потенциальной осуществимости:

$$X \xrightarrow{\text{type}_\text{K}=\text{"Могу"}} Y \xrightarrow{\text{type}_\text{K}=\text{"Синтезировать"}} Z \xrightarrow{\text{type}_\text{K}=\text{"Печать"}} W \quad (3)$$

Рассмотрим эти типы отдельно без объектов:

$$\text{type}_\text{K} = \text{"Могу"} \rightarrow \ \text{type}_\text{K} = \text{"Синтезировать"} \rightarrow \text{type}_\text{K} = \text{"Печать"} \quad (4)$$

Такое представление последовательности типов морфизмов, во-первых, не является бессмысленным, а является вполне осмысленной фразой на естественном языке (логическим высказыванием), состоит из аристотелевых категорий, является результатом сжатия вполне конкретизированной и неизвестной, до её обнаружения и классификации (распознавания) категории в аристотелеву, состоящей из объектов окружающего нас мира, взаимодействующих друг с другом, во-вторых, такое представление типов морфизмов можно приложить к алгоритму ИИ, как минимум, имитирующему человеческие действия, в-третьих, требует ввода в рассмотрение некоторого отличного, от показанных выше, *способа представления* типов.

Аналогично типам *"*doc"* и *"printf"*, из примера выше, типы *"Могу"*, *"Синтезировать"*, *"Печать"* сами по себе также, строго по Брауэру, небеззаконны, содержат структуру — являются категорией в теоретико-категорном смысле, аристотелевой категорией, логическим высказыванием, а также вероятно содержат предписания по дешифрованию и управление переработкой, сжатые при перекодировании некоторой категории, образованной в результате переработки данных источников (пускатель, реле, процессор принтера и т.д), в аристотелеву.

3.4. Переход к интуиционистскому представлению о композиции типов в алгоритме ИИ

Заметим, что если бы (4) делал человек, то логично представить, что у него есть исправные: компьютер и принтер, в котором заправлен картридж и вставлена бумага. Человек обозревает мир вокруг себя, а его интеллект делает нужные выводы. Если человек ошибается на счет заправки картриджа, то все равно имеет возможность проверить этот факт, распечатав один лист и оценив полученное качество печати. Иначе дело обстоит с алгоритмом ИИ, его тип морфизма *«могу»* или, напротив, *«не могу»* являются результатом переработки множества данных, что можно понять как изначальное состояние типа этого морфизма как *«λ-пусто»*, т.е. такой тип морфизма может появиться в зависимости от ряда логических условий. Это отличает интеллект человека от искусственного.

Состояние такого типа морфизма как *«λ-пусто»* также свойственно человеку, но имеет несколько иную форму чем у ИИ, а именно, пока он не попробует: прыгнуть, плыть, бежать и т.д., причем меняясь со временем: «могу прыгнуть дальше, чем тогда, когда мне было 6 лет», до: «не могу прыгнуть, потому, что стал пожилым человеком и надо поберечься» или «имеется препятствие».

Тип морфизма такой как *«Печать»* имеет уже известную формализацию в информатике, а вот тип морфизма *«Синтезировать»* указывает на то, что, например, алгоритм ИИ, содержит в своем составе некоторый дополнительный алгоритм (знание о потенциальной осуществимости), который получив на вход вывод *«Могу»* (указание об осуществимости), вызывает предписание этого алгоритма о конструировании (творчестве, которое не обязательно завершится успехом), иначе правилах построения или, в термине который применялся ранее: выводе.

Заметим:

1. Некоторые известные в теории типов типы являются творческими последовательностями по Брауэру для некоторых алгоритмов, так как их успешность необходимо проверить.

2. Тогда в теории типов должны быть типы, которые беззаконны по Брауэру, т.е. абстракции подменяющие такие типы как «вольт» и наделенные его полным смыслом и свойствами.

3. *Вывод* проработан, например, П. Мартином-Лёфём в своей теории, возможное брауэрово творчество при выводе — нет и это по-прежнему открытый вопрос всех теорий типов.

Вероятно, эти замечания мог бы учесть и сам П. Мартин-Лёф в своей интуиционистской теории типов, обратившись к рассуждениям Брауэра, но тогда следовало бы признать, что некоторые типы сложно формализовать на определённом этапе развития теоретического знания о типах, оставив дискуссию открытой. Недосказанность сохранилась и сегодня, другие известные теории типов никак не учитывают рассуждения Брауэра, что наводит на мысль, что и эти теории далеки от совершенства.

