Логические исследования
2019. Т. 25. № 2. С. 61–74
УДК 168.3

Logical Investigations
2019, Vol. 25, No. 2, pp. 61–74
DOI: 10.21146/2074-1472-2019-25-2-61-74

Г. Джапаридзе

Арифметика на основе логики вычислимости*

Георгий Джапаридзе
Villanova University.
800 Lancaster Avenue, Villanova, PA 19085, USA.
Институт философии РАН.
Российская Федерация, 109240, г. Москва, ул. Гончарная, д.12, стр.1.
E-mail: giorgi.japaridze@villanova.edu

Аннотация: Статья представляет собой краткий обзор теорий чисел, основанных на *логике вычислимости (CoL, computability logic)*, имеющей игровую семантику в терминах вычислительных задач и ресурсов. Такие теории, называемые *кларифметиками (clarithmetics)*, являются консервативными расширениями арифметики Пеано первого порядка.

В первом разделе статьи излагаются концептуальные основания *CoL* и описывается соответствующий фрагмент ее формального языка с так называемыми параллельными связками, связками и кванторами выбора (choice), а также слепыми (blind) кванторами. Как синтаксически, так и семантически это консервативное обобщение языка классической логики.

Кларифметика, основанная на соответствующем фрагменте *CoL* в том же смысле, что и арифметика Пеано основана на классической логике, рассматривается во втором разделе. В нем представлены аксиомы и правила вывода системы кларифметики, названной **CLA11**, а также изложены основные результаты, относящиеся к свойствам этой системы: конструктивная корректность, экстенсиональная и интенсиональная полнота.

В заключительном разделе рассматриваются два потенциальных приложения кларифметики: кларифметика как язык декларативного программирования и как инструмент для разделения классов вычислительной сложности. Если кларифметику или подобные теории на основе *CoL* рассматривать как языки программирования, оно сводится к поиску доказательств, поскольку программы могут быть механически извлечены из доказательств; такие программы также служат своей собственной верификации, тем самым полностью нейтрализуя пресловутую (и неразрешимую в общем случае) проблему верификации. Второе приложение сводит проблему разделения различных классов вычислительной сложности к разделению соответствующих версий кларифметики, потенциальная польза от этого вытекает из убеждения, что разделение теорий в целом должно быть проще, чем прямое разделение классов сложности.

* Статья по материалам выступления 14 ноября 2018 на онлайн-семинаре сектора логики Института философии РАН (см. видео https://youtu.be/w9-1Xm8MMmQ).
Перевод с английского В.И. Шалака.

Ключевые слова: логика вычислимости, арифметика Пеано, игровая семантика, конструктивная логика, интуиционистская логика, линейная логика, интерактивная вычислимость

Для цитирования: *Джапаридзе Г.* Арифметика на основе логики вычислимости // Логические исследования / Logical Investigations. 2019. Т. 25. № 2. С. 61–74. DOI: 10.21146/2074-1472-2019-25-2-61-74

1. Логика вычислимости (*CoL*): формальная теория вычислимости в том же смысле, в котором классическая логика является формальной теорией истинности

Статья посвящена теориям чисел, основанным на *логике вычислимости (computability logic)*. В качестве сокращения для логики вычислимости я буду использовать аббревиатуру *CoL*. Начнем с ответа на вопрос, что такое логика вычислимости? Это предложенный мной некоторое время назад подход, который я характеризую как формальную теорию вычислимости в том же смысле, в котором классическая логика является формальной теорией истинности. Сравним эти две логики, чтобы понять, что имеется в виду.

В классической логике центральным семантическим понятием является *истинность*, формулы представляют *утверждения*, и основная польза от классической логики заключается в том, что она дает ответы на два вопроса: 1) Верно ли, что P (всегда) *истинно*? 2) Верно ли, что *истинность* P (всегда) следует из *истинности* Q?

Итак, что такое логика вычислимости и в чем заключается ее отличие от классической? Все одинаково за исключением того, что вместо понятия истинности используется вычислимость. Поэтому ее центральным семантическим понятием является *вычислимость*. Формулы представляют не утверждения, как раньше, а *вычислительные задачи*, поскольку вычислимость – это свойство вычислительных задач. Логика вычислимости дает систематический ответ на вопросы: 1) Верно ли, что P (всегда) вычислимо? 2) Верно ли, что вычислимость P (всегда) следует из вычислимости Q? Более того, она дает ответ на эти вопросы в конструктивном смысле. А именно, если в ней можно установить, что P вычислима, то это не просто утверждение о существовании вычисления (алгоритма) для P, а утверждение о том, как именно вычислить P. Аналогичным образом она говорит нам, как построить алгоритм для P по алгоритму для Q.

