Расширение математической логики средствами ООП для достижения необходимой выразительности при разборе вопросов вычислений и вычислимости.

Д. Л. Гуринович
22 июня 2025 года

УДК 510.6+510.51+511.11

Если Вы имеете отношение к программированию и немного знакомы с математикой, то можете заметить, что никакой математической теории для программирования фактически нет. А если внимательней отнестись к этому вопросу, то нетрудно заметить, что у математиков нет даже понимания, как теоретически выразить представление чисел при помощи данных разного размера — что, вообще-то является обязательным требованием для решения вопросов о вычислениях и их скорости.

Более того, в очень показательной работе String theory (мы вернемся к ней в конце статьи) было сделано некоторое обобщение о том, что разное представление чисел при помощи строк (разного размера!) оказываются логически эквивалентными между собой и поэтому — говоря простыми словами — нет нужды разбираться в этом вопросе. (!) Достаточно, якобы, остановиться на арифметике Пеано или даже арифметике 2-го порядка (которая тоже, как они доказывают, не годится для выражения чисел при помощи строк разного размера — потому что они все равно получаются логически эквивалентны, то есть — логически неотличимы между собой) для работы с вопросами о вычислениях.

Корни этой недостаточной выразительности у математических инструментов уходят (как увидим) аж к Аристотелю и его современникам или почти современникам (таким, как Евклид) которые собрали математическую логику в целостную науку, стали её применять, но один «маленький нюанс» не почтили своим пониманием и оставили его на откуп «прикладникам», которые на практике этим «нюансом» для своих вычислений пользовались (иначе и вычислить ничего невозможно), но никогда его теоретически не осознавали так, чтобы сделать его частью математической логики.

Но теперь, когда вопросы вычислимости математики пытаются решить на теоретическом уровне — теперь без этого нюанса математикам-теоретикам обойтись невозможно (хотя они очень пытаются, как увидим, остаться в рамках прежних чрезмерно ограниченных в своей выразительности методов и остаются поэтому в тупике уже много десятилетий) и пора им его понять для теоретического понимания вычислений… К тому же понять его теперь просто, потому что практики его даже в некоторой степени формализовали (были до

известной степени вынуждены это сделать) для своих практических дел в программировании и назвали «Объектно-ориентированное программирование» – «ООП».

В чём же принципиальное упущение (недостаточная выразительность) арифметики Пеано (и любых её арифметических расширений) в вопросе вычислимости по сравнению с применяемыми практиками (теми же программистами) представлениями чисел посредством строк? В том, что число-строка не равна другому числу-строке не только в случае, если они соответствуют разному количеству операций увеличения на 1 от нуля, но и в том случае, если форма записи у них – разная. А именно:

Число 121 (десятичная запись), это:

1 × 10^2 + 2 × 10^1 + 1 × 10^0

А в двоичном виде это:

1 × 2^6 + 1 × 2^5 + 1 × 2^4 + 1 × 2^3 + 0 × 2^2 + 0 × 2^1 + 1 × 2^0

Если основание системы счисления указывать нижним индексом, используя для десятичной записи индекс «D», а для двоичной записи индекс «B», то для арифметики Пеано верно:

$121_D = 1111001_B$

Но эти же числа не могут быть равными, если рассматривать их как числа-строки:

$121_D \neq 1111001_B$

И для практиков это различие принципиально! А ведь ещё можно записывать числа в духе «стандартной интерпретации», когда десять «записано» при помощи «кучи» из одиннадцати счётных палочек, а ноль — это одна счётная палочка (нужно же его как-то обозначать). Стандартная интерпретация — это интерпретация «по умолчанию» в классическом для математической логики двухтомнике «Основания математики» Д. Гильберта и П. Бернайса. Такая интерпретация близка к «общепринятому» пониманию, что натуральные числа нужны для пересчёта объектов и десять объектов означает «кучу» из десяти пересчитанных объектов — что на 1 меньше «стандартной интерпретации». Индекс для стандартной интерпретации пусть будет «S». Тогда в рамках арифметики верно:

$121_D = 11\ldots 1_S$, где многоточие использовано для того, чтобы заменить собой 119 (десятичное сто девятнадцать) цифр 1 или «счётных палочек». Ведь их общее количество должно быть 122 (десятичное сто двадцать два), чтобы число, записанное в «стандартной интерпретации» данной строкой было таким же числом, которое соответствует строке «121» в десятичной записи.