3.5. Обнаружены слабые в категорной смысле типы?

Однако, повторно обнаружился дополнительный алгоритм. Обратим внимание на сами композиции (1), (2) и (3), (4), таковые иллюстрируют неоднородные (гетерогенные) типы.

Здесь композиции неоднородных (гетерогенных) типов ассоциативны или нет? Очевидно, что нет.

Тогда возникает потребность как-либо определить математически такие неассоциативные композиции. Для этого можно воспользоваться теориями категорий и -категорий, в которых известны такие композиции, а объекты и стрелки образуют категории, причем если таковые неассоциативны, то слабые.

Если типы способны образовывать слабые категории, то некоторые выводы теории типов и известные формализации следует взять под сомнение или пересмотреть.

Говоря проще, алгоритм может переработать слабые категории, но при этом целесообразно принимать во внимание возможный колмогоровский хаос [6], и ввести в рассмотрение возможность потенциальной осуществимости не так как ожидалось, а, например, с ошибками (брауэровым творчеством), возможно, критическими, потенциально, приводящими алгоритм к краху, а сам объект к уничтожению.

Если при этом, приложить полученные промежуточные выводы о слабости типов в категорном смысле к алгоритму ИИ, то можно понять, что это не совсем то, что планировалось для такого алгоритма, желательно, что таковой оперировал только строгими категориями, без хаоса. *Но как достичь такого результата?* Например, для алгоритма ИИ строить высказывания только на «логически защищенном» искусственном языке. Иначе говоря,

необходим искусственный язык, в котором определена логика, которая способна защитить алгоритм, как минимум, от краха. *Интересно, что это за логика?*

Тезис Бишопа против теории типов

4.1. Зачем нужно еще одно исследование о типах?

Со времен А.А. Маркова и П. Мартина-Лёфа, несмотря на бурное развитие информатики, типы так и не измелись — это все те же черточки Маркова [5] и черточки, конечные деревья, целочисленные матрицы, релейные схемы Мартина-Лефа [7], проще говоря алфавиты символов. К этим алфавитам добавляют словари слов, призванных заместить недостаточную выразительность только букв алфавитов — «специализированные» последовательности (в смысле [6]).

Типические последовательности, хорошо показанные в работе [6], толи «черточками», толи «единицами» (см. оригинал статьи), и это наводит на мысль, что есть не только «ноль» или «единица», о которых шла речь в этой работе, но и буквы алфавита, так как, интуитивно кажется, что они типичны (последний тезис рассмотрим далее более подробно).

Например, в гомотопической теории типов (ГТТ) В.А. Воеводского, как показано в работе [2] — «В рамках гомотопической семантики типы интерпретируются как абстрактные пространства (для которых определены основные понятия теории гомотопий), а термы — как точки таких пространств. ГТТ позволила установить, что «не все типы одинаковы»: только типы специального вида (гомотопического уровня), а именно типы, содержащие самое большее единственный терм, можно отождествить с высказываниями; типы другого специального вида отождествляются с множествами; типы более высоких уровней мы оставим в стороне. Таким образом, первоначальная идея о том, что всякий тип допускает интерпретации как высказывание, множество и т.д., в контексте ГТТ не выглядит убедительной. Однако по всякому данному типу высшего порядка можно каноническим образом построить соответствующее высказывание, искусственно отождествляя все термы данного высшего типа. Такую процедуру в ГТТ называют обрезанием (truncation)».

Читай словосочетание «построить соответствующее высказывание», так: все те же «черты и резы» сохранились, но в пространстве, где это можно осуществить «каноническим образом» (последнее словосочетание крайне неконструктивно и поэтому вызывает подозрения).

Любой современный стандарт, протокол, спецификация, в смысле, принятом в информатике, является таким алфавитом и/или словарем, теми же «чертами и резами». Вероятно, идея так кодировать типы не оставит человечество.

4.2. Почему не принять идею В.А. Воеводского в теории алгоритмов, ведь она, говорят, показала себя уверенно в математических доказательствах?