Естественным образом получается, что классические утверждения (предикаты, высказывания) являются специальными случаями вычислительных задач, и классическая истинность является специальным, простейшим случаем вычислимости. Это, в конечном итоге, делает классическую

логику консервативным фрагментом логики вычислимости. Консервативным фрагментом в том смысле, что язык CoL более выразителен, чем язык классической логики, и содержит последнюю просто как фрагмент, но если мы ограничим его до языка классической логики, то семантика CoL порождает тот же класс значимых формул, что и классическая логика.

Прежде всего необходимо договориться о нашем понимании того, что такое *вычислительные задачи*. Если обратиться к Чёрчу, вычислительные задачи – это просто функции, которые требуется вычислить. Но для нас понимание вычислительных задач носит более общий характер. А именно, вычислительная задача – это игра с нулевой суммой между машиной, обозначенной символом $\top$, и ее окружением/средой, обозначенным символом $\bot$. Функции – это просто частные случаи таких игр, а здесь мы имеем дело с вычислительными задачами произвольных степеней интерактивности (вычисление функции недостаточно интерактивно, поскольку состоит всего из двух шагов: получения входа и порождения выхода).

Я не буду уточнять и определять, что именно подразумевается под «машиной», но давайте понимать ее просто как алгоритм. Таким образом, вместо машины всегда можно думать об интерактивном алгоритме, о следовании некоторой механической процедуре. Будем говорить, что машина M *выигрывает* (*вычисляет*, *решает*) игру/задачу G, е. и т. е. M выигрывает G независимо от того, в какой среде она действует. О такой машине M будем говорить, что она является *алгоритмическим решением* (алгоритмической *выигрышной стратегией*) для G. Задача/игра *вычислима*, е. и т. е. она имеет алгоритмическое решение.

Логические операторы CoL (связки, кванторы) представляют операции над играми. Имеется целый «зоопарк» таких операторов, но здесь мы рассмотрим лишь небольшой фрагмент, состоящий из

$$\neg, \wedge, \vee, \rightarrow, \forall, \exists, \sqcap, \sqcup, \bigsqcap, \bigsqcup.$$

Здесь мы видим все операторы классической логики и кое-что в дополнение к ним. Эти операторы, как и все операторы CoL, представляют в языке операции над играми. Операторы обычного классического начертания являются консервативными обобщениями своих классических аналогов в том смысле, что они автоматически сохраняют классический смысл применительно к утверждениям классической логики.

Сеанс (run) игры – это последовательность ходов, каждый из которых имеет префикс $\top$ или $\bot$ для указания того, кто из игроков сделал ход. Я не даю никаких формальных определений, но, конечно, они существуют.

Атомарные предложения, такие как $2+2=4$, являются играми без ходов, то есть играми, чей единственный допустимый сеанс – пустой сеанс $\langle\rangle$.

Такая игра выиграна машиной, если предложение истинно в классическом смысле, и проиграна, если ложно. То же самое просматривается для всех формул, построенных из атомов и операторов $\neg, \wedge, \vee, \rightarrow, \forall, \exists$, как только мы их определим.

Таким образом, (пустой сеанс) $2 + 2 = 4$ или $\forall x(x = x)$ выигрывает машина, а $2 + 2 = 5$ или $0 = 1 \wedge 2 = 2$ проигрывает. Это означает, что задачи/игры $2+2 = 4$ и $\forall x(x = x)$ вычислимы (на машине, которая ничего не делает), в то время как $2 + 2 = 5$ и $0 = 1 \wedge 2 = 2$ невычислимы.

Главное, что следует запомнить – формулы классической логики представляют собой игры без ходов. Кто-то может растеряться и спросить: «Как мы можем назвать что-то игрой, если в ней нет ходов ?!» Что ж, позвольте мне напомнить вам ситуацию с понятием нуля. У римлян не было этого понятия, и последующая европейская традиция долгое время сопротивлялась идее принятия нуля в качестве законного числа. Считалось, что число должно представлять количество, а ноль означает отсутствие количества вообще, так что это не значащее число. Но теперь мы знаем, что мы не сможем достичь много в математике без нуля. Аналогичным образом, вышеперечисленные игры без ходов имеют смысл в рамках подхода *CoL*.

Давайте теперь посмотрим на операции над играми, с которыми мы будем иметь дело.

Первой рассмотрим *конъюнкцию выбора* двух игр A и B, которая записывается в виде $A \sqcap B$. В этой игре первый ход делает (только) среда, которая должна выбрать один из двух допустимых ходов «*left*» или «*right*». После этого игра продолжается по правилам выбранного конъюнкта A или B. Этот выбор является не только привилегией окружающей среды, но и ее обязанностью, потому что если среда в данном случае не сможет сделать ход (выбор), то она проиграет, и машина, соответственно, будет считаться победителем.