Но число, записанное строкой единиц в «стандартная интерпретация» отличается от десятичной записи того же числа экспоненциально сильно своим размером! Что для вопросов вычислений — важнейшее и принципиальнейшее отличие, а для арифметики Пеано — просто ничто. Вот в чём недостаточная выразительность арифметики Пеано — и не только в этом.

Сразу скажу ответ, почему у математиков не получалось до сих пор построить теорию — неважно, первого или второго порядка — в которой отличие в размерах для разных представлений чисел имелось бы, а не «превращалось в ничто».

Допустим, у нас в теории T (какой-то) есть переменная с именем «a», значением которой является 121 в десятичной записи и переменная с именем «b», значением которой является то же самое число, но в двоичной записи:

$a = 121_D$; $b = 1111001_B$

Если мы изучаем вопросы вычислимости, то у нас есть способы перехода от одного способа записи чисел к другому и — вот тут самая «засада»! — есть способ менять, фактически, понимание того, как записано число на противоположное для всех «аналогичных» функций. А именно, если у нас есть функция, чтобы получать символ с i-го места строки, пусть s(), то верно:

$s(a, 1_D) = 1_D$; $s(a, 2_D) = 2_D$; $s(a, 3_D) = 1_D$;

$s(b, 1_B) = 1_B$; $s(b, 10_B) = 1_B$; … ; $s(b, 1001_B) = 0_B$; ...

Но благодаря функции «перекодирования» чисел мы легко получаем анти-функцию s1(), которая вместо десятичного представления «видит» двоичное и обратно. И мы получим картину:

$s1(b, 1_D) = 1_D$; $s1(b, 2_D) = 2_D$; $s1(b, 3_D) = 1_D$;

$s1(a, 1_B) = 1_B$; $s1(a, 10_B) = 1_B$; … ; $s1(a, 1001_B) = 0_B$; ...

И все остальные функции — размера, времени вычислений и т. д. могут получить свои анти-аналоги с чуть другими именами, в которых исходно десятично-записанное будет «толковаться» как двоично-записанное и наоборот. При этом никаких противоречий не возникнет — потому что это будут формально другие функции с другими именами.

И вот здесь мы видим «провал выразительности» математической логики в её «классическом» понимании: она никак не «привязывает» функции к тому, с чем они работают. С точки зрения классической математической логики у нас «просто функции» – «неизвестно о чём». И функция s1() ничем не хуже и не лучше «исходной» функции s(). Более того, мы можем построить теорию T1, где сможем аксиоматизировать именно функцию s1() и аксиоматизировать или определить вместе с ней остальные анти-аналоги «исходной» функции s() и остальных «исходных» функций теории T. И за счет функции преобразования между разными представлениями мы сможем определить все эти «исходные» функции. И, таким образом, мы видим, что исходная теория T полностью логически эквивалентна своему анти-аналогу теории T1.

Для не-математиков я, надеюсь, объяснил понятно, в чём тут проблема с неоднозначностью из-за логически эквивалентных теорий, которые возникают из-за наличия функций преобразований (взаимно-однозначных!) от одних представлений данных к другим представлениям данных. Дать строгое доказательство я не могу — потому что мои рассуждения касаются неопределенно-широкого круга теорий и речь о недостаточной выразительности классической математической логики — которую невозможно, конечно, формально доказать в рамках этой недостаточно выразительной классической логики.

Но для математиков, понимающих, что «обозначения функций» не появляются «просто так» сошлюсь на теорему об условиях, когда эти новые обозначения функций можно использовать так, как если бы они были аксиоматизированы:

Введение новых функциональных букв и предметных констант. Смотри, например, Э. Мендельсон, «Введение в математическую логику», Глава 2. Теории первого порядка. Раздел 9. Введение новых функциональных букв и предметных констант. Следствие 3.3.

Речь в данном следствии о том, что если доказано существование и единственность некоторого результата при списке заданых параметров, то можно использовать обозначение функции со списком данных параметров в качестве своих аргументов и результатом в качестве значения. Случай «перекодировки» от одних данных к другим (шифрование и дешифрование) — как раз такой. Поэтому с этой стороны — всё корректно.

