Сложность вычислений на абстрактных машинах

И.А. Лавров

В настоящее время мы являемся свидетелями стремительного развития использования вычислительных средств, как в научных исследованиях, так и при решении многих важных прикладных задач. При этом спектр подобных задач неизмеримо расширяется с каждым новым этапом совершенствования технических возможностей. Работа на персональных компьютерах входит во все сферы человеческой деятельности. Задачи обработки различных видов информации с помощью вычислительной техники стала одним из стратегических направлений современной науки и техники.

Современная практика использования вычислительных устройств идет в основном по путям новых инженерных решений, создания новой архитектуры вычислительных машин вплоть до суперкомпьютеров, организации самих вычислений (параллельные вычисления, системное программирование и др.). Все это направлено на значительное увеличение (на порядки) скорости самих вычислений и объемов используемых материалов. Это позволяет включить в орбиту реальных вычислений все более сложные задачи, требующие «большого счета» и огромных ресурсов.

Вместе с тем ясно, что физическая природа процессов, протекающих в подобных вычислительных устройствах, а также объемы ресурсов и используемых материалов, ставят каждый раз перед вычислителями ряд объективных ограничений.

Еще большие трудности возникают не только в физической и технической объективности, а уже в самом абстрактном подходе к понятиям, связанным с вычислительными процессами. Здесь обнаруживаются принципиально «неразрешимые» задачи на современном уровне понимания эффективности эффективных вычислений.

Теоретическим фундаментом для создания и использования вычислительных машин является современная теория алгоритмов. Ее возникновение связано с попыткой точного математического описания такого интуитивно ясного понятия как «***алгоритм***». Почти одновременно в 30-х годах XX в. и позднее в разных формах были даны подобные возможные описания (К. Гедель [29], А.Черч [28], С.Клини [32], Э.Пост [36], А.Тьюринг [45], А.А.Марков [9]). Э.Пост и А.Тьюринг избрали форму такого описания в виде абстрактных машин; они теперь называются ***машинами Тьюринга***. К.Гедель, А.Черч и С.Клини рассматривали эффективные вычисления на языке подходящим образом введенных арифметических функций; такие функции стали называться ***частично рекурсивными***. Сейчас их предпочитают называть ***частично вычислимыми функциями***. А.А.Марков ввел понятие «***нормального алгорифма***». В этом понятии была осуществлена идея точной детерминированности последовательности выполнений преобразований слов в подходящем алфавите.

Как было доказано, все эти понятия, созданные с совершенно различных точек зрения, описывают некое единое «абсолютное» понятие, которое и разумно избрать в качестве математического понятия ***алгоритма***, ***эффективного процесса***, ***эффективного вычисления***.

В настоящее время на русском языке имеется несколько замечательных книг по общей теории алгоритмов (например, [7], [16], [19], [22] и др.)

Задачей, следующей за сформированием понятия «алгоритм» в различных его ипостасях, является вопрос о возможном осознании «сложности» алгоритмов. Это так естественно. В нашей обычной речи мы часто используем такие обороты речи: « X решил трудную (или простую) задачу», «а Y решил еще более сложную задачу, чем X » и пр. Каким образом можно формализовать наше интуитивное понимание сложности?

Эта задача оказалась довольно сложной, а в некоторых направлениях и просто не имеющей удовлетворительного решения.

Общий подход к понятию сложности ***конструктивных объектов*** был разработан А.Н.Колмогоровым [6]. Каждый конструктивный объект может быть задан различными ***конечными описаниями*** в тех или иных терминах, избранных в подходящих математических или лингвистических языках. Тогда ***сложностью конструктивного объекта по Колмогорову*** считается наименьший объем описания такого объекта.

Ясно, что заданные формы описаний приводят к различным представлениям о сложности. Так, в вышеуказанной ситуации, когда под алгоритмом понимается машина Тьюринга, частично рекурсивная функция, нормальный алгорифм либо что-то другое, соответствующее выбранной концепции, можно получить точные варианты понятия сложности алгоритма. Мы рассмотрим три подобные ипостаси понятия сложности: для машин Тьюринга, частично вычислимых функций и подмножеств множества натуральных чисел. Как мы увидим, все эти направления имеют свою специфику и трудности.