Дизъюнкция выбора $A \sqcup B$ отличается лишь тем, что здесь первый ход/выбор делает (только) машина и проиграет, если не сделает его. Таким образом, роли машины и окружающей среды здесь меняются.

Универсальная квантификация выбора игры $A(x)$ записывается в виде $\sqcap x A(x)$ и, по сути, является «большой конъюнкцией выбора» в том смысле, что это игра, в которой первый допустимый ход, как и в случае конъюнкции выбора, делает среда, которая, однако, вместо выбора между *left* и *right* просто выбирает некоторое натуральное число n, после чего игра продолжается как $A(n)$. Если такой шаг не будет сделан, среда проиграет.

Экзистенциальная квантификация выбора $\sqcup x A(x)$ отличается лишь тем, что в ней первый ход должна сделать машина, которая проиграет, если не сделает этого хода.

Приведем пример. $\langle \bot 6, \top left\rangle$ – сеанс игры $\sqcap x(Even(x) \sqcup Odd(x))$, выигранной машиной. Следующая последовательность показывает, как при этом «изменяется» игра:

$$\sqcap x(Even(x) \sqcup Odd(x)) \Rightarrow Even(6) \sqcup Odd(6) \Rightarrow Even(6).$$

В соответствии с этим сценарием среда выбрала 6 для x, после чего игра продолжилась как $Even(6) \sqcup Odd(6)$. В ответ машина сделала ход *left*, который означает, что был выбран левый дизъюнкт, и игра продолжилась как $Even(6)$. Но на этом она и завершилась, поскольку никаких дальнейших ходов не было сделано (не могли быть сделаны). Машина выиграла, так как $Even(6)$ является истинным высказыванием, а истинные высказывания автоматически означают победу машины.

Отрицание $\neg$ читается как «*не*» и является операцией перемены ролей игроков: ходы и выигрыши машины становятся ходами и выигрышами окружающей среды, и наоборот. Более строго: для сеанса Φ пусть *негативный образ* Φ означает результат замены всех меток в Φ на противоположные. Например, отрицательным образом $\langle \bot 6, \top left\rangle$ будет $\langle \top 6, \bot left\rangle$. Тогда для игры G ее отрицание $\neg G$ будет игрой, сеансы которой будут отрицательными образами сеансов G, окружающая среда и машина поменяются местами, где данный игрок выигрывает сеанс, е. и т. е. другой игрок выигрывает в G негативный образ этого сеанса.

Пример: у нас есть $\neg\sqcap x(Even(x) \sqcup Odd(x)) = \sqcup x(Odd(x) \sqcap Even(x))$. Рассмотрим отрицание $\sqcap x(Even(x) \sqcup Odd(x))$. Без отрицания это универсально квантифицированная игра, в которой среда делает первый ход. Но отрицание меняет роли игроков, и теперь именно машина должна сделать первый ход. Вот почему $\sqcap x$ становится $\sqcup x$. По тем же причинам $\sqcup$ становится $\sqcap$, *Even* становится *Odd* ($= \neg Even$), а *Odd* становится *Even*.

Мы видим, что законы де Моргана остаются в силе для операторов выбора. На самом деле *CoL* имеет несколько (4+) видов конъюнкций, дизъюнкций и кванторов, и законы де Моргана выполняться для всех из них.

В применении к атомарному высказыванию S (или любой другой игре без ходов) $\neg$ ведет себя точно так же, как классическое отрицание, потому что единственный допустимый сеанс в играх, представленных атомарными высказываниями, – это пустой сеанс $\langle\rangle$ и отрицательный образ $\langle\rangle$ – то же самое, что $\langle\rangle$. Таким образом, только победители меняются местами, и машина выигрывает «сеанс» S (иными словами, S истинно), е. и т. е. среда выигрывает его в игре $\neg S$ (то есть S является ложным).

То же самое справедливо для всех операторов, которые выглядят как операторы классической логики: $\wedge$, $\vee$, $\rightarrow$, $\forall$, $\exists$. Применительно к играм без ходов поведение таких операторов является в точности классическим. Их

новые значения совпадают с классическими, и это происходит естественно/автоматически, а не благодаря постулатам, построенным «вручную».

Независимо от того, является S игрой без ходов или нет, имеет место $\neg\neg S = S$. Закон двойного отрицания остается в силе, потому что двойная перемена ролей возвращает каждого игрока к первоначальной роли.

Параллельная конъюнкция игр A и B, которая записывается в виде $A \wedge B$, представляет игру, в которой A и B играются одновременно (никакой выбор между ними не делается), и чтобы выиграть, машина должна выиграть в обоих конъюнктах. Чтобы указать, в каком конъюнкте делается данный ход, перед ним должен стоять префикс «*left*» или «*right*».

