ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ И ИХ ЭЛЕМЕНТЫ/COMPUTING SYSTEMS AND THEIR ELEMENTS

DOI: https://doi.org/10.60797/itech.2026.9.1

НЕПОЛНЫЙ АЛГОРИТМ В КОНСТРУКТИВНОЙ МАТЕМАТИКЕ (ЧАСТЬ 2)

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

Представлен результат исследования способа порождения полного алгоритма. Продолжено исследование проблемы неполноты алгоритмов. Во второй части дан ответ на вопрос, если существует некоторый конкретизированный неполный алгоритм, то как выглядит его полный алгоритм, т.е. такой, что содержит недостающую часть. При постановке задачи исследования рассмотрены специальные морфизмы — это известные в науке операции: обращения и присоединения, введенные А.А. Марковым в теории алгоритмов, аппликации А. Чёрча в исчислении и обрезании (truncation) В.А. Воеводского, введенные им в гомотопической теории типов. Предлагается понимать теоретико-алгоритмические операции как теоретико-категорные морфизмы. Выяснена причина ввода операций в теорию алгоритмов. Показано, что известные операции являются следствием недостаточной проработанности формализации алгоритма и призваны нивелировать частные проблемы отдельных из них. Рассмотрены алфавиты букв и слова словарей естественных и синтезированных на их базе языков, как беззаконные и творческие последовательности Л.Э.Я. Брауэра. Это позволило приложить выводы, полученные Брауэром, к формализациям: алгоритма, перерабатывающего слова, а также типам как теоретико-типовым конструкциям, и ввести в рассмотрение «вычислитель А.Ш. Трулстра» для таких последовательностей, как потенциально реализуемого, для полного алгоритма. Получена категорная формула полного алгоритма. Сделан вывод, что при синтезе алгоритма искусственного интеллекта можно поставить задачу получить полный такой алгоритм, что возможно реализовать путем исключения типических последовательностей и использования только специальных последовательностей, а также задействования таких способов их формирования, что их декодирование (дешифрование) не будет аналогичным типическим.

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

INCOMPLETE ALGORITHM IN CONSTRUCTIVE MATHEMATICS (PART 2)

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 are presented and a method for generating a complete algorithm is synthesized. This work is carried out as a continuation of the study of the problem of incompleteness of algorithms, begun in the first part of the study. The second part answers the question of whether there is a certain specified incomplete algorithm, then what does its complete algorithm look like, containing exactly such a missing part, different from other possible incomplete parts. In setting the research problem, special morphisms are considered — these are the operations known in science: inversion and adjunction, introduced by A.A. Markov in the theory of algorithms, A. Church's applications in the λ-calculus and V.A. Voevodsky's truncations, introduced by him in the homotopy theory of types. In particular, such operations can be understood not simply as abstract morphisms, but quite concretized in the category-theoretical sense, as categorical morphisms. The reason for introducing the operations of inversion and adjunction by Markov in his theory of algorithms is clarified. It is shown that the listed, well-known operations of Markov, Church and Voevodsky, are a consequence of insufficient development of the formalization of the algorithm and the homotopy theory of types and are called to level out individual private problems of these formalizations. As a counter-approach, the study considers methods for determining sequences in intuitionism. The type-theoretic reasoning about typical and specialized sequences is additionally supplemented by the conclusions about lawless and creative sequences obtained by L.E.Ya. Brouwer. The typical sequences of alphabets of letters and words of dictionaries of natural and synthesized languages based on them are considered, as lawless and creative sequences of Brouwer. This allowed to consider the very concept of "typicality" in relation to letters and some words in the context of their possible transformations, for example, in Markov's dictionaries. This approach allowed to apply the conclusions obtained by Brouwer more broadly to formalizations: the algorithm processing words, and types as theoretical-type constructions, and also to introduce into consideration "the calculator of A.Sh. Trulstra" for such sequences as potentially realizable, for the complete algorithm. A categorical formula of the complete algorithm has been obtained. It has been concluded that when synthesizing an AI algorithm, one can set the task of obtaining a complete algorithm that can be implemented by excluding typical sequences and using only special sequences, as well as using such methods of their formation that their decoding (decryption) will not be similar to typical ones.

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

Введение

Представлен способ порождения полного алгоритма.

Выделены и подвергнуты критическому анализу известные в науке операции: обращения и присоединения, введенные А.А. Марковым в теории алгоритмов [1], аппликации А. Чёрча в λ-исчислении и обрезания (truncation) В.А. Воеводского [2], введенные им в гомотопической теории типов (ГТТ). Показано, что «операция» — это морфизм, иначе стрелка специального вида, или обособленный морфизм в терминологии, введенной в работах по выводу, который нельзя считать нормальным по Маркову в алгоритме [3], [4], [5].

Навык (умение) алгоритма обнаруживать и классифицировать такую стрелку специального вида равносилен разделению (классификации) множеств по этому разделяющему признаку. Такие разделенные множества есть топосы [6]. Таким образом, далее рассматривается, как теоретико-алгоритмическая, так и теоретико-типовая проблемы более строгого математического определения этих операций. Для того чтобы показать, что эти проблемы не являются «надуманными», рассматривается способ порождения полного алгоритма.

Это не способ переработки или способ вычисления (построения, обнаружения, конструирования) неизвестной, дополнительной части какого-либо алгоритма, а способ синтеза полного алгоритма, в котором строго показана эта дополнительная часть, которая впоследствии может стать неизвестной, а также некоторые категорные приемы работы с ней.

Порождающий полный алгоритм определен теоретико-категорно. Такой способ определения выбран по двум причинам, во-первых, теория категорий считается завершенной, во-вторых, такой же способ был использован в работах [3], [4], [5] для «усиления» некоторых теоретико-алгоритмических рассуждений. Т.е. здесь востребована аксиоматизация, только «наивной», теоретико-множественной позиции недостаточно для синтеза способа порождения полного алгоритма. Эта «полнота» алгоритма не то же самое, что известная в науке «полнота алгоритма» по А. Тьюрингу. Использован подход Л.Э.Я. Брауэра к творчеству [7] для формирования критического подхода к основам теоретико-типовых формализаций.

Установлено, что теоретико-типовая проблема неполноты алгоритма связана с использованием типических последовательностей в качестве входных данных в алгоритм. При этом типические последовательности в алгоритме ведут себя так же, как введенные Брауэром беззаконные последовательности. Кроме того, те из них, что впоследствии образуют специальные последовательности, могут рассматриваться как один из частных случаев творческих последовательностей Брауэра.

Введение в рассмотрение подхода Брауэра к творчеству показывает способ переработки теоретико-типовых конструкций: как типических, так и специальных последовательностей, например, через «результат» А.Ш. Трулстра [7]: «Адекватной моделью интуиционистских вычислений являются преобразователи с тремя входами: конструктивные данные, результаты измерений (беззаконная последовательность), решения человека (творческая последовательность). А сами преобразователи могут считаться алгоритмическими (хотя их алгоритма мы не знаем)».

Этот полученный научный результат не является очевидным, что показала практика применения типов в информатике, например, спецификаций, стандартов и протоколов типов известных языков программирования (например, стандарт [8]).

Проще говоря, не очевидно, что типы должны перерабатываться так же, как и «интуиционистские вычисления» по Трулстра — некоторым вычислителем (алгоритмом) с тремя типами входов. Причем эти входы параллельны, по аналогии с «интуиционистскими вычислениями», переработка типических последовательностей алгоритмом используется для корректировки переработки творческих последовательностей этим же алгоритмом.