Но раз математическая логика в её классическом понимании не «привязывает» функции к тому, с чем они работают, то кто их «привязывает»? Практики, конечно. А сейчас программисты догадались делать эту привязку явно, разработав средства ООП. Эти средства позволяют понять «привязку» функций к объектам и для физиков, работающих в рамках классической механики или релятивистской механики — хотя сами физики до ООП не додумались. А именно:

У нас есть объект — пусть это классическая материальная точка, которую мы обозначим именем «cP». И у неё есть характеристики — её координаты и время в выбранной нами инерциальной системе отсчета, скорость, масса, энергия, время на её собственных часах и т. д. И если записывать в синтаксисе ООП, то мы получим для всех этих «привязанных» характеристик такую картину:

cP.**R**; cP.T; cP.**V**; cP.m; cP.E; cP.t; ...

При этом между этими характеристиками есть соотношения, например:

cP.E = cP.m × (cP.**V**)^2 / 2; cP.t = cP.T

Аналогично, у нас есть релятивистская материальная точка — пусть имя этого объекта будет «rP» с «привязанными» к данному объекту характеристиками с теми же именами, что для классической материальной точки. Но соотношения между этими характеристиками будут уже другими.

И то, что физик не путает между собой соотношения для материальной точки — классической или релятивистской — говорит о том, что он привязывает — на самом деле — соответствующие характеристики к выбранному им объекту. То есть — действует в логике ООП, но не отдавая себе в этом «формального» отчета, не формализуя эту свою «привязку» явно, зримо и по согласованным с другими физиками стандартам.

Программистам же пришлось отдать себе явный и зримый отчёт в том, что и к каким объектам они «привязывают» — слишком много таких «объектов» в программе у среднего программиста, если сравнивать с количеством «объектов» в работе среднего физика. Поэтому программисты раньше всех не смогли «держать в голове» все эти «привязки» и сформулировали явные стандарты и логику подобных «привязок».

Наверняка в математике можно вместо использования синтаксиса ООП добавить нужную нам теперь дополнительную семантику иным синтаксисом или соглашениями. Например, можно условиться, что менять имена некоторых «прикреплённых к объектам» функций нельзя, даже если при таком переименовании всё «логически эквивалентно». Но это внесёт изменения в то, что по смыслу (семантике) оставалось неизменным уже более 2,3 тысяч лет — с момента смерти Аристотеля. Да, обозначения для значений буквами придумали позже,

но семантика «произвольных значений» была и тогда — смотри алгоритм Евклида, например, всё это отлично работало и работает. А вот семантика «привязки» функций к объектам не была осознана — по крайней мере до такой степени, чтобы стать частью коллективного разума математиков.

И проще теперь — на мой взгляд — воспользоваться наглядным и проверенным миллионами программистов и временем с момента изобретения ООП (Алан Кэй и Дэн Ингаллс, язык Smalltalk, в 1983 году была выпущены общедоступная реализация, известная как Smalltalk-80 Version 2) синтаксисом ООП, расширив им формальную логику и добавив в неё необходимую для работы с вычислениями и вычислимостью семантику таким способом.

Как синтаксис ООП решает проблему «логически эквивалентных» подходов для математики и позволит ей строить теории с необходимой для рассмотрения вопросов о вычислениях и вычислимости выразительностью?

Рассмотрим путь решения этой проблемы на примере размера строки — функции l().

В рамках классической математической логики допустим, что для значения переменной «a» верно: $l(a) = 3_D$, но при этом $l_1(a) = 111_B$,. А, с другой стороны, для значения переменной «b» верно: $l(b) = 111_B$, но при этом $l_1(b) = 3_D$.

И размер какой строки — которая в переменной «a» или которая в переменной «b» - больше размера строки в переменой-визави? Мы не можем сказать, формально говоря — потому что оба подхода логически эквивалентны. С точностью до переименования l() на $l_1()$ и обратно.