Параллельная дизъюнкция $A \vee B$ отличается лишь тем, что в ней достаточно выиграть хотя бы в одном из дизъюнктов.

Пример: $\langle \bot right.6,\ \top left.6,\ \bot left.left,\ \top right.left \rangle$ – сеанс игры

$$\sqcup x(Odd(x) \sqcap Even(x)) \vee \sqcap x(Even(x) \sqcup Odd(x)).$$

Смысл первого хода $\bot right.6$ заключается в том, что среда выбирает *6* для x в правом $\vee$-дизъюнкте. В результате игра превращается в

$$\sqcup x(Odd(x) \sqcap Even(x)) \vee (Even(6) \sqcup Odd(6)).$$

Ход $\top left.6$, сделанный в ответ машиной, означает выбор того же числа *6* для x в левом $\vee$-дизъюнкте, и игра продолжается как

$$(Odd(6) \sqcap Even(6)) \vee (Even(6) \sqcup Odd(6)).$$

Следующий ход $\bot left.left$, сделан средой, и его смысл заключается в выборе левого $\sqcap$-конъюнкта в левом $\vee$-дизъюнкте, что упрощает игру до $Odd(6) \vee (Even(6) \sqcup Odd(6))$. Последний ход $\top right.left$, сделанный машиной, превращает игру в $Odd(6) \vee Even(6))$. Машина ее выигрывает, потому что $Even(6)$ истинно и, следовательно, она выигрывает в одном из двух $\vee$ -дизъюнктов.

Параллельная импликация $\rightarrow$ понимается как стандартное сокращение: $A \rightarrow B =_{df} \neg A \vee B$. Из-за префикса $\neg$ роли игроков меняются в A, превращая его в *вычислительный ресурс*, которым может использоваться $\top$ при решении задачи B. Вычислительная интуиция, связанная с этой комбинацией задач, наиболее интересна, поскольку интуитивно $A \rightarrow B$ можно рассматривать, как задачу *редукции/сведения* B к A. Например, рассмотрим следующие два предиката:

- $Halts(x, y)$ = «Машина Тьюринга x останавливается для входных данных y»;

- $Accepts(x, y)$ = «Машина Тьюринга x допускает входные данные y», т. е. результатом вычисления является *1*.

Известная *проблема остановки* может быть записана следующим образом: $\sqcap x \sqcap y(Halts(x, y) \sqcup \neg Halts(x, y))$. В этой игре среда выбирает некоторые конкретные значения для x и y, и машина должна ответить, останавливается x для y ($\top left$) или нет ($\top right$). Аналогично для *проблемы допущения* $\sqcap x \sqcap y(Accepts(x, y) \sqcup \neg Accepts(x, y))$. Обе проблемы, как известно, неразрешимы, но можно показать, что параллельная импликация от первой ко второй вычислима.

$$\overbrace{\overbrace{\sqcap x \sqcap y(Halts(x, y) \sqcup \neg Halts(x, y))}^{\text{Проблема остановки}} \rightarrow \overbrace{\sqcap x \sqcap y(Accepts(x, y) \sqcup \neg Accepts(x, y))}^{\text{Проблема допущения}}}^{\text{Редукция проблемы допущения к проблеме остановки}}$$

Выигрышная стратегия основана на редукции консеквента к антецеденту. Грубо говоря, это делается следующим образом. В то время как у обоих игроков есть начальные законные ходы в этой игре, машина ждет, пока среда не выберет некоторые значения m и n для x и y в консеквенте (в противном случае среда проигрывает) и, таким образом, трансформирует игру в

$$\sqcap x \sqcap y(Halts(x, y) \sqcup \neg Halts(x, y)) \rightarrow Accepts(m, n) \sqcup \neg Accepts(m, n).$$

Затем машина выбирает те же m и n в антецеденте и игра продолжается как

$$Halts(m, n) \sqcup \neg Halts(m, n) \rightarrow Accepts(m, n) \sqcup \neg Accepts(m, n).$$

Теперь опять у обоих игроков есть законные ходы: $\top$ в консеквенте и $\bot$ в антецеденте, но для машины разумно подождать, чтобы среда пошла первой. Если среда не может сделать ход, она проигрывает в антецеденте, и, таким образом, машина выигрывает всю игру. Поэтому предположим, что среда делает ход в антецеденте, сообщая, останавливается ли m для n или нет. Мы можем предположить, что все сообщаемое окружающей средой является истинным, иначе она проигрывает. Если она говорит, что m не останавливается для n, то машина выбирает $\neg Accepts(m, n)$ в консеквенте и выигрывает, потому что если m не останавливается, то она не допускает n. В противном случае, если среда сообщает, что m останавливается для n, машина имитирует работу m для входа n, и по результату вычисления определяет, допускает ли m и n или нет. Затем она соответственно выбирает между $Accepts(m, n)$ и $\neg Accepts(m, n)$ в консеквенте и выигрывает.