Хотелось бы отметить одно специальное направление теории алгоритмов – теорию нумераций. А.Н.Колмогоров предложил систематически изучить возможные обобщения идеи К.Геделя о нумерациях нечисловых областей объектов натуральными числами. Такая нумерация логических формул позволила К.Геделю [29] доказать одну из самых знаменитых в математике теорем – теорему о неполноте арифметики Пеано натуральных чисел. Реализацией построения подобной теории нумераций занимались В.А.Успенский [21], А.И.Мальцев [8], Ю.Л.Ершов [2]. Многие зарубежные специалисты также сумели в терминах теории нумераций доказать важные результаты в теории алгоритмов. В рамках этой теории нумераций удалось обнаружить тесные связи между теорией алгоритмов и классическими направлениями математики.

Интенсивное развитие теории алгоритмов привело к глубокому пониманию сути эффективных вычислений. Были решены многие проблемы, возникли новые подходы и методы для решения возникающих в самом процессе исследований задач. Стала намного понятнее общая объективная картина, связанная с эффективностью вычислений и их сложностями. Вместе с тем остаются еще много проблем, которые ждут своего решения.

Отметим еще одну особенность теории алгоритмов – многие результаты этой теории носят относительно отрицательный характер. Для различных классических или возникших в ходе исследований алгоритмических проблем доказано, что желаемых эффективных алгоритмов (эффективных вычислений) не существует. В этой ситуации, используя соответствующие понятия сложности, приходится строить различные иерархические системы, раскладывающие подобные проблемы на различные уровни сложности. В данной статье будет приведен ряд таких иерархий.

В таких иерархических системах те проблемы, которые обладают разрешающими их эффективными алгоритмами, обычно занимают самый нижний уровень. И с этой точки зрения они становятся теоретически равносложными между собой.

На практике даже для таких проблем разрешающие их алгоритмы имеют свои разнообразные уровни сложности. Те редкие проблемы, для которых доказана их алгоритмическая разрешимость, имеют обычно очень сложные алгоритмы. Более того, те задачи, у которых области исследования конечны, очевидно алгоритмически разрешимы, хотя бы с помощью ***переборных*** алгоритмов. Но такие прямые переборные алгоритмы уже при небольших исходных параметрах имеют грандиозные по величине варианты, требующие перебора. Проблемы подобного рода изучаются в дискретной математике. Здесь часто трудно указать возможные нижние и верхние оценки для спектра перебора. Мы не касаемся в данной статье подобных вопросов.

Мне хотелось дать хотя бы самые общие и первоначальные представления о тех направлениях изучения сложности вычислений на абстрактных вычислительных устройствах, не ограниченных в своих ресурсах, и вычисления на которых могут продолжаться как угодно долго.

Некоторые фрагменты данной статьи автор использовал в своих докладах на VIII японско-российском симпозиуме по вычислительной аэрогидродинамике в Tohoku University (Япония, г.Сендай, 2003 г.) и на школе-семинаре в Международном математическом центре им. С.Банаха (Польша, г. Бедлево, 2004 г.). К докладам был проявлен большой интерес. Слушателями были в основном специалисты и молодые исследователи, занимающиеся вопросами использования вычислительных устройств и методов для решения сугубо прикладных задач. Это дает мне уверенность, что знакомство с материалами статьи будет полезным и для тех, кто занимается системным программированием, прикладными задачами и пр.

1. Сложность машин Тьюринга.

Анализируя интуитивное понятие алгоритма, А.Тьюринг [45] предложил в качестве абстрактной интерпретации вычислительных процессов работу на некоторых идеализируемых устройствах. Задача А.Тьюринга заключалась в отыскании элементарных шагов вычислений с тем, чтобы любое «мыслимое» вычисление могло быть скомбинировано из таких элементарных актов. При этом необходимо было добиться, чтобы спектр получающихся алгоритмов покрывал все известные к этому времени способы вычислений и включал в себя и то, что могло бы быть сконструировано в дальнейшем.