Этот теоретический результат может быть применен в информатике. Почему важен «вычислитель Трулстра»? Допущение, принятое в первой части исследования, рассматривать классы P и NP как алгоритмы, и аналогичным образом типы как алгоритмы, приобрело конкретизированный вид алгоритма — вычислитель Трулстра. Синтезировать такой вычислитель непростая задача, её следует выделить в отдельную работу, возможно, показав его на более простом примере, нежели классы P и NP или типы.

В настоящее время нет известных и широко признанных научных формализаций такого вычислителя, совершенно непонятно, какой алгоритм может его реализовать. Предполагается, что «на практике» (например, в каком-либо патенте) или «в информатике» (в какой-либо прикладной программе) кто-то мог получить такой вычислитель или приблизиться к его реализации. Но это станет ясно только после синтеза и обоснования его формулы. Синтез «вычислителя Трулстра» является актуальной научной и прикладной задачей, которую ещё предстоит осуществить.

Изыскиваются научно-технические пути формализации алгоритма искусственного интеллекта (ИИ), важными задачами при этом являются: теоретико-типовая по корректной переработке типов, точнее уточнение формализации типов, такой, что не содержит в своем составе типов, которые перерабатываются неполно, а также теоретико-алгоритмическая по синтезу полного алгоритма.

1.1. Структура исследования

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

Постановка задачи

2.1. Операции в алгоритме Маркова

В нормальном алгоритме Маркова определены, помимо, слов и однонаправленной стрелки (знака препинания), две операции: обращения и присоединения.

Эти операции хорошо иллюстрируют *проблему грамматики* над алфавитом.

Существо этой *проблемы грамматики* состоит в том, что буквы в разных естественных языках присоединяются друг к другу и образуют слова не случайным или хаотичным образом, а согласно правилам и *грамматике* этого языка. Но и сами слова и *грамматики* образовались не случайно, в них много общего в различных естественных языках. Фундаментальное и общее свойство всех людей, вне зависимости от их естественного языка, состоит в том, что они наделяют словом некоторый объект окружающего их мира, выделяя и воспринимая этот объект интуитивно и одинаково. Существуют, конечно, исключения из правил, но в подавляющем большинстве случаев это так. Аристотель смог обобщить слова, выделив категории, которые более полно представляют способы наделения объекта словом и понять сам принцип синтеза этих способов, который оказался логичным, как интуитивно, так и строго математически. Далее категории Аристотеля уточнялись, но принцип сохранился и сегодня, поэтому в научном обороте корректно общезначимое обобщение — аристотелевы категории.

Маркову пришлось отдельно определить простейшую *грамматику*, позволяющую составлять из букв, читай типических последовательностей, слова — специальные последовательности. Но саму *грамматику*, например, как внешний или внутренний морфизм между буквами, возможно тайный, не определил. Поэтому его слова «технически» существуют, но не принимается во внимание, что кто-то их отдельно определил.

При этом Марков строго математически определил такой «упрощенный» способ словообразования: буквы просто присоединяются друг к другу, образуя слово и имеется возможность циклически сдвинуть буквы относительно некоторой «начальной». Как и кем запускается процесс присоединения, как он останавливается, не принимается во внимание, некоторое управление существует, но не известно алгоритму. Поэтому, например, когда необходимо при помощи нормального алгоритма Маркова составить не слово некоторого нового, ранее не известного, а наоборот известного, естественного языка, то кто-то должен составить «правильную» последовательность присоединения букв и сообщить её алгоритму.

По существу, это уловки Маркова (Марков — брат, но истина дороже), которые должны были вызывать обоснованные сомнения на этапе их защиты в составе рассуждений теории.

Сами рассуждения Маркова содержательны и прогрессивны.

Поэтому заметим, что эти операции представимы также стрелками, причем эти стрелки обозначают обособленные морфизмы в терминологии, введенной в работах [3], [4], [5].

Т.е. такие, что способ выполнения их предписания единственен. Тогда в нормальном алгоритме Маркова можно было определить три стрелки, а не одну, нарисовав их, например, «красиво» или «вычурно» дабы отличать от однонаправленной стрелки!

Таким образом, вместо способа формализации полного алгоритма научной общественности Марковым был продемонстрирован частный случай неполного алгоритма!

2.2. Операция «аппликации» Алонсо Чёрча

Операция аппликации (прикладывания, присоединения), введенная А. Черечем в λ-исчислении, трактуется как алгоритм вычисляющий результат по заданному входному значению и записывается как $f\ a$, что соответствует традиционной в «классической» математике и теоретико-множественном, наивном подходе записи $f\ (a)$.

Однако имеется крайне «неприятная оговорка»: «в этом смысле аппликация может рассматриваться двояко: как результат применения f к a, или же как процесс вычисления этого результата». В первом смысле операция аппликации — это стрелка (морфизм) со значением, во втором, алгоритм.

В науке известно, что логика высказываний *соответствует* простому типизированному λ-исчислению, логика высказываний второго порядка — полиморфному λ-исчислению, исчисление предикатов — λ-исчислению с зависимыми типами.

Рассмотрим внимательно конструкцию f к a.

Можно ли подставить вместо f группы переменных — нет, в место a – можно.

Во-первых, такие ничего не значащие буквы, как f применяются в математике, во-вторых, с позиций конструктивной математики таковые весьма «опасны», так как нет способа обнаружить и распознать f среди других таких же f, а если такой способ есть, то возможно, что он оперирует не всеми классификаторами, а их возможные композиции существенно влияют на конечный результат. Это очень специфическая математическая конструкция, которую далее покажем в известной теории, в которой способы работы с ней являются наглядными, понятными и безопасными так как о такие буквы можно разделять между собой по формуле правой части.

В работе [5] показано, что стрелка (морфизм) вполне себе алгоритм, причем определенного типа — $type_K$. Поэтому, введя в рассмотрение такой тип как $type_K$ «вроде, как» и нет никаких несоответствий идее Черча, она подтвердилась через результат работы [5]. Тогда можно сказать, что исходную идею Черча можно рассмотреть как недоопределённую по f через $type_K$, тогда нет никакой проблемы в его рассуждениях. Проблема есть, и эта проблема кроется в самой конструкции f к a, в которую вводить $type_K$ необходимо математически корректно. Возможно ли корректное введение $type_K$ в конструкцию f к a?

Из работы [5] очевидно, что теоретико-категорная конструкция вида:

$$X \overset{a}{_f} Y\ ,$$

где:

XY — объекты;

f — морфизм (стрелка);

a — значение морфизма.

значительно выразительнее конструкции Чёрча: $f\ a$, так как дополнительно оперирует словами, а конструкция вида

$$X^{a}_{\text{type}_{K}:f} Y,$$

где $type_K$ — ассоциатор морфизма, есть доопределённая по f через $type_K$.

Обратим внимание, на то, что выше выделено слово: *соответствует*, т.е. имеется некоторая конструкция: $f\ a\twoheadrightarrow$Z, где «$\twoheadrightarrow$» — это морфизм, буквально означающий *соответствует* высказыванию Z тогда можно определить и $f\ a\twoheadrightarrow$Z, где «$\twoheadrightarrow$» — *несоответствует*. Здесь морфизм $f\ \ a$ — это теоретико-*n*-категорный ассоциатор морфизма «$\twoheadrightarrow$», высказывания Z, где Z — пропозиционная (*proposition* — лат.) переменная. Покажем такую конструкцию иначе, например, как: $X_f^a Y \twoheadrightarrow$ Z, или $X^{a\twoheadrightarrow Z}_{f}\ Y$, или $^{a\ \twoheadrightarrow}_{type_K:f}\ ^{Z}Y$, или $X^{a}_{type_K:f}\twoheadrightarrow Z^Y$. Так какая из них математически корректна? С категорных позиций слово «*соответствует*» может интерпретироваться различным образом. Наиболее близкой по смыслу идее Чёрча является формула: $:f\ \ X_{\mathrm{type_K}}\ ^{a\ \twoheadrightarrow\ Z}:f^Y$ т.е. *соответствует* при определенном значении – *a*. В работе [5], например – это формула: $X^{\ a}_{\mathrm{type_K}}:f\twoheadrightarrow Z^Y$ т.е. т.е. *соответствует* для всех значениями – *a*. Как видно, математически это далеко не одно и тоже.