Слепыми кванторами мы называем $\forall$ и $\exists$, где $\exists$ является дуалом де Моргана для $\forall$ ($\exists = \neg\forall\neg$), поэтому рассмотрим только слепую универсальную квантификацию $\forall x A(x)$.

В отличие от $\sqcap$, с $\forall$ не связаны никакие ходы, т.е. значение для x не указывается (ни одним из игроков), и, чтобы выиграть, машина должна играть «вслепую» так, чтобы гарантировать успех в $A(x)$ для каждого возможного значения x.

В качестве примера рассмотрим игру

$$\forall x(Even(x) \sqcup Odd(x) \to \sqcap y(Even(x+y) \sqcup Odd(x+y))).$$

Следующий сценарий разворачивается в соответствии с сеансом $\langle \bot right.5,\ \bot left.right,\ \top right.left\rangle$: Среда выбирает значение 5 для y в консеквенте, приводя игру к виду

$$\forall x(Even(x) \sqcup Odd(x) \to Even(x+5) \sqcup Odd(x+5)).$$

Машина должна сказать, является ли $x+5$ четным или нечетным. Но она не знает значения x. Может ли она все же определить четность $x+5$? Да, может, если машина знает, является ли x четным или нечетным, но не зная, каким именно числом. И эту информация о четности x можно получить из антецедента, в котором среда обязана совершить ход. Сделав ход $\bot left.right$ и сведя игру к виду $\forall x(Odd(x) \to Even(x+5) \sqcup Odd(x+5))$, среда сообщает, что нечетно. Но тогда $x+y$ является четным, поэтому последний ход $\top right.left$ приводит к $\forall x(Odd(x) \to Even(x+5))$ и делает машину победителем. Обратите внимание на то, как $\forall$ и $\to$ сохранялись в приведенной выше последовательности игр. Как правило, классическая структура игры, когда она развивается подобным образом, остается неизменной, все изменения происходят исключительно в компонентах выбора ($\sqcap, \sqcup, \bigsqcap, \bigsqcup$).

2. Кларифметики: формальные теории чисел, базирующиеся на CoL

В *CoL* стандартные понятия временной (TIME) и пространственной (SPACE) сложности вычислений естественным образом консервативно обобщаются на интерактивные вычисления. Также вводится новая мера *амплитудной сложности*, которая связана с размерами ходов $\top$ относительно размеров ходов $\bot$.

Вместо вычислимости в принципе мы можем теперь так же осмысленно говорить о вычислимости с ограниченными ресурсами (амплитуда, пространство, время). Например, о вычислимости в полиномиальном пространстве, а не просто вычислимости.

Пример: каждая из задач в левом столбце ниже вычисляется за полиномиальное время. Правая колонка говорит нам, что это означает в стандартных терминах.

$\sqcap x \sqcap y(x = y \sqcup x \neq y)$	"=" разрешимо за полиномиальное время
$\sqcap x \sqcap y \sqcup z(z = x + y)$	"+" вычислима за полиномиальное время
$\sqcap x(\exists y(x = y \times y) \to \sqcup y(x = y \times y))$	если существует «квадратный корень», то он вычислим за полиномиальное время
$\sqcap x \sqcup y(p(x) \leftrightarrow q(y))$ ($A \leftrightarrow B$ сокращение для $(A \to B) \wedge (B \to A)$)	p за полиномиальное время редуцируется к q

Кларифметики – это формальные теории чисел, основанные на *CoL* в том же смысле, что и **PA** (арифметика Пеано) основана на классической логике. Они обычно строятся языке, логический словарь которого мы рассмотрели в предыдущем разделе: $\neg, \wedge, \vee, \to, \forall, \exists, \sqcap, \sqcup, \bigsqcap, \bigsqcup$. Нелогический словарь является стандартным: $0, ', +, \times, =$ (x' означает $x+1$). Чтобы продолжить, нам нужны некоторые технические соглашения.

- Под *ограничительным термом* (bound) мы подразумеваем *pterm* (терм или псевдотерм) для некоторой монотонной функции (термин был придуман Джорджем Булосом). Строго говоря, псевдотермы не совсем термы, но можно считать, что в языке у нас есть для них специальные термы. Это не приведет ни к каким изменениям, потому что каждая формула с псевдотермами может быть эквивалентно переписана как формула с одними лишь настоящими термами. Терминологически мы отождествляем pterm'ы с функциями, которые они представляют.
- *Класс ограничительных термов* – это некоторое множество ограничительных термов, замкнутых относительно переименования переменных.
- *3-сложность* – это тройка $\mathbf{R} = (\mathbf{R}_{amplitude}, \mathbf{R}_{space}, \mathbf{R}_{time})$ классов ограниченных термов, удовлетворяющих некоторым условиям *регулярности*. Я не буду вводить здесь эти условия, но они естественны. Достаточно сказать, что все естественно возникающие 3-сложности регулярны, если $\mathbf{R}_{ampitude}$ по крайней мере линеен (содержит все линейные функции), $\mathbf{R}_{space}$ логарифмичен и $\mathbf{R}_{time}$ полиномиален.