Аналогичные рассмотрения абстрактных вычислительных устройств были проведены Э.Постом [36].Однако в настоящее время подобные вычислительные средства называются ***машинами Тьюринга.***

Не вдаваясь в излишние подробности, можно сказать, что основными элементами машин Тьюринга являются:

1) ***память*** машины, в которую имеется возможность вносить информацию с помощью соответствующего данной машине набора ***символов***; считается, что память всякой машины Тьюринга потенциально бесконечна – это как раз то, что отличает абстрактные машины от реальных вычислительных устройств;

2) ***читающее устройство***, позволяющее считывать информацию, находящуюся в данный момент в памяти; обычно считается, что в отдельный момент времени читающее устройство может обозревать лишь конечное число единиц памяти (чаще всего только одну единицу); устройство может передвигаться с тем, чтобы в следующие моменты времени оно могло обозревать другие куски памяти;

3) ***инженерная конструкция***, которая может в каждый отдельный момент времени находиться в одном из ***внутренних состояний***; обычно одно или несколько таких состояний считаются ***заключительными***, т.е. машина, попав в такое заключительное состояние, останавливается;

4) ***программа работы машины***, указывающая, что и как будет делать машина, если она обозревает конкретную информацию и находится в конкретном внутреннем состоянии.

Работа каждой машины Тьюринга заключается в выполнении последовательных тактов. В начале работы в память данной машины Тьюринга с помощью соответствующих символов вносится информация, машина приводится в одно из выделенных внутренних ***начальных состояний***, читающее устройство обозревает некоторый конкретный, заранее оговоренный кусок единиц памяти. Оказавшись в подобной ситуации, машина Тьюринга в соответствии со своей программой проделывает один такт работы, исполняя то, что программа предписывает.

После выполнения такого такта работы машина Тьюринга снова попадает в некоторую ситуацию и согласно программе проделывает новый такт работы. Удобно время работы машин Тьюринга измерять тактами ее работы.

Включившись для работы, машина Тьюринга будет такт за тактом проделывать свою работу согласно своей программе. Если в какой-то момент времени машина попадет в одно из заключительных состояний, то на этом ее работа заканчивается – машина Тьюринга ***остановится***. Прочитав информацию, размещенную в памяти на этот момент, мы будем иметь результат работы данной машины Тьюринга, отреагировавшей на ту информацию, которую мы задавали с самого начала работы.

Если заключительное состояние не встретится ни на одном этапе работы, то машина Тьюринга будет работать ***бесконечно***. В этом случае надо считать, что мы не получили

никакого результата. Правда, в ходе работы машины Тьюринга мы никогда не будем уверены, что нам встретился именно такой случай – у нас все время будет сохраняться надежда, что остановка машина произойдет на последующих шагах.

Итак, каждая машина Тьюринга может быть полностью задана набором из n ***символов*** $s_1, s_2, ..., s_n$, которые могут наноситься в память данной машины, набором из m ***активных внутренних состояний*** $q_1, q_2, ..., q_m$ данной машины (в это число мы не включаем заключительные состояния) и ***программой*** из $n \times m$ команд, указывающей, что делает машина, обозревая, например, символ $s_i (1 \le i \le n)$, находясь в состоянии q_j $(1 \le j \le m)$. Заметим, что в командах могут встречаться заключительные состояния.

Обычно в такой команде указывается:

а) каким символом заменяется символ s_i;

б) куда передвинется читающее устройство, т.е. какие единицы памяти будут обозреваться на следующем такте;

в) в какое внутреннее состояние перейдет конструкция машины.

Для определенности можно считать, что если в какой-то единице памяти в данный момент не находится никакого символа, то там размещен специально выделенный символ, например, s_1. Таким символом s_1 снабжены все машины Тьюринга. Договоримся также, что в самом начале в ее памяти во всех единицах написан символ s_1 и такая память считается ***пустой***.

Если мы внесем в память машины Тьюринга M некую информацию I (с помощью символов машины), «включим» машину, обозревая ранее оговоренную единицу памяти в некотором начальном состоянии, то машина будет «обрабатывать» эту информацию согласно своей программе. Как мы уже говорили, если машина не остановится, то мы не получим окончательной информации. Если же машина остановится, то информация J, размещенная в этот момент в памяти, будет называться ***результатом*** работы машины M на начальной информации I и обозначаться через $J = M(I)$.