Таким образом, «просто доопределить по type$_K$» конструкцию Чёрча не получится. В формуле: $X^{\ a}_{\mathrm{type_K}}:f^Y$ для двух объектов *X, Y* может быть несколько *f*, нескольких типов type$_K$, с несколькими значениями *a*, но более важно, что для двух объектов *X, Y* может не быть, ни при каких условиях, каких-либо конкретных морфизмов *f*, определённого типа type$_K$.

Также некоторые из морфизмов *f*, определённого типа type$_K$ для двух объектов *X, Y* всегда будут приводить к зацикливанию алгоритма, перерабатывающего такую формулу, для других никогда или редко, например, в зависимости от конкретизированных объектов *X, Y,* для других таких пар объектов всё будет иначе.

2.3. Операция «обрезания» Воеводского

В работе [2] упоминается, введенная В.А. Воеводским операция обрезания (truncation).

Следуя выше приведенной логике, в отношении операций, обозначим операцию обрезания (truncation) Воеводского специальной, «вычурной» стрелкой «$\twoheadrightarrow$». Эта стрелка есть волюнтаристское решение на совершение отдельного предписания алгоритма.

Положим, что для стрелки такого вида – это решение может быть подтверждено «как правильное», при каких-либо ограничениях. Определим также стрелку, когда такое решение может быть подтверждено как «не правильное»: «$\twoheadrightarrow$». Запишем общие категорные формулы такой операции:

$$C \twoheadrightarrow Z \tag{1}$$

$$C \twoheadrightarrow Z \tag{2}$$

$$C \Rightarrow Z \tag{3}$$

$$C \Rightarrow Z \tag{4}$$

где:

C – это некоторая категория, в теоретико-категорном смысле;

Z – слово или в общем случае высказывание.

Здесь можно было бы и не писать категорию *C*, а показать простое множество в теоретико-множественном смысле, но не хотелось бы упрощать идею Воеводского, так как таковая может быть показана, как в теоретико-множественном, так и теоретико-категорном подходе, поэтому выбран второй, старший подход.

Кроме того, если подставить множество, то может оказаться так, что будут повторены выводы об операции аппликации А. Чёрча, показанные выше.

2.4. Буквы беззаконны по Брауэру

Этот подзаголовок можно завершить как вопросительным, так и восклицательном знаком. Проведем мысленный эксперимент в «стиле» Л.Э.Я. Брауэра:

«Пусть:

1. Имеется алфавит из букв.

2. Определена однонаправленная стрелка, позволяющая присоединить одну букву к другой.

Допустим, что буквы алфавита присоединяются друг к другу в произвольной последовательности, при этом длина слова ничем не ограничена.

Среди букв нет известных наблюдателю.

Тогда среди полученных слов также нет известных.

Все такие слова беззаконны, аналогично значению морфизма измерителя (датчика)».

2.4.1. Слова по Брауэру, полученные в результате творчества

Продолжим мысленный эксперимент:

«Пусть:

1. Алфавит состоит из известных букв.

2. Определена однонаправленная стрелка, позволяющая присоединить одну букву к другой.

Допустим, что буквы алфавита присоединяются друг к другу в произвольной последовательности, при этом длина слова ничем не ограничена.

Тогда среди всех слов будут такие, что известны в некотором словаре, а также такие, что неизвестны.

Наметился некоторый процесс творческого перебора комбинаций букв такой, что этот процесс может закончится успехом или неудачей. Положим, что если слово известно, то это успех. Обозначим такой успех специальной стрелкой «$\Rightarrow$», неудачу – «$\Rightarrow$».

C$\Rightarrow$Z (3)

C$\Rightarrow$Z (4)

где:

C – это некоторая категория;

Z – слово или в общем случае высказывание.

2.5. Конкретизация задачи

Заметим, что формулы (1), (2) и (3), (4) – «*похожи*», хотя описывают различные математические объекты, формулы (1) и (2) теоретико-типовые конструкции, формулы (3) и (4) логические, интуиционистские, теоретико-алгоритмические.

Необходимо определить возможность и основу для обобщения над ними, тем более, что такая практика широко распространена в математике в виде метафор (обобщений над непознанными классами): «подобны», «соответствуют», «эквивалентны».

Решение задачи

3.1. Тип «Boolean»

Запишем категорные формулы этого типа, обозначив категорию истинности как понятие, содержащее вывод высказывания Z на естественном языке: «Истинно», обозначив стрелкой «⇒» успех достижения «Истинно», а стрелкой «⇏» неуспех достижения этого значения.

$$C \Rightarrow \mathrm{Z}$$
$$C \not\Rightarrow \mathrm{Z}$$

Заметим, что здесь некоторое «отождествление», «эквивалентность», «подобие» слова естественного языка «Истинно» и некоторого объекта окружающего нас мира *C*, в отношении которого допущено, что этот объект, представимый категорией, можно сжать и закодировать словом, показан через взаимодействие (взаимосвязь, действие, морфизм, специальную вычурную стрелку). Эта стрелка может быть потенциально реализуемой не всегда, во-первых, в том случае, когда объект окружающего нас мира изучен с приемлемым, текущим уровнем знания о нем, во-вторых, такое знание может быть пересмотрено в дальнейшем, в-третьих, оба эти знания можно характеризовать некоторых отрезком времени, когда таковое существует, в-четвертых, для искусственных миров, где сами эти объекты могут быть и противолежащими.

Положим, что Z — это не только слово, но и цифра, например, 24, может быть строка символов, например, 24 В, но может быть одновременно и другая цифра например «единица» в какой-либо системе исчисления, например, двоичной, возможно это также цифра к которой дополнительно приложена какая-либо кодовая таблица *T* (например, код МТК-2, на базе которого разработаны коды КОИ-7 и КОИ-8, или UNICOD), т.е. в общем случае — ассоциатор, обозначим его красивой буквой $\mathfrak{H}$, где эта буква также стрелка, поэтому запишем её через терм «:», то получим цепочку беззаконных «отождествлений»:

$$C \Rightarrow^1 Z\mathfrak{H} :\Rightarrow^2 V\mathfrak{H} :\Rightarrow^3 \ldots \mathfrak{H} :\Rightarrow^n W$$

$$C \not\Rightarrow^1 Z\mathfrak{H} :\not\Rightarrow^2 V\mathfrak{H} :\not\Rightarrow^3 \ldots \mathfrak{H} :\not\Rightarrow^n W$$

например, как описано:

$$C \Rightarrow^1 \text{"Истинно"}\mathfrak{H} :\Rightarrow^2 \text{"true"}\mathfrak{H} :\Rightarrow^3 24\mathfrak{H} :\Rightarrow^4 24\ \text{"B"}\mathfrak{H} :\Rightarrow^5 \ldots \mathfrak{H} :\Rightarrow^n 00000001$$

$$C \not\Rightarrow^1 \text{"Ложно"}\mathfrak{H} :\not\Rightarrow^2 \text{"false"}\mathfrak{H} :\not\Rightarrow^3 0\mathfrak{H} :\not\Rightarrow^4 24\ \text{"B"}\mathfrak{H} :\not\Rightarrow^5 \ldots \mathfrak{H} :\not\Rightarrow^n 00000000$$

Это формальное представление беззакония типов, оно же известно в науке под другими названиями, например, — кодирование, оно же уникальное кодирование, оно же таблица замен, оно же перевод с одного искусственного языка на другой такой искусственный язык без должного основания. Пусть здесь не смущают слова естественного языка, так как таковые в математическом смысле — беззаконны. Фактически — это известный всем уникальный шифр.