Потому, что принимается во внимание тезис Э. Бишопа: «Do not ask whether a statement is true until you know what it means. (Не спрашивай, истинно ли утверждение, пока не знаешь, что оно означает)» [8, С. 6], который нашел свое подтверждение при исследовании *N*, *cN*, *cNnet*–схем алгоритма Маркова [4] и воплотился в специальный способ подсчета индукции, специфика, которого состоит в том, что таковой «подражает» способу преодоления парадокса Гемпеля, синтезированному Байесом.

Проще говоря, только дедуктивный алгоритм (способ), не учитывающий индукцию, также вызывает сильные подозрения в «незавершенности», читай в ограниченной применимости, с позиций конструктивной математики.

Имеются серьезные основания полагать, что идея Воеводского не сможет реализовать, приведенный тезис Бишопа, так как имеется объект-исключение из ГТТ — это тип — индукция, которая есть возрастающая или убывающая мера истинности или ложности высказывания, а не высказывание, которое вполне определенно представимо типом для алгоритма. Можно попробовать обрезать (truncation) индукцию, искусственно её отождествляя с высказыванием, но все такие отождествления *являются логически ложными*!

Альтернативный взгляд на формализацию типов можно получить из подходов: Л.Э.Я. Брауэра — определить беззаконные последовательности [9, С. 124], если для их разделения использовать имена (названия, проще говоря, слова, представимые в алфавите Маркова), то эти имена и есть типы; и Э. Бишопа — считать всё алгоритмами [9, С. 128], [10], понимай и типы тоже.

Тогда напрашивается промежуточный вывод: подходы Брауэра и Бишопа позволяют понять, что происходит с современным представлением о типах, они стали не только алфавитами и словарями, но и частично включили в себя идеи этих математиков, например, использовать для кодирования и декодирования типа алгоритм.

Это надо показать более отчетливо, так как такой промежуточный вывод не лежит на поверхности.

4.3. Тип «индукция» где ты?

Самый беглый взгляд на известные формализации алгоритма А. Тьюринга, А. Чёрча, А.А. Маркова показывает, что таковые не имеют никакой защиты от парадокса Гемпеля. Причем, Марков это отчетливо понимал, вводя свой «тезис» и «марковскую логику» в теорию алгоритмов (работа [5]). Вероятно, А.А. Марков адекватно оценивал, что в его распоряжении полностью отсутствует чисто логический механизм противодействия этому парадоксу, поэтому ввел собственную альтернативу двойному отрицанию. Вводить в рассмотрение индукцию, дополняя нормальный алгоритм механизмом, похожим на теорему Байеса об индукции, преодолевающей парадокс Гемепеля, было, видимо, не целесообразно по субъективным причинам.

Альтернативный подход к «истинности» и «ложности», например, известный в естественном языке индейцев Южной Америки — аймара, или, например, туканский язык народа туюка, в которых существует дополнительная характеристика глагола — категория эвиденциальности: визуальное и невизуальное свидетельство, вещественное доказательство, пересказывательность, умозаключение, реализуемые изменением слова через суффикс, читай образование внутренней категории в слове, а не соединение отдельных слов; требовал бы пересмотра способов формирования аристотелевых категорий и высказываний, «подвешивая» проблему двойного отрицания и, как следствие, парадокса Гемпеля, через ввод понятий: «логически неопределимо» и «логически неуверенно, при любом

значении индукции», но этот альтернативный подход так и не сложился, его по-прежнему предпочитают не замечать. Видимо науке нужен «новый» Брауэр для «продвижения» новаторства в логике. Поэтому нам никогда не узнать, как бы выглядели, известные формализации алгоритма с таким механизмом, например, нормальный алгоритм Маркова, способный преодолевать парадокс Гемпеля.

В целом этот парадокс есть достаточное основание для того, чтобы не считать известные формализации А. Тьюринга, А. Чёрча, А.А. Маркова завершенными формализациями алгоритма, даже в условиях, наложенных и признанных частью научного сообщества, ограничений на них, например, главного — *вычислимых функций*. Вопрос синтеза формализаций этих алгоритмов открыт и сегодня, хотя бы на этом простом основании.