- *3-сложным решением* $\mathbf{R}$ задачи G является решение для G, амплитудная сложность которого ограничена сверху некоторой функцией из $\mathbf{R}_{amplitude}$, пространственная сложность ограничена сверху некоторой функцией из $\mathbf{R}_{space}$ и временная сложность ограничена сверху некоторой функцией из $\mathbf{R}_{time}$. Другими словами, это выигрышная стратегия/алгоритм $\top$, время выполнения которой задается $\mathbf{R}_{time}$, пространство выполнения задается $\mathbf{R}_{space}$ а амплитуда – $\mathbf{R}_{amplitude}$.
- $|x|$ – стандартная функция «длина двоичного представления x», т.е. $\lceil log_2(x+1) \rceil$.
- $Bit(x,y)$ – стандартная функция для предиката «y-й (справа, начиная счет с нуля) бит бинарного представления x равен 1», т.е. $\lfloor x/2^y \rfloor \ mod\ 2 = 1$.
- $\underline{s}$ (соотв. $|\underline{s}|$) обозначает n-ку $(s_1, \ldots, s_n)$ (соотв. $(|s_1|, \ldots, |s_n|)$), где $s_1, \ldots, s_n$ – переменные.
- Пусть B – класс ограничительных термов. Будем говорить, что формула F *B-ограничена*, е. и т. е. каждая $\sqcap$-подформула F имеет вид $\sqcap x(|x| \leq b(|\underline{s}|) \to F)$ и каждая $\sqcup$-подформула имеет вид $\sqcup x(|x| \leq b(|\underline{s}|) \wedge F)$, где x, $\underline{s}$ попарно различные переменные, не связанные кванторами $\forall$, $\exists$ в F, и b ограничительный терм из B.

Был разработан ряд систем кларифметики, соответствующих различным классам сложности. В этой статье мы рассмотрим только одну такую систему, названную **CLA11**. Это скорее не конкретная теория, а схема кларифметических теорий, порождающая теории $\mathbf{CLA11^R}$ для каждой 3-сложности $\mathbf{R}$.

Фиксируем некоторую 3-сложность $\mathbf{R} = (\mathbf{R}_{amplitude}, \mathbf{R}_{space}, \mathbf{R}_{time})$, чтобы определить соответствующую теорию $\mathbf{CLA11^R}$. Логическим базисом (аксиомы и правила) такой теории является некоторая корректная и полная в очень сильном смысле аксиоматизация **CL12** логики *CoL*. Это логический базис $\mathbf{CLA11^R}$ в том же смысле, что и классическая логика – логический базис арифметики Пеано. Здесь мы приведем только нелогические постулаты $\mathbf{CLA11^R}$.

Нелогическими *аксиомами* $\mathbf{CLA11^R}$ являются:

1. $\forall x \neg(0 = x')$
2. $\forall x \forall y(x' = y' \to x = y)$
3. $\forall x(x + 0 = x)$
4. $\forall x \forall y(x + y' = (x + y)')$
5. $\forall x(x \times 0 = 0)$
6. $\forall x \forall y(x \times y' = (x \times y) + x)$
7. $\forall$-замыкание $F(0) \wedge \forall x(F(x) \to F(x')) \to \forall x F(x)$ для каждой не содержащей $\sqcap, \sqcup, \bigsqcap, \bigsqcup$ формулы F

8. $\sqcap x \sqcup y(y = x')$
9. $\sqcap x \sqcup y(y = |x|)$
10. $\sqcap x \sqcap y(Bit(x, y) \sqcup \neg Bit(x, y))$

Формулы (1)–(7) — это не что иное, как аксиомы стандартной арифметики Пеано **PA**. Конечно, (7) является не аксиомой, а схемой аксиом, порождающей бесконечно много частных аксиом. Кроме того, у нас есть три дополнительные аксиомы (8)–(10). Первая выражает вычислимость функции «следования за». Вторая выражает вычислимость логарифмической функции. Третья говорит, что вопрос о том, равен ли единице y-й бит (двоичного представления) x, разрешим.

Наряду с $\mathbf{CLA11^R}$ мы также рассматриваем $\mathbf{CLA11^R} + \mathbf{TA}$, где **TA** служит обозначением «Арифметики истины». $\mathbf{CLA11^R} + \mathbf{TA}$ отличается от $\mathbf{CLA11^R}$ тем, что ее аксиомы включают в качестве дополнительных аксиом все истинные предложения классической арифметики (а не только аксиомы Пеано). Конечно, эта система, в отличие от $\mathbf{CLA11^R}$, больше не является рекурсивно аксиоматизируемой.