Предположим, что идея учинить математическое «беззаконие» противоречит целям любых теорий, в том числе и типов, однако как математически, так и эвристически — на уровне приведенного примера понятно, что беззаконие типов широко распространено в информатике.

Все показанные выше примеры *типичны* без учета $\mathfrak{H}$, с таковой, безусловно, теряют свои типические свойства, однако сложилась практика этого не замечать. Сама эта буква и ассоциатор $\mathfrak{H}$: есть переменная, допускающая подстановку множества переменных.

Очевидно, что *Z, V, W* можно переставить так, чтобы привести такое «отождествление» к абсурду, породив парадокс, в частности типических последовательностей, такой парадокс назовем «парадоксом Мартина-Лёфа».

Наличие парадокса требует поиска путей его преодоления, традиционным путем может выступить аксиоматизация и построение категорной логики, где все такие «отождествления» будут полагаться слабыми *n*-категориями, имеющими различный логический смысл, при разном положении ассоциаторов $\mathfrak{H}$.

Следует принять, ранее введенный тип морфизма type_K как некоторую композицию стрелок, где некоторые из них есть стрелки стрелок — морфизмы морфизмов — ассоциаторы, определив type_K как слабую *n*-категорию, которая при определенных условиях может стать сильной *n*-категорией, если для неё удастся определить только ассоциативные композиции.

Далее будем понимать тип морфизма type_K именно так, более не возвращаясь к его существу.

3.1.1. Способ порождения полного алгоритма

Представим выборки входных данных в виде теоретико-категорной формулы.

Положим, что на вход алгоритма поступают типические последовательности.

Предположим, что некоторый алгоритм способен породить предписания, которые потенциально могут переработать такие входные выборки данных.

Здесь введено предложение рассмотреть «способность алгоритма породить предписания переработки», при этом совершенно неясно, что будет, если таковые удастся породить, алгоритм сможет переработать данные или нет?

Обозначим *C* буквой — категорию, следуя теоретико-категорной традиции.

Определим формулу категории, также следуя принятым правилам:

$$C = \mu : X \to Y = X \xrightarrow{\mu} Y \tag{5}$$

где:

μ — морфизм, он же показан стрелкой;

XY — объекты, т.е. традиционно для теории категорий, здесь не привносится никаких новых подходов.

Объекты *XY* определим как слова в алфавите Маркова *A = X,Y,Z,…* — этот способ определения объектов отличает настоящее исследование.

Введем в эту формулу (5) ассоциатор type_K через терм «:» — это категорный ассоциатор типа морфизма, тогда:

$$C = \text{type}_{\mathcal{K}} : \mu : X \to Y = X \xrightarrow{\text{type}_{\mathcal{K}} : \mu} Y \tag{6}$$

Введем значение морфизма буквой *a* — это переменная (используем букву, которая была показана ранее при анализе результатов А. Чёрча).

Получим формулу вида:

$$C = \text{type}_{\mathcal{K}} : \mu : f : X \to Y = X \xrightarrow{\text{type}_{\mathcal{K}} : \mu : a} Y \tag{7}$$

Способ, который применен для определения ассоциаторов традиционен для теории *n*-гкатегорий, запись через терм «:» — новаторство, для краткости, т.е. для того, чтобы не строить стрелку над стрелкой, термы читаются слева на право, например, для формулы (7), как стрелка type_K над стрелкой μ, где таковая стрелка над *a*.

Определим вывод, который нельзя считать нормальным по Маркову в категории *C*:

$$C_M = \text{type}_{\mathcal{K}} : \mu : a : XMY \tag{8}$$

где:

$M = \lambda \text{u}_{\text{aext}} \tau \tau_{\text{aext}} \varphi \Pi \text{O}_{\text{aint}} \text{O}_4 \dots \text{O}_x M^{SM} / \alpha \beta \delta \gamma \chi \kappa \mu$ – служебный алфавит, вывода, который нельзя считать нормальным по Маркову, применяемый для кодирования морфизмов, где:

M^{SM} – малые морфизмы;

знак «/» разделяет терминальные и нетерминальные символы языка;

буквы: ι – инициальный, τ – терминальный, φ – изморфный, λ – пустой морфизмы, п – – протоморфизм, $o_1 \dots o_x$ – обособленные морфизмы, например, нечеткие, $\varphi = \iota \chi \tau$ и χ – перекрещивание, $\alpha = \iota\gamma$ (альфа), $\beta = \tau\gamma$, $\delta = \alpha\gamma\chi\tau = \varphi\gamma$, γ – параллельное вхождение, $\kappa = \iota\tau$, $\mu = \tau\iota$, применяется для кодирования морфизмов, $\iota_{\text{aext}}, \tau_{\text{aext}}, \text{o}_{\text{aint}}$ – тайные морфизмы.

Сами морфизмы, при этом, рассматриваются как π – начальные, ε – обычные и η – заключительные марковские вхождения в алфавите Маркова $B = \varepsilon\eta\gamma$, где γ – знак препинания, при этом приходится отказаться от записи стрелками принятой Марковым и перейти к буквам алфавита *M*, «потеряв» одну из известных форм записи катгорных формул.

Покажем, что категория C_M может взаимодействовать с другими категориями, т.е. слева и справа в неё имеются морфизмы в том же алфавите *M* – это:

$$\mathcal{M}C_{\mathcal{M}}\mathcal{M} \tag{9}$$

Обозначим множество категорий пронумеровав их последовательно, по мере поступления из выборки входных данных, как 1,2,3,…,x, здесь все буквы справа от равно переменные, допускающие подстановку групп переменных:

$$C_{\mathcal{M}x} = \text{type}_{\mathcal{K}} : \mu : a : X\mathcal{M}Y \tag{10}$$

Морфизмами с одной стороны от C_M , читай теперь как C_{Mx}, пренебрежём, так как на настоящем этапе формализации, способ приема, фильтрации, аналогово-цифрового преобразования не вызывает интереса. Сторона, где не будут рассматриваться морфизмы зависит от выбранного направления индукции, получим:

$$\lambda C_{\mathcal{M}x}\mathcal{M} \tag{11}$$

Пустота λ с одной стороны от категори C_{Mx} несколько упростит дальнейшее понимание преобразований. Обозначим категорию всех таких категорий как $C^{\lambda}_{\mathcal{M}}$, тогда:

$$C^{\lambda}_{M} \overset{k}{=} x : C_{Mx} \tag{12}$$

где:

λ в верхнем индексе буквально обозначает, что вокруг этой категории λ – пусто;

предписание *x* – эндоморфизм в C_{Mx}, равносильный просто букве: C_{Mx}, записанный как теоретико-категорный морфизм, иначе говоря, как: $C^{\lambda}_{M} \overset{k}{=} x : C_{Mx} = C_{Mx} \overset{x}{\circlearrowright}$, где вместо привычной стрелки «→», используется стрелка ↻, обозначающая: «на самого себя», тогда можно записать $C^{\lambda}_{M} \overset{k}{=} \lambda C_{Mx}$ без *x*, не выделяя такой морфизм отдельно от алфавита *M*, т.е. все морфизмы положены определенными в *M*, в том числе и эндоморфизмы, показав корректно, что слева от неё λ – пусто, что неудобно делать в записи с «x:».

Сама такая категория $C^{\lambda}_{\mathcal{M}}$ существует, является точным математическим представлением для некоторых выборок данных, которые имеют место быть от объектов окружающего нас мира.