Как несложно заметить, «классическая» научная традиция опрометчиво «простила» парадокс Гемпеля этим известным формализациям алгоритма. Тем не менее это не отменяет фактического положения дел, которое говорит о том, что синтез формализации алгоритма необходимо продолжить. Аналогичным образом дело обстоит и с теориями типов. Формализация типов также является открытым научным вопросом, поэтому же основанию — учету индукции. Видимо Бишоп это отчетливо понимал и фактически его тезис намекает, что есть случаи, когда в распоряжении алгоритма есть объективно только индукция и ничего более. Причем интересно то, что в некоторых случаях у этого явления имеет место быть временная обусловленность Соломона Крипке.

Выдвинем гипотезу о том, что одной из причин, по которой индукция не стала частью формализации алгоритма, иначе, примером-основанием, являются типические последовательности, которые имеют место быть, при представлении данных на естественном и искусственном языках, формально в теории типов, реально также и в теории алгоритмов.

Эти последовательности демонстрируют, что тезис Бишопа, есть интуитивная догадка о потенциальной проблеме приложения логики при переработке данных, вне зависимости от того, какие это данные, возникшей при рассмотрении двоичных последовательностей, которые использовались для представления объектов в его рассуждениях. Поэтому введение, например, кроме «нуля» и «единицы» некоторой третьей типической последовательности ситуацию и содержание тезиса Бишопа не изменит. Тогда можно пойти дальше в рассуждениях.

Что будет, если расширить алфавит типических последовательностей, например, до размера алфавита естественного языка? Проблема распространиться на весь алфавит?

Проблема распространится на весь алфавит, так как сам алфавит не несет правил образования из типических последовательностей — специальных, а эти правила, по Бишопу и де-факто — алгоритмы, в естественных языках, называемые грамматики (будем пользоваться именно этим термином исключительно для упрощения дальнейших рассуждений). Именно при составлении таких грамматик появляется логика, т.е. алгоритм неизвестен значит нет логики, есть только индукция, алгоритм известен есть и логика, и индукция. *Грамматика и есть дополнительная часть алгоритма, делающая его полным?* Если расширить представление о том, что есть грамматика, то — да.

4.4. Конструктивно неполный алгоритм

Пусть есть алгоритм, перерабатывающий слова и их морфизмы со значениями. Пусть значения морфизмов ограничены только типом «Boolean». Такой алгоритм фактически перерабатывает типические последовательности, хотя во входных данных есть и слова. Алгоритм, перерабатывающий значения морфизмов — типические последовательности, в качестве входных данных — это неполный алгоритм.

Для того чтобы такой алгоритм стал полным, ему необходимо сообщить грамматику, позволяющую раскодировать: слова их морфизмы со значениями, в последовательности, определяемой этой грамматикой. Эту грамматику, например, может знать только тот, кто разработал данный алгоритм и не сообщать её никому, тогда для любого наблюдателя таковая останется неизвестной.

Такая грамматика может быть уникальна. Иногда таковая может совпадать с грамматикой какого-либо другого объекта, тогда, дополнительно, возникает задача различения этих объектов и их грамматик. Неполнота, обусловленная входными данными — типическими последовательностями, свойственна всем алгоритмам, вне зависимости от способа их формализации.

Источниками типических последовательностей, помимо выборок входных данных, могут являться спецификации типов (стандарты типов), что обусловлено теоретико-типовыми положениями, иначе говоря, принятой в них формализацией. По-видимому, такая теоретико-типовая формализация является спорной (оспариваемой, дискуссионной). Однако пересмотра таковой не наблюдается.

Веской причиной пересмотра этой формализации может являться задача синтеза алгоритма ИИ. Для этого алгоритма не только наличие типических последовательностей в типах вызывает серьезные вопросы, но и в самих данных. Предлагается для алгоритма ИИ исключить саму возможность оперировать только типическими последовательностями.

Заключение

Конкретизированные специальные последовательности, полученные из типических последовательностей с грамматиками, были приняты научным сообществом, как *вычислимые функции* по Тьюрингу и Чёрчу.

Тогда закономерны вопросы:

«Как называются неконкретизированные, потому, что ещё не были классифицированы или просто неизвестные специальные последовательности? В этом же контексте, как называются типические последовательности? Невычислимые или невычисленные функции? Как понимать слово функции как структуру: объект-морфизм? При этом морфизм не определен? Алгоритмы А. Тьюринга, А. Чёрча, А.А. Маркова полны по Тьюрингу, но неполны с позиций конструктивной математики и это одно из их фундаментальных свойств?