По мысли А.Тьюринга каждый вычислительный процесс может быть смоделирован на подходящей машине. Это так называемый ***тезис Тьюринга***, разделяемый большинством специалистов. Надо заметить, что все известные вычислительные процессы, для которых имеются алгоритмические способы вычислений, удовлетворяет данному тезису. С другой стороны, до сих пор никто не смог предъявить пример такого процесса, который можно было бы признать эффективным (алгоритмическим), но не поддающимся моделированию на машинах Тьюринга. Все это является сильнейшим доводом в пользу справедливости тезиса Тьюринга.

Принципы работы всех машин Тьюринга совершенно одинаковы, их конкретная работа состоит в неукоснительном исполнении своей программы. По этой причине можно, не теряя общности, отождествлять каждую машину Тьюринга с ее программой. Сама же программа носит совершенно конструктивный характер – программу можно задавать конечной таблицей с двумя входами. В клетках таблицы на пересечении i-ой

строки и j-го столбца написана команда о том, что должна сделать данная машина Тьюринга, если она в какой-то момент времени будет обозревать символ s_i, находясь во внутреннем состоянии q_j.

Мы может последовательно перебирать различные программы машин Тьюринга, переходя от одного массива таблиц к другому (это массив связан с фиксированными числами i и j), постепенно увеличивая число строк и столбцов. Массив таблиц, определяемый конкретными числами i и j, конечен. Мы также можем постепенно перебирать все таблицы таких массивов.

Такое перечисление различных программ машин Тьюринга приводит нас к перечислению, или, как говорят, ***нумерации*** всевозможных машин Тьюринга:

$$M_0, M_1, M_2, ... \qquad (1)$$

Мы не требуем от подобной нумерации каких-либо особых свойств. Важны лишь следующие обстоятельства:

1) если нам задана программа машины *M*, то мы можем эффективно указать ее номер в нумерации (1);

2) если нам задана машина M_i, то мы можем эффективно указать ее программу команд.

Итак, мы имеем счетный список (нумерацию) (1) всех машин Тьюринга, а по тезису Тьюринга и всех вычислительных процессов и алгоритмов. Число n называется ***номером*** или ***индексом*** машины M_n.

Интересно отметить, что, не смотря на некоторый произвол, так построенной нумерации (1), она обладает важным свойством ***главности***: для любой нумерации $\{N_n\}$ машин Тьюринга, удовлетворяющей свойствам (1)-(2) существует «хорошо» вычислимая функция $\alpha(x)$ (что это означает, будет ясно из дальнейшего) такая, что $N_n = M_{\alpha(n)}$.

Одним из крупнейших достижений А.Тьюринга было создание концепции ***универсальной машины***. Такая машина Тьюринга U работает следующим образом: если ей задать число n и информацию I, то U «обработает» информацию I как машина M_n. Таким образом, машина U в некотором смысле моделирует любую машину Тьюринга. Кстати, наши современные вычислительные устройства, как правило, работают по принципу универсальных машин Тьюринга. Такое устройство способно исполнять различные задания при соответствующем сообщении о том, какой же процесс нужно проделать.

Что можно было бы назвать сложностью алгоритма (вычисления)? Вполне естественно для этого можно использовать «сложность» машины Тьюринга, реализующей этот алгоритм.

В качестве варианта понятия сложности машины Тьюринга (сложности алгоритма и вычислений) К.Шеннон [42] предложил рассматривать размер программы команд машины Тьюринга, т.е. величину $n \times m$. Напомним, что здесь n — число допустимых для машины символов, а m — число ее внутренних состояний, не считая заключительных.

Сам К.Шеннон указал, что каждая машина Тьюринга размера $n \times m$ может быть заменена на машины размеров $2 \times k$ и $l \times 2$. Под заменой нужно подразумевать построение для машины M другой машины T такой, что при некоторой кодировке информации машины M и T осуществляют «одинаковую» обработку исходной информации.

То, что любой набор символов можно соответствующим образом заменять двумя символами, хорошо известно. Действительно, азбука Морзе отлично справляется с подобными задачами.

Наиболее интересно оценить размеры универсальных машин Тьюринга. Первое впечатление – такие машины должны быть довольно сложными. Но это не так!

Были построены универсальные машины Тьюринга размеров 6×8 и 5×8 (С.Ватанабе [46]) и даже 4×7 (М.Минский [12]).