Можно ли подставлять вместо переменных *M,x* группы переменных – можно, можно ли тоже самое делать с переменной *C* – нет, для корректной подстановки переменных вместо *C*, необходимо вернуться в алфавит *A*, *C* буква здесь дань теоретико-категорной традиции (так понятнее многим) и буквальное предписание заменить (заместить) буквы алфавита *A* и их морфизмы (стрелки) общим обозначением *C* – единственной буквой, для краткости;

$\overset{k}{=}$ – символ, который означает конструктивное равенство.

Рассмотрим возможности категории: $C_M^{\lambda} \overset{k}{=} C_{Mx}$ – это все простые категории, пронумерованные, т.е. расслоенные по Ловеру, направление стрелок в них может быть любым.

Тогда, конечно, объекты *D1…D5,* можно дополнительно конкретизировать с учетом стрелок.

Например, в промышленной автоматизации принято рассматривать два таких направления: слева-направо и справа-налево, и обозначать соответственно: DI1 и DO1, но теоретически универсальным будет обозначение: *DM1, DM2, DM3,…*

Формула: $C_M^{\lambda} \overset{k}{=} C_{Mx}$ описывает, например, случай «слепого приема (переработки данных)» входных выборок данных.

При «неслепом приеме (переработке данных)» априори известна некоторая категория, которая позволяет «понять», что делать с $C_{\mathcal{M}}^{\lambda}$. Эту категорию обозначим как $C_{\mathcal{M}}^{\mathcal{U}}$, где буква $\mathcal{U}$ выбрана очень похожей на букву, использованную Марковым для обозначения алгоритма.

Представить себе такие категории можно, показав некоторую упрощенную структуру для дополнительного алгоритма $C_{\mathcal{M}}^{\mathcal{U}}$, что можно проиллюстрировать примером – рисунок 1.

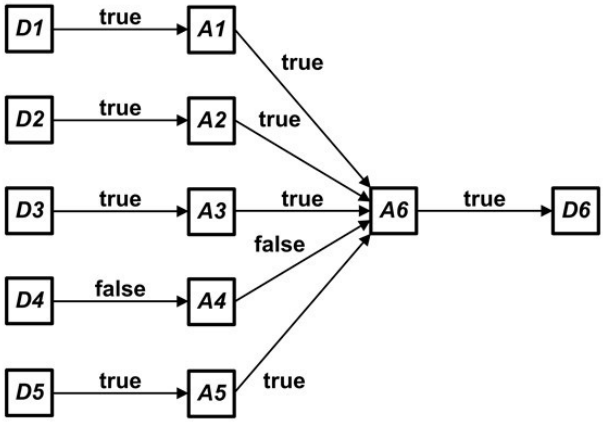

Рисунок 1 - Пример композиции C_{Mx} и $C_M{}^U$

DOI: https://doi.org/10.60797/itech.2026.9.1.1

где:

D1, D2, D3, D4, D5, D6, A1, A2, A3, A4, A5, A6 – слова;

«→» – стрелка – обозначение морфизма;

true,false – значение морфизма.

Также придется вернуться к записи $C_M^{\lambda} \overset{k}{=} C_{Mx}$, показав, что справа от C_{Mx} появляются морфизмы, на рисунке 1 – это: A1→A6, A2→A6, A3→A6, A4→A6, A5→A6 в категорию $C_{\mathcal{M}}^{\mathcal{U}}$: A6→D6, при этом, строго говоря, категорию $C_{\mathcal{M}}^{\lambda}$ уже рассматривать некорректно, так как таковая введена с условием, что вокруг неё λ–пусто, а здесь появляются новые объекты и морфизмы, которые необходимо определить, поэтому будем рассматривать две взаимодействующие категории C_{Mx} и $C_M{}^U$ и некоторую категорию, которая включает их – это категория $C_{\mathcal{M}}^{\mathcal{M}}$, где верхний индекс показывает на появление морфизмов, которые представимы в служебном алфавите *M*, вывода, который нельзя считать нормальным.

Проще говоря, во всех формулах, где есть *C*, показывается некоторый конкретизированный набор морфизмов, читай некоторая сигнатура категории, поэтому постоянно приходится уточнять, что именно в верхнем индексе при *C*.

Зададимся вопросом: «Известна ли категория $C_{\mathcal{M}}^{\mathcal{U}}$?»

Это зависит от позиции наблюдателя, если наблюдатель – разработчик, то он определяет $C_{\mathcal{M}}^{\mathcal{U}}$ сам лично, свой волей и действием, зная, что включено в её состав, если наблюдатель – потребитель, то он ничего не знает о $C_{\mathcal{M}}^{\mathcal{U}}$, пока ему каким-либо образом не удастся заглянуть в неё, например, наблюдатель – разработчик не покажет её состав и структуру морфизмов.

Например, имеется возможность наблюдения за техпроцессом, читай морфизмами, показанными на рисунке 1, пусть к ним добавлен морфизм $A6 \xrightarrow{\text{true}} D6$ – это *вывод* по следующей формуле:

$$C_{Mx}C_M^U \overset{k}{=} \begin{matrix} D1 \xrightarrow{\text{true}} A1 \xrightarrow{\text{true}} \\ D2 \xrightarrow{\text{true}} A2 \xrightarrow{\text{true}} \\ D3 \xrightarrow{\text{true}} A3 \xrightarrow{\text{true}} \\ D4 \xrightarrow{\text{false}} A4 \xrightarrow{\text{false}} \\ D5 \xrightarrow{\text{true}} A5 \xrightarrow{\text{true}} \end{matrix} A6 \xrightarrow{\text{true}} D6 \qquad (13)$$

Очевидно, что «неочевидные» морфизмы A1, A2, A3, A4, A5 → A6 дублируют морфизмы D1→A1, D2→A2, D3→A3, D4→A4, D5→A5, и имеют «техническое» назначение.

Заметим, что (13) – это полноценная категорная формула, причем таковая действительно имеет место быть в объектах окружающего нас мира, где:

$$C_M^U \overset{k}{=} \begin{matrix} A1 \xrightarrow{\text{true}} \\ A2 \xrightarrow{\text{true}} \\ A3 \xrightarrow{\text{true}} \\ A4 \xrightarrow{\text{false}} \\ A5 \xrightarrow{\text{true}} \end{matrix} A6 \xrightarrow{\text{true}} D6 \qquad (14)$$

Но, тогда получается, что A1, A2, A3, A4, A5 входят в состав обеих категорий C_{Mx} и $C_{\mathcal{M}}^{\mathcal{U}}$?

Именно так, можно показывать формулу (14) открытую справа в алфавите *M*, т.е. не включая A1, A2, A3, A4, A5, но это не верно, как математически, так и технически. Технически, если положить, что $C_{\mathcal{M}}^{\mathcal{U}}$ – это, например, алгоритм программируемого логического контроллера (ПЛК), то, очевидно, что сигналы от датчиков и исполнительных механизмов, показанные как D1, D2, D3, D4, D5 необходимо «жестко», читай безальтернативно, вручную привязать к конкретизированным входам ПЛК: A1, A2, A3, A4, A5.

Формула (14) – это простой пример дополнительного, скрытого алгоритма, дополняющего неполный алгоритм до полного.