Наличие задачи поиска дополнения к неполному алгоритму, способного сделать таковой полным, а также то, что как Тьюринг, так и Чёрч с Марковым хорошо о ней были осведомлены, и открыли научную дискуссию о проблеме,

получившей название: «равенства классов P и NP» (проблема перебора)», говорит о том, что с «полнотой» и «полнотой по Тьюрингу» в теории вычислимости следует быть внимательным.

Для того чтобы уточнить формализации неполных алгоритмов, необходимо дополнительно исследовать:

1. Способ порождения полного алгоритма.
2. Способы добавления к неполному алгоритму, алгоритмов, перерабатывающих грамматики типических последовательностей.
3. Способ замещения неизвестного дополнительного алгоритма, перерабатывающего грамматики типических последовательностей.
4. Способ модернизации как неполного, так и алгоритма, перерабатывающего грамматику типической последовательности.
5. Подход к обособлению алгоритма, которому не требуется перерабатывать типические последовательности — выделение его формализации отдельно от неполных алгоритмов.

Вызывает интерес пример *априори* полного алгоритма.

Этот пример: *N, cN, cNnet*–схемы алгоритма Маркова, перерабатывающие слова и морфизмы нетипические последовательности.

Полнота этого алгоритма находится в прямой зависимости от входных данных, если таковые есть слова естественного языка, читай типические последовательности, к которым грамматики естественного языка уже приложены, а значения морфизмов этих слов не типические последовательности, то алгоритм полон, если входные данные буквы, читай — типические последовательности, к которым не приложены грамматики, или значения морфизмов — типические последовательности, то алгоритм неполон.

Этот теоретический вывод шире, его также можно распространить на проблему «равенства классов P и NP» (проблему перебора)», т.е. — это условии равенства этих классов.

Выводы

Из представленного вывода о «равенстве классов P и NP» можно синтезировать условия *априорного* неравенства таковых, через определение классов как алгоритмов, а различных входных данных, в том числе типов, через типические и специальные последовательности.

Для развития формализации алгоритма, в том числе при поиске подходящей для алгоритма ИИ, необходимо соответствующее теоретико-типовое обобщение, в противном случае алгоритм ИИ может «унаследовать» всё туже неполноту алгоритма, что фактически будет означать его несостоятельность (крах).

При синтезе алгоритма ИИ, на первом этапе его формализации, целесообразно ввести ограничения на входные данные: таковые должны быть словами и морфизмами со значениями нетипическими последовательностями.

Главным научным выводом данной работы можно считать целесообразность поставить под сомнение интуитивное понятие «типичность» или «нетипичность» символов лежащую в основе теорий типов и предложить другой подход. Здесь этот альтернативный подход предложен в «мягкой» форме – использовать для типов беззаконные и творческие последовательности Брауэра в выделенных алфавитах Маркова. Как перерабатывать такие типы алгоритмом?

Конфликт интересов

Не указан.

Рецензия

Все статьи проходят рецензирование. Но рецензент или автор статьи предпочли не публиковать рецензию к этой статье в открытом доступе. Рецензия может быть предоставлена компетентным органам по запросу.

Conflict of Interest

None declared.

Review

All articles are peer-reviewed. But the reviewer or the author of the article chose not to publish a review of this article in the public domain. The review can be provided to the competent authorities upon request.

Список литературы / References

1. Martin-Lof P. Intuitionistic Type Theory / P. Martin-Lof. — Napoli : Bibliopolis, 1984. — 91 p.
2. Ковалев С.П. Проблема обоснования в формальном представлении знаний / С.П. Ковалёв, А.В. Родин // Вестник Томского государственного университета. Философия. Социология. Политология. — 2018. — № 46. — С. 22–29. DOI: 10.17223/1998863X/46/3.
3. Колмогоров А.Н. К определению алгоритма / А.Н. Колмогоров, В.А. Успенский // Успехи математических наук. — 1958. — Т. 13, № 4(82). — С. 3–28.
4. Коновалов В.А. с*Λ*net-схема алгоритма Маркова / В.А. Коновалов // Вестник компьютерных и информационных технологий. — 2024. — Т. 21, № 12. — С. 45–52. DOI: 10.14489/vkit.2024.12.pp.045-052.
5. Марков А.А. Теория алгорифмов / А.А. Марков, Н.М. Нагорный. — 2-е изд., испр. и доп. — М. : Фазис, 1996. — 493 с.
6. Колмогоров А.Н. Алгоритмы и случайность / А.Н. Колмогоров, В.А. Успенский // Теория вероятностей и ее применения. — 1987. — Т. 32, № 3. — С. 425–455. DOI: https://doi.org/10.1137/1132060.
7. Мартин-Леф П. Очерки по конструктивной математике / П. Мартин-Леф; пер. с англ. Г.Е. Минца; под ред. А.Г. Драгалина. — Москва : Мир, 1975. — 135 с.
8. Bishop E. Schizophrenia in Contemporary Mathematics / E. Bishop // Contemporary Mathematics. — 1985. — Vol. 39. — P. 1–32.
9. Непейвода Н.Н. Конструктивная математика: обзор достижений, недостатков и уроков. Часть III / Н.Н. Непейвода // Логические исследования. — 2014. — Т. 20. — С. 110–148.