К аксиомам $\mathbf{CLA11^R}$ добавляются два следующих нелогических *правила* вывода:

- *Индукция* $\dfrac{F(0), \qquad \sqcap x(F(x) \to F(x'))}{\sqcap x \leq b(|\underline{s}|)F(x)}$

  где x и $\underline{s}$ – попарно различные переменные, $F(x)$ является $\mathbf{R}_{space}$-ограниченной формулой, и b ограничительный терм из $\mathbf{R}_{time}$;

- *Выделение* $\dfrac{\sqcap y(p(y) \sqcup \neg p(y))}{\sqcup |x| \leq b(|\underline{s}|) \forall y < |x|(Bit(y, x) \leftrightarrow p(x)))}$

  где x, y и $\underline{s}$ – попарно различные переменные, $p(y)$ свободная от $\sqcap, \sqcup, \sqcap, \sqcup$ формула, не содержащая x, а b ограничительный терм из $\mathbf{R}_{amplitude}$.

Как видим, в приведенном выше правиле индукции, в отличие от стандартного аналога, вместо $\forall$ используется $\sqcap$. Интуиция здесь заключается в том, что, если мы знаем, как вычислить $F(0)$, и также для всех $x \leq b(|s|)$ знаем, как редуцировать $F(x')$ к $F(x)$, то мы знаем, как вычислить $F(x)$. Ограничение заключается в том, что $F(x)$ должна быть формулой, ограниченной $\mathbf{R}_{space}$, и ограничительный терм b должен быть взят из $\mathbf{R}_{time}$.

Интуиция, связанная с правилом выделения, заключается в том, что, если мы можем разрешить предикат p, то мы можем сгенерировать (следовательно, можем *выбрать*) такое число x, что n-й бит x равен 1, если и только если p истинно для n. Таким образом, разрешимый предикат p генерирует свое собственное число в том смысле, что он сообщает нам каждый бит двоичного представления x. Опять же, здесь есть ограничение, согласно которому b должен быть взят из $\mathbf{R}_{amplitude}$.

Теперь мы готовы сформулировать основной результат, приняв пару дополнительных терминологических соглашений. Под *арифметической задачей* мы понимаем задачу/игру, которая выражена некоторой формулой языка $\mathbf{CLA11^R}$. Мы говорим, что такая задача *представима* в $\mathbf{CLA11^R}$, если она выражается некоторой теоремой $\mathbf{CLA11^R}$. Обратите внимание, что одна и та же арифметическая задача может быть выражена множеством различных формул, и для того, чтобы задача квалифицировалась как представимая, достаточно, чтобы только одна из таких формул была доказуема.

Теорема 1. *Для любой 3-сложности* $\mathbf{R}$ *имеет место следующее:*

- ***Конструктивная корректность****: Для каждой теоремы* F *из* $\mathbf{CLA11^R + TA}$ *задача, выраженная* F*, имеет решение 3-сложности* $\mathbf{R}$*, и это решение может быть автоматически извлечено из доказательства* F*.*
- ***Экстенсиональная полнота****: Каждая арифметическая задача с решением 3-сложности* $\mathbf{R}$ *представима в* $\mathbf{CLA11^R}$*.*
- ***Интенсиональная полнота****: Каждая формула, выражающая задачу с решением 3-сложности* $\mathbf{R}$*, доказуема в* $\mathbf{CLA11^R + TA}$*.*

Параметр $\mathbf{R}$ в $\mathbf{CLA11^R}$ можно навести механическим, прямым, «каноническим» способом, чтобы получить корректную и полную теорию относительно целевой вычислительной 3-сложности.

Например, для

линейной амплитуды + логарифмического пространства + полиномиального времени

мы можем выбрать $\mathbf{R}$ с

$\mathbf{R}_{amplitude} = \{$все термы, построенные из x с помощью $0, ', +\}$ — это именно те термы, которые описывают все линейные функции.

$\mathbf{R}_{space} = \{$все pterm'ы, построенные из $|x|$ с помощью $0, ', +\}$ — можно показать, что это множество термов, выражающих все логарифмические функции.

$\mathbf{R}_{time} = \{$все термы, построенные из x с помощью $0, ', +, \times\}$ — опять же, можно показать, что это именно те термы, которые выражают полиномиальные функции.