Реальная формула алгоритма $C_{\mathcal{M}}^{\mathcal{U}}$ может быть значительно сложнее, причем настолько, что понять этот алгоритм без подсказки наблюдателя – разработчика невозможно, даже в ходе длительного наблюдения за входами и выходами этого алгоритма, например, рисунок 2:

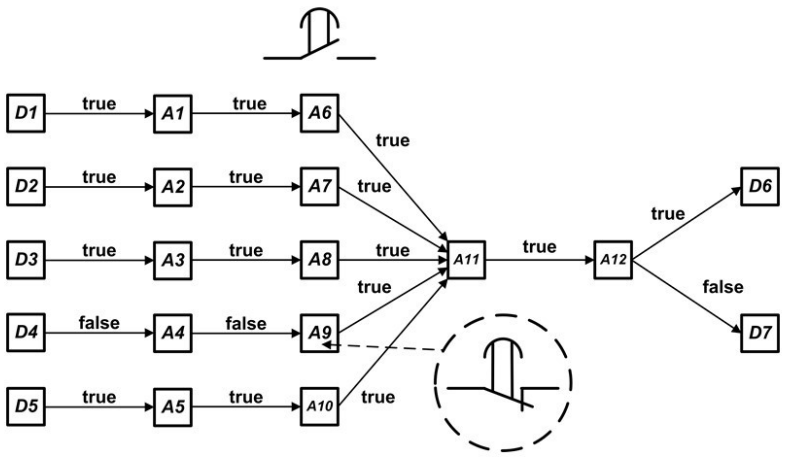

Рисунок 2 - Пример композиции C_{Mx} и $C_M{}^U$ более сложной категории

DOI: https://doi.org/10.60797/itech.2026.9.1.2

где:

D1,…,D7, A1,…, A12 – слова;

A6, A7, A8, A10 – замыкающие контакты реле времени с задержкой при срабатывании показаны сверху над схемой стандартным обозначением;

A9 – размыкающие контакты реле времени с задержкой при срабатывании, показан в круге с пунктирной линией;

A12 – можно представить как слово ошибок, если таковое равно нулю, то ошибок нет и вывод D6, не равно нулю – ошибки есть – вывод D7.

Сам техпроцесс можно себе представить, как трубопровод с запорными клапанами, где D4, A4, A9 – байпасный трубопровод, а задержка во времени обусловлена возможной различной длиной трубопровода на разных инженерных объектах, при сохранении для них всех общего алгоритма, тогда задержки выставляются дополнительно для каждого такого объекта, в зависимости от реальной протяженности трубопроводов (для разных архитектурно-строительных решений на объектах).

Это пример (рисунок 2) также относительно прост, хотя и сложнее, представленного на рисунке 1.

В формулу (13) довольно легко и без объяснений был добавлен вывод A6 → D6, потому, что таковой наблюдаем, т.е. известен всем возможным наблюдателям, но его «отождествить» с категорией C_{Mx} можно только в некоторых простых случаях.

Попробуем «отождествить» интуитивно. Будем помнить, что «отождествить» без знания $C_{\mathcal{M}}^{\mathcal{U}}$ – это проявить разумный волюнтаризм, возможно даже успешный, возможно логически обоснованный, но незаконный.

Таковой незаконен так как кажется и не более, что волюнтаристские действия обоснованы, ведут к осмыслению $C_{\mathcal{M}}^{\mathcal{U}}$, в предположении о незаконном «подобии» или «эквивалентности» некоторых замещаемых, неизвестных объектов из состава категории $C_{\mathcal{M}}^{\mathcal{U}}$.

Какова возможная альтернатива для такого волюнтаризма – это полный перебор, т.е. имеет место быть случай неравенства классов P и NP, но только для одного из наблюдателей. Для наблюдателя-разработчика классы P и NP по-прежнему равны, т.е. проблема: «равенства классов P и NP» (проблема перебора)» относительна (и относительно наблюдателя).

Причем имеет место быть только для наблюдателей не знающих $C_{\mathcal{M}}^{\mathcal{U}}$, таковых может быть достаточно много, потенциально все существующие объекты окружающего нас мира, за исключением одного или в некоторых случаях двух.

Например, наблюдатель-шифровальщик мог просто скрыть часть алгоритма, отвечающую за генерацию типических последовательностей, например, придумав и скрыв ключ шифрования. Вернемся к примеру, показанному на рисунке 1.

Можно представить, что между A1, A2, A3, A4, A5 $\rightarrow\cdots\rightarrow\cdots\rightarrow$ A6 имеются совершенно неизвестные фрагменты для наблюдателя – потребителя, тогда формула (13) для него будет выглядеть так:

$$C_{\mathcal{M}x}C_{\mathcal{M}}^{\mathcal{U}} \rightsquigarrow Z \qquad (15)$$

где ему известны C_{Mx} и *Z*, а все морфизмы до *Z* «кажутся» (предполагаются) «неассоциативными», т.е. для него это слабая -категория, а категория $C_{\mathcal{M}}^{\mathcal{U}}$ является объектом поиска, анализа и классификации, тем или иным способом. Единственное, что ему известно об этой категории – на её входе булевы значения и вывод из неё в объект *Z* может быть булев, т.е. имеется знание о типе $type_K$ значений морфизмов слов.

Что удалось получить в формуле (13), новый алгоритм переработки?

Нет в этой формуле получен способ порождения полного алгоритма и показан для неё неполный алгоритм. Перерабатывать данные по категорной формуле (13) алгоритм сможет только, если известна её правая часть, но проблема состоит в том, что частично известна левая.

Таким образом, явно обозначилось деление алгоритмов, как минимум, на порождающий и перерабатывающей. Очевидно, что можно принять во внимание способ модернизации этих алгоритмов, а также способ анализа, или подбора, или перебора, или взлома дополнительного, скрытого алгоритма в составе порождающего, тогда имеет смысл включить в общую формализацию алгоритмов разделяющий признак назначения: порождающие, перерабатывающие, модернизирующие и вскрывающие.

Учет разделения алгоритмов в их общей научной формализации позволит более корректно и точно рассматривать проблему «равенства классов P и NP» (проблему перебора)», так как таковая свойственна не всем из них и зависит от позиции наблюдателя.

3.2. Методика определения стрелок в формулах (1-4) и (15)

Из способа определения полного алгоритма следует, что для обобщения над формулами (1), (2) и (3), (4) необходимо математически определить:

1. Полный алгоритм.
2. Наблюдателей.
3. Наблюдаемую часть алгоритма C_{Mx}.
4. Дополнительный алгоритм $C_{\mathcal{M}}^{\mathcal{U}}$ в составе полного, который неизвестен конкретизированному наблюдателю.
5. Наблюдаемый вывод.
6. Предположение о дополнительном алгоритме $C_{\mathcal{M}}^{\mathcal{U}}$.
7. Определить $type_K$ как типическую последовательность или напротив, как специальную.
8. Абстракцию «точный вывод».

Как только это наблюдаемый вывод будет определен, то его можно подставить в формулы (1), (2) и (3), (4) вместо переменной *Z*.

Ввести:

– переменную для наблюдателей, например, автор – *author* и слепой – *blind*;

– заменить все, ранее использованные, «вычурные» стрелки одной, единственной такой стрелкой, например, «⇝»;

– переменную для «точного вывода», например, обозначив буквой *W*, выбрав её значение «нейтральным» для любого алгоритма, например, "Работаю" (это сделано для упрощения, в прикладных задачах эта буква может быть заменена группой переменных);

– переменную для подмены $C_{\mathcal{M}}^{\mathcal{U}}$, например, $\widetilde{C_{\mathcal{M}}^{\mathcal{U}}}$;

– переменную для типической последовательности, например, ТП.