10. Новая философская энциклопедия : в 4 т. / Ин-т философии РАН; Нац. обществ.-науч. фонд; предс. науч.-ред. совета В.С. Степин. — 2-е изд., испр. и доп. — Москва : Мысль, 2010. — 4 т.

Список литературы на английском языке / References in English

1. Martin-Lof P. Intuitionistic Type Theory / P. Martin-Lof. — Napoli : Bibliopolis, 1984. — 91 p.

2. Kovalev S.P. Problema obosnovaniya v formalnom predstavlenii znaniy [The problem of justification in formal knowledge representation] / S.P. Kovalev, A.V. Rodin // Vestnik Tomskogo gosudarstvennogo universiteta. Filosofiya. Sotsiologiya. Politologiya [Bulletin of the Tomsk State University Journal of Philosophy, Sociology and Political Science]. — 2018. — № 46. — P. 22–29. DOI: 10.17223/1998863X/46/3. [in Russian]

3. Kolmogorov A.N. K opredeleniyu algoritma [On the definition of algorithm] / A.N. Kolmogorov, V.A. Uspensky // Uspekhi matematicheskikh nauk [Advances in Mathematical Sciences]. — 1958. — Vol. 13, № 4(82). — P. 3–28. [in Russian]

4. Konovalov V.A. s-Net-skhema algoritma Markova [s-Net-scheme of Markov's algorithm] / V.A. Konovalov // Vestnik kompyuternykh i informatsionnykh tekhnologiy [Bulletin of Computer and Information Technologies]. — 2024. — Vol. 21, № 12. — P. 45–52. DOI: 10.14489/vkit.2024.12.pp.045-052. [in Russian]

5. Markov A.A. Teoriya algoritmov [Theory of algorithms] / A.A. Markov, N.M. Nagorny. — 2nd ed., rev. and enl. — Moscow : Fazis, 1996. — 493 p. [in Russian]

6. Kolmogorov A.N. Algoritmy i sluchaynost' [Algorithms and randomness] / A.N. Kolmogorov, V.A. Uspensky // Teoriya veroyatnostey i ee primeneniya [Theory of Probability and Its Applications]. — 1987. — Vol. 32, № 3. — P. 425–455. DOI: https://doi.org/10.1137/1132060. [in Russian]

7. Martin-Löf P. Ocherki po konstruktivnoy matematike [Essays on constructive mathematics] / P. Martin-Löf; transl. from English by G.E. Mints; ed. by A.G. Dragalin. — Moscow : Mir, 1975. — 135 p. [in Russian]

8. Bishop E. Schizophrenia in Contemporary Mathematics / E. Bishop // Contemporary Mathematics. — 1985. — Vol. 39. — P. 1–32.

9. Nepeyvoda N.N. Konstruktivnaya matematika: obzor dostizheniy, nedostatkov i urokov. Chast' III [Constructive mathematics: review of achievements, shortcomings and lessons. Part III] / N.N. Nepeyvoda // Logicheskie issledovaniya [Logical Investigations]. — 2014. — Vol. 20. — P. 110–148. [in Russian]

10. Novaya filosofskaya entsiklopediya : v 4 t. [New Philosophical Encyclopedia : in 4 vols.] / Institute of Philosophy RAS; National Social Science Foundation; chairman of scientific-editorial council V.S. Stepin. — 2nd ed., rev. and enl. — Moscow : Mysl', 2010. — 4 vols. [in Russian]