Аналогичным образом, без особых усилий можно генерировать примеры **CLA11** для огромного числа вариантов 3-сложности. Таким способом могут быть получены все осмысленные тройки сложности. Например,

- *полиномиальная амплитуда + полиномиальное пространство + полиномиальное время*

- *линейная амплитуда + линейное пространство + квазиполиномиальное время*
- *линейная амплитуда + полиномиальное пространство + экспоненциальное время*
- *суперэкспоненциальная амплитуда + элементарное пространство + примитивно-рекурсивное время*

... осталось только их поименовать!

3. Потенциальные приложения кларифметик

Основным мотивоми изучения **CLA11** и в целом кларифметики является то, что (1) такие теории можно рассматривать, как языки декларативного программирования, и (2) их можно потенциально использовать (возможно, надеюсь) как инструмент для разделения классов вычислимости.

(1) **CLA11** *как язык декларативного программирования*

Допустим, вам нужна программа для вычисления функции целочисленного квадратного корня, которая делает это за полиномиальное время, логарифмическое пространство и линейную амплитуду. Все, что вам нужно сделать, чтобы получить такую программу, это:

1. Установить параметр $\mathbf{R}$ соответствующей 3-сложности (см. конец раздела 2.);
2. Записать формулу $\sqcap x(\exists y(x = y \times y) \to \sqcup y(x = y \times y))$, выражающую вашу задачу в языке $\mathbf{CLA11^R}$;
3. Найти $\mathbf{CLA11^R}$-доказательство этой формулы.

Как только вы найдете доказательство, компилятор извлечет из него искомую программу. Поскольку $\sqcap x(\exists y(x = y \times y) \to \sqcup y(x = y \times y))$ фактически является спецификацией такой программы, доказательство также будет служить автоматической проверкой того, что программа соответствует ее спецификации. Таким образом, пресловутая и вообще неразрешимая проблема проверки программ полностью нейтрализуется.

Помимо этого, шаги доказательства можно рассматривать как (наилучшие из возможных) комментарии соответствующих шагов программы.

Более того, это из области фантастики на данный момент, но все же, если разработать достаточно эффективный прувер, шаг 3 (поиск доказательства) может быть делегирован компилятору, а работа «программиста» будет состоять лишь в том, чтобы просто записать цель/спецификацию $\sqcap x(\exists y(x = y \times y) \to \sqcup y(x = y \times y))$. Таким образом, всех нас можно квалифицировать как программистов, потому что мы можем очень легко понять язык, на котором написана такая формула.

(2) **CLA11** *как инструмент для разделения классов вычислимости.*

Основные открытые проблемы в теории вычислений связаны с разделением различных естественно возникающих классов вычислительной сложности, наиболее известным примером которых является проблема сравнения P с NP. Чтобы показать, что два класса (3-)сложности **R1** и **R2** не равны, достаточно разделить две теории $\mathbf{CLA11^{R1}}$ и $\mathbf{CLA11^{R2}}$. Разделение теорий должно быть проще, чем непосредственное разделение классов сложности. В конце концов, мы видели несколько весьма впечатляющих примеров успешного разделения теорий, таких как разделение разных версий теории множеств или доказательство результатов о независимости. У нас нет сравнимых успешных примеров в разделении классов сложности, что до сих пор в основном было бесплодной борьбой.

Обширный онлайн-обзор CoL и базирующихся на ней теорий можно найти на сайте

`www.csc.villanova.edu/~japaridz/CL/`.

Наиболее близкие по тематике публикации: [Japaridze 2015], [Japaridze 2016c], [Japaridze 2014], [Japaridze 2015], [Japaridze 2016a], [Japaridze 2016b], [Japaridze 2016c].

Литература

Japaridze 2010 – Japaridze, G. “Towards applied theories based on computability logic”, *Journal of Symbolic Logic*, 2010, Vol. 75, pp. 565–601.

Japaridze 2011 – Japaridze, G. “Introduction to clarithmetic I”, *Information and Computation*, 2011, Vol. 209, pp. 1312–1354.

Japaridze 2014 – Japaridze, G. “Introduction to clarithmetic III”, *Annals of Pure and Applied Logic*, 2014, Vol. 165, pp. 241–252.

Japaridze 2015 – Japaridze, G. “On the system CL12 of computability logic”, *Logical Methods is Computer Science*, 2015, Vol. 11, No. 3, Paper 1, pp. 1–71.

Japaridze 2016a – Japaridze, G. “Build your own clarithmetic I: Setup and completeness”, *Logical Methods is Computer Science*, 2016, Vol. 12, No. 3, Paper 8, pp. 1–59.

Japaridze 2016b – Japaridze, G. “Build your own clarithmetic II: Setup and completeness”, *Logical Methods is Computer Science*, 2016, Vol. 12, No. 3, Paper 12, pp. 1–62.

Japaridze 2016c – Japaridze, G. “Introduction to clarithmetic II”, *Information and Computation*, 2016, Vol. 247, pp. 290–312.