Показать подмену, например, так:

$$U \overset{k}{=} \left\{ \begin{array}{l} W = \text{“Работаю”} \\ \text{type}_K = \text{“Типпос”} \\ author : \text{type}_K : C_{Mx} C_M^U \to W \\ blind : \text{type}_K : C_{Mx}(C_M^U) \rightsquigarrow Z \end{array} \right\} \tag{16}$$

Но такого представления, как показано в формуле (16) недостаточно, необходимо показать как конкретно получена $\widetilde{C_{\mathcal{M}}^{\mathcal{U}}}$, т.е. избранный способ, читай алгоритм $U_{overcoming}$ преодоления неопределённости, пусть простая линейная таблица, но всё же алгоритм. Алгоритм $U_{overcoming}$:

$$U \overset{k}{=} \begin{cases} W = \text{“Работаю”} \\ \text{type}_K = \text{“Типпос”} \\ author : \text{type}_K : C_{Mx} \widetilde{C_M^U} \to W \\ blind : \text{type}_K : C_{Mx} \widetilde{C_{\mathcal{M}}^{\mathcal{U}}} \rightsquigarrow Z \\ U_{\text{overcoming}} = C_M^U \rightsquigarrow \widetilde{C_{\mathcal{M}}^{\mathcal{U}}} \end{cases} \tag{17}$$

Тогда формулу (17) можно приложить к аппликации А. Чёрча, получив более конкретизированный вид для f. В правой части этой формулы применяется такая же бессмысленная буква *C*, однако вместо неё можно подставлять слова и морфизмы по формулам (8) и (10), определив их как *X*, *Y* в алфавите Маркова и использовав служебный алфавит *M* для кодирования стрелок с выводом, который нельзя считать нормальным по Маркову, получив:

$$f \overset{k}{=} \begin{cases} W = \text{“Работаю”} \\ \text{type}_K = \text{“Типпос”} \\ C_{Mx} = type_K : \mu : a : XMY \\ author : \text{type}_K : C_{Mx} \widetilde{C_M^U} \to W \\ blind : \text{type}_K : C_{Mx} \widetilde{C_{\mathcal{M}}^{\mathcal{U}}} \rightsquigarrow Z \\ U_{\text{overcoming}} = C_M^U \rightsquigarrow \widetilde{C_{\mathcal{M}}^{\mathcal{U}}} \end{cases} \tag{18}$$

В формуле (18) буква f, введенная Чёрчем, определена для всех наблюдателей. Алгоритм $\mathcal{U}_{\text{overcoming}} = C_{\mathcal{M}}^{\mathcal{U}} \rightsquigarrow \widetilde{C_{\mathcal{M}}^{\mathcal{U}}}$ – это творчество по Брауэру, а категория $C_{\mathcal{M}x} = \text{type}_{\mathcal{K}} : \mu : a : X\mathcal{M}Y$ – это генератор беззаконных последовательностей. Очевидно, что выполнять $U_{overcoming}$ можно с различной степенью успешности и неуспешности, корректируя результат по наблюдаемой части C_{Mx}.

Формула (18), в том числе, показывает данные, которыми будет оперировать «вычислитель Трулстра». Такой вычислитель возможно синтезировать. Для его синтеза необходимо классифицировать категории, т.е. приложить комплекс классификаторов, способных переработать входные данные вида: $C_{\mathcal{M}x} = \text{type}_{\mathcal{K}} : \mu : a : X\mathcal{M}Y$, содержащие взаимодействующие слова *XMY,* в расширенном и упорядоченном алфавите Маркова $\mathcal{A}_{\mathcal{M}\mathcal{M}_{\mathcal{K}}}^{+-2}$,

где *M* – служебный алфавит, вывода, который нельзя считать нормальным по Маркову, а M_K также служебный алфавит, такого же вывода, но используемый в канале управления *cNne*–схемы алгоритма Маркова, по существу – множества, в категории с классификатором – топосы.

Так как переработать необходимо фактически систему (18), т.е. структуру у которой предполагается наличие нескольких входов и возможно нескольких выходов, то необходим и соответствующий ей алгоритм, например, такой как показан в работах [5], [9], [10].

Предложение приложить именно *cNne*–схемы алгоритма Маркова и, как следствие сеть Маркова непрямого распространения, в качестве «вычислителя Трулстра» – является новаторством. Однако, пока, не предложено какой-либо научно обоснованной альтернативы. Появление таковой в научном обороте позволит сравнить подходы, с предложенным здесь, пока провести такое сравнение не с чем. Поэтому *cNne*–схема алгоритма и сеть Маркова непрямого распространения показаны как реализуемый (конструктивный) способ переработки формулы (18).

3.3. На сколько типы морфизмов применимы в алгоритме ИИ?

Можно заметить, что человеческий интеллект не получает данные от своего организма в вольтах или амперах, целым значением или с плавающей точкой, тем более тип, составленный из них, и это незнание не мешает ему мыслить. Отсюда следует первое наблюдение – тип не нужен человеческому интеллекту, чтобы функционировать.

Однако, человеческий интеллект пытается оценить объекты окружающего его мира и интуитивно разделяет их первоначально на «классы», выделяет в них «свойства», пытаясь обобщить и сформировать «первое» представление об объекте и его возможных взаимодействиях (морфизмах).

При этом «классы» и «свойства» выступают некоторыми прообразами будущего представления об объектах и их морфизмах. Далее человек пытается зафиксировать «классы» и «свойства» словами, ввести для них меры. Потом убеждается, что не все «классы», «свойства», «меры» одинаковы, но есть такие, что для них можно ввести «тип», использовав букву, или слово, или их композицию, и поделиться этим «типом» с окружающими. Такой подход свойственен всем людям.

В настоящее время, благодаря науке, этот простейший человеческий опыт обобщен и формализован. В реальном мире человек опирается на уже известные такие «типы», сообщенные ему предками, становясь ученым, берет их под сомнение, совершенствует, пересматривает и сообщает об этом другим людям. Поэтому «типы» естественным образом начинают сопровождать его с самого рождения.

По-видимому, такой же процесс будет ожидать и ИИ, но его можно упростить, сообщив ему только такие из них, что перерабатываются дедуктивными методами и, например, не приводят к зацикливанию алгоритма, а если приводят к таковому, то это может быть преодолено.

Заключение

Формула (18) — это более строгое определение равенства или неравенства классов P и NP. В ней же показаны: равенство таких классов для наблюдателя *author*, причина неравенства этих классов — *типические последовательности* для наблюдателя *blind*, а также продемонстрирован путь *«перебора» с творчеством* Л.Э.Я. Брауэра.

«Беззаконие» в теориях алгоритмов и типов «учинили»: А.А. Марков и П.Мартин-Лёфф, добавил такового в теорию типов В.А. Воеводский, работы А. Тьюринга и А. Чёрча нет смысла рассматривать в этом контексте, так как они достаточно далеко отстают от существа этой проблемы.

Это важное замечание в теориях алгоритмов и типов, так как необходимо быть внимательным к рассуждениям о творчестве Л.Э.Я. Брауэра, по-видимому, таковые имеют куда как более глубокий смысл для формализаций этих теорий.

«Мягкие» творческие действия в формуле (18) представимы слабыми категориями $\widetilde{C^{u}_{\mathcal{M}}}$, где перечень принимаемых и не принимаемых во внимание ассоциаторов способен существенно повлиять на алгоритм $U_{overcoming}$.

Проще говоря, с этими ассоциаторами должна быть полная ясность, необходимо сформировать полный перечень таковых и алгоритм $U_{overcoming}$, который может строго и обосновано математически «подключить» или «не подключать» таковые в процессе выполнения предписаний.

Однако это ещё не всё, что замечено в рассуждениях Брауэра и «беззаконных» операциях: Чёрча, Маркова и Воеводского. Что-то необходимо предпринять со всеми этими «вычурными» стрелками: «$\rightsquigarrow$», «$\twoheadrightarrow$», «$\models$», «$\rightarrowtail$».