Противоречий нет между этими эквивалентными подходами — функции разные, но и имена у них разные. Но если мы внедряем ООП, то у нас получиться такой расклад:

$a.l() = 3_D$, и $b.l() = 111_B$

При этом альтернативы длине строки, задаваемой «свойством» s.l() — нет. У нас аксиоматизировано имя свойства длины строки и это имя «l», а альтернативного свойства с именем «l_1» нет. И поэтому значения, которое могло бы выдать свойство с именем «l_1», у нас нет, потому что нет свойства с таким именем, поэтому альтернатива — если бы она была — была бы значением свойства с прежним именем «l» и было бы (если делать по аналогии с обычными функциями выше при классической математической логике):

$a.l() = 111_B$, и $b.l() = 3_D$

Но тут получается противоречие с предыдущей парой равенств. А значит, альтернатива, которая «могла бы быть» — быть не может.

Поэтому для разбора вопросов вычислений и вычислимости мы должны преодолеть проблему «логически эквивалентных» наборов функций, но для этого у нас есть решение — и это решение — синтаксис и методы ООП. Поэтому синтаксис ООП (или его аналог, но зачем искать аналог проверенной на практике методике?) должен быть добавлен в формальную логику для того, чтобы она предоставляла необходимые инструменты для построения теорий, достаточно выразительных для работы с вычислениями и вычислимостью.

Разберем коротко не только научный, но и психологический тупик, в котором находится современное математическое сообщество в вопросах — крайне востребованных практически и с огромным влиянием на экономику и разум общества — вычислений и вычислимости. На

примере статьи String Theory. John Corcoran; William Frank; Michael Maloney. The Journal of Symbolic Logic, Vol. 39, No. 4 (Dec., 1974), 625-637 (гиперссылка: https://philarchive.org/rec/CORST?all_versions=1).

Самое занятное, что мы видим 4 несовместимых между собой с логической точки зрения стороны в целях, результатах и в толковании данной работы:

1. Сами математики пытаются доказать недостаточную выразительность своих усилий для вопросов вычислений, когда пытаются рассмотреть разные способы представления чисел — при помощи не просто разных строк, но даже разных сколь угодно сильно по своим размерам строк. Ведь эти попытки дают (на их взгляд) логически эквивалентные результаты.

2. Но неудача таких попыток неприемлема с точки зрения задач теории алгоритмов, потому что не могут быть теоретически неотличимыми (логически эквивалентными или «синонимичными», как пишут в данной статье) объекты, которые обязательно должны отличаться между собой из-за сколь угодно сильных отличий в принципиальных для вопросов вычислений характеристиках — своих размерах и, следовательно, времени на свою обработку в процессе вычислений.

3. Математикам, участвовавшем в этих доказательствах логической эквивалентности (чьи усилия обобщает статья String theory) удаётся построить такое доказательство о логической неотличимости представлений чисел строками очень разного размера, которое они и их коллеги-математики считают корректным (хотя там есть принципиальные ошибки в самом построении доказательства — вроде равенства нулей, когда нули записаны в разных представлениях строками и за несколько раз увеличения на 1 приведут такие «равные» нули к явно разным строкам, но речь об ошибке в толковании результата доказательства, считающегося корректным). И никто не видит, что тем самым они все соглашаются с неудачей в построении теории, где должны различаться между собой те характеристики (размер данных, например, при разных способах записи числа), которые принципиально должны отличаться между собой для решения вопросов вычислений.

4. И на основании того, что доказана — на их взгляд — неудача в достижении необходимой для вопросов вычислений (для теории алгоритмов в том числе) выразительности, делается вывод (с которым согласны все математики), что в качестве теории для решения вопросов вычислений оставляем арифметику, недостаточная выразительность которой (даже в рамках логики второго порядка) как раз и была — на их взгляд — доказана.

Получается, что мы имеем ещё одну «муху Аристотеля» с, якобы, 8 ногами? Карлу Линею удалось убедить биологов в 18 веке, что у мухи 6 ног, а удастся ли кому-то в 21 веке убедить математиков в очевидно недостаточной выразительности логики в её нынешнем виде для решения вопросов вычислений, включая вопрос вычислимости?

Средства и синтаксис ООП ждут, чтобы их использовали — притом это совсем не трудно и уже во многом проработано программистским сообществом. Но теперь надо использовать данное достижение программистского сообщества и в математической логике для достижения необходимой выразительности при решении крайне востребованных обществом и принципиальных для самой математики вопросов вычислений и вычислимости.

Это будет самый первый и принципиальный шаг, а строить теории для вычислений и вычислимости на базе логики с ООП-расширением потребует рутинных усилий, но не преодоления принципиальных тупиков понимания.