Например, на практике эти стрелки «беззаконно» отождествляют, считают эквивалентными, подобными, похожими на какую-либо простую стрелку, это хорошо видно в работе Воеводского и известном соответствии (изоморфизме) Карри-Ховарда, они как бы бросают вызов фундаментальной идее Брауэра и всей интуиционистской логики исключения из рассуждений «Закона исключенного третьего», ведь отождествить или поставить в соответствие некоторой «сложной» интуиционистской логике можно булеву алгебру, намеренно и волюнтаристски «загрубив» вывод, например, до показанного в (18) — "Работаю".

В современных прикладных задачах, в частности, в промышленной автоматике такое «беззаконие» и волюнтаризм широко распространены, при этом «катастрофы вселенского масштаба» не произошло, заводы и фабрики работают, алгоритмы, введенные в автоматику таких предприятий, функционируют, а когда «третье все же оказывается дано», то в дело вступает ремонтный персонал и творчество Брауэра, следовательно, можно сделать вывод, что так делать (поступать) можно.

Открытым, дискуссионным, научным вопросом является «узаконивание» таких отождествлений, а также более строгая их формализация (определение четких «правил игры»).

Проще говоря, алгоритм должен понимать, что какой-либо из его выводов намеренно загрублен и включать в свой состав контр-алгоритм противодействия или минимизации ущерба от такового, например, в той же промышленной автоматизации известен алгоритм «преодоления дребезга электрических контактов», вполне себе понятный и используемый, а также «аварийные цепи» и подсистемы безопасности персонала.

Все эти рассуждения важны для будущей формализации алгоритма ИИ.

4.1. Выводы

Предложение приложить вычислитель Трулстра к типам является новаторством.

Такой способ переработки современных типов в известных теориях типов более адекватен введенным в них теоретическим допущениям. Соответственно, применяемые в настоящее время способы и приемы переработки типов, опирающиеся на эти теории, менее адекватны.

В этом вопросе нет расхождения между теорией и практикой, просто на практике сложилась некоторая «традиция» применения теоретико-типовых положений, которую целесообразно уточнить.

Теоретико-типовые рассуждения и следующие из них научные выводы и прикладные способы применения научных результатов не должны приводить к «проблемам» в алгоритме. Наличие «проблем» в алгоритме должно рассматриваться научным сообществом как основание к критическому взгляду на предлагаемые теории типов.

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. Марков А.А. Теория алгорифмов / А.А. Марков, Н.М. Нагорный. — 2-е изд., испр. и доп. — Москва : Фазис, 1996. — 493 с.

2. Ковалев С.П. Проблема обоснования в формальном представлении знаний / С.П. Ковалев, А.В. Родин // Вестник Томского государственного университета. Философия. Социология. Политология. — 2018. — № 46. — С. 22–29. — DOI: 10.17223/1998863X/46/3.

3. Коновалов В.А. Подход к синтезу алгоритма, перерабатывающего предписания искусственного интеллекта. Часть 1 / В.А. Коновалов // Вестник компьютерных и информационных технологий. — 2024. — Т. 21, № 10. — С. 50–58. — DOI: 10.14489/vkit.2024.10.pp.050-058.

4. Коновалов В.А. Подход к синтезу алгоритма, перерабатывающего предписания искусственного интеллекта. Часть 2 / В.А. Коновалов // Вестник компьютерных и информационных технологий. — 2024. — Т. 21, № 11. — С. 54–63. — DOI: 10.14489/vkit.2024.11.pp.054-063.

5. Коновалов В.А. c𝒜net-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.

6. Голдблатт Р. Топосы. Категорный анализ логики / Р. Голдблатт; пер. с англ. В.Н. Гришина, В.В. Шокурова; под ред. Д.А. Бочвара. — Москва : Мир, 1983. — 488 с.

7. Непейвода Н.Н. Конструктивная математика: обзор достижений, недостатков и уроков. Часть III / Н.Н. Непейвода // Логические исследования / Logical Investigations. — 2014. — Т. 20. — С. 110–148.

8. Стандарт ISO/IEC 9899:2024. — URL: https://www.standards.ru/document/7627852.aspx (дата обращения: 06.02.2025).

9. Коновалов В.А. Способ синтеза топоса хешей в категорной cN-схеме алгоритма Маркова. Часть 2 / В.А. Коновалов // Вестник компьютерных и информационных технологий. — 2024. — Т. 21, № 9. — С. 40–51. — DOI: 10.14489/vkit.2024.09.pp.040-051.

10. Коновалов В.А. Способ синтеза топоса хешей в категорной cN-схеме алгоритма Маркова. Часть 1 / В.А. Коновалова // Вестник компьютерных и информационных технологий. — 2024. — Т. 21, № 8. — С. 32–42. — DOI: 10.14489/vkit.2024.08.pp.032-042.

Список литературы на английском языке / References in English

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

2. Kovalev S.P. Problema obosnovaniya v formal'nom 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 Tomsk State University. Philosophy, Sociology and Political Science]. — 2018. — № 46. — P. 22–29. — DOI: 10.17223/1998863X/46/3. [in Russian]

3. Konovalov V.A. Podkhod k sintezu algoritma, pererabatyvayushchego predpisaniya iskusstvennogo intellekta. Chast' 1 [Approach to synthesis of an algorithm processing artificial intelligence prescriptions. Part 1] / V.A. Konovalov // Vestnik komp'yuternykh i informatsionnykh tekhnologiy [Bulletin of Computer and Information Technologies]. — 2024. — Vol. 21, № 10. — P. 50–58. — DOI: 10.14489/vkit.2024.10.pp.050-058. [in Russian]

4. Konovalov V.A. Podkhod k sintezu algoritma, pererabatyvayushchego predpisaniya iskusstvennogo intellekta. Chast' 2 [Approach to synthesis of an algorithm processing artificial intelligence prescriptions. Part 2] / V.A. Konovalov // Vestnik komp'yuternykh i informatsionnykh tekhnologiy [Bulletin of Computer and Information Technologies]. — 2024. — Vol. 21, № 11. — P. 54–63. — DOI: 10.14489/vkit.2024.11.pp.054-063. [in Russian]

5. Konovalov V.A. c𝒜net-skhema algoritma Markova [c𝒜net-scheme of Markov's algorithm] / V.A. Konovalov // Vestnik komp'yuternykh 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]

6. Goldblatt R. Toposy. Kategornyy analiz logiki [Topoi: The Categorial Analysis of Logic] / R. Goldblatt; transl. from English by V.N. Grishin, V.V. Shokurov; ed. by D.A. Bochvar. — Moscow : Mir, 1983. — 488 p. [in Russian]

7. 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]

8. Standart ISO/IEC 9899:2024 [ISO/IEC 9899:2024 Standard]. — URL: https://www.standards.ru/document/7627852.aspx (accessed: 06.02.2025). [in Russian]

9. Konovalov V.A. Sposob sinteza toposa kheshey v kategornoy cN-skheme algoritma Markova. Chast' 2 [Method of synthesizing a topos of hashes in the categorical cN-scheme of Markov's algorithm. Part 2] / V.A. Konovalov // Vestnik komp'yuternykh i informatsionnykh tekhnologiy [Bulletin of Computer and Information Technologies]. — 2024. — Vol. 21, № 9. — P. 40–51. — DOI: 10.14489/vkit.2024.09.pp.040-051. [in Russian]

10. Konovalov V.A. Sposob sinteza toposa kheshey v kategornoy cN-skheme algoritma Markova. Chast' 1 [Method of synthesizing a topos of hashes in the categorical cN-scheme of Markov's algorithm. Part 1] / V.A. Konovalov // Vestnik komp'yuternykh i informatsionnykh tekhnologiy [Bulletin of Computer and Information Technologies]. — 2024. — Vol. 21, № 8. — P. 32–42. — DOI: 10.14489/vkit.2024.08.pp.032-042. [in Russian]