Исследования по оптимизации $n \times m$ до сих пор продолжаются. Однако, как нам известно, окончательные результаты еще не получены. Сошлемся лишь на работы [15], [41], где указаны универсальные машины Тьюринга размеров

2×18, 24×2, 19×2, 10×3, 7×4, 5×5, 3×10 и даже 4×6.

Итак, для проведения всевозможных, самых замысловатых вычислений необходимо всего 24 команды!

2. Вычисления функций.

На реальных вычислительных устройствах могут выполняться разнообразные действия. Однако ясно, что все такие действия и процессы можно истолковывать как вычисления подходящих функций. Действительно, символы, с которыми оперирует данное вычислительное устройство, могут быть переобозначены (или интерпретированы) числами, а те действия, которые совершаются над символами, будут в этом случае отражать соответствующие функциональные зависимости.

Все это в полной мере относится и к абстрактным машинам Тьюринга. Для придания точности интерпретации вычислений функций на машине Тьюринга M нам нужно заранее оговорить некоторый подходящий способ задания машине Тьюринга информации о числах:

1) для задания аргументов в информацию на этапе начала работы машины; в этом случае будем говорить, что машине заданы значения аргументов;
2) для извлечения результирующего значения функции из информации, полученной на этапе остановки работы машины (т.е. когда она придет в одно из заключительных состояний, например q_0); в этом случае будем говорить, что машина вычислила значение функции.

Итак, можно считать, что машина Тьюринга M ***вычисляет*** функцию $f(x_1, x_2, ..., x_n)$, если, начав работу на аргументах $x_1, x_2, ..., x_n$, она остановится через конечное число шагов и выдаст значение $y = f(x_1, x_2, ..., x_n)$.

Здесь нас подстерегают две неприятности. Во-первых, заключительная информация при остановке машины может не позволить нам извлечь значение функции (например,

эта информация составлена не по правилам заранее оговоренного способа задания информации о числах). Нам ничего не остается делать, как считать, что в данном случае значение функции $f(x_1, x_2, ..., x_n)$ при таких значениях аргументов оказалось неопределенным.

Вторая трудность состоит в следующем. Может так случиться, что, начав работу на каком-то наборе значений аргументов, данная машина M никогда не остановится, т.е. никогда не придет в заключительное состояние. И в этом случае удобно считать, что наша функция при данных значениях аргументов неопределенна. Необходимо при этом понимать, что в процессе работы, проделав конечное число шагов, мы (а тем более машина M) часто не в состоянии понять, что попали именно в подобную ситуацию. Действительно, в любой момент времени (сделав конечное число шагов работы машины M), у нас нет никаких возможностей для точного прогноза дальнейшей работы.

Реальные возможности встречи с этими трудностями заставляют нас не ограничиваться лишь функциями, всюду определенными (тотальными) на соответствующих множествах. Приходится вводить в рассмотрение функции, которые носят название ***частичных***. Желание ограничится лишь всюду определенными функциями в большинстве случаев эффективно не осуществимо.

На практике чаще всего приходится оперировать с функциями действительного переменного, т.е. когда и область определения и область значений данной функции есть множество действительных чисел. Однако мощностные соображения заставляют нас в этих случаях вести лишь приближенные вычисления и заменять вычисление таких функций комплексами вычислений функций, рассматриваемых на множествах менее мощных, чем действительные числа. Действительно, чтобы сообщить машине какое-то действительное число α, нам потребуется выполнить бесконечное (счетное) число актов задания десятичных знаков числа α, а это мы не сможем проделать за конечное время.

По этой причине нам придется ограничиться вычислениями ***арифметических*** функций, т.е. таких функций, аргументы и значения которых представляют натуральные числа из множества $N = \{0,1,2,...\}$.

Итак, мы будем рассматривать лишь частичные арифметические функции $f(x_1, x_2, ..., x_n)$, у которых значения аргументов $x_1, x_2, ..., x_n$ и ее значение являются натуральными числами. Для технических упрощений ограничимся в основном одноместными функциями $f(x)$. Рассмотрение общего случая многоместных функций мало, чем отличается от рассматриваемого.

Если M — произвольная машина Тьюринга, то, задавая ей в начало работы значение аргумента x (с помощью оговоренного способа задания числа), мы либо через конечное число шагов получим значение y (машина остановится и из полученной информации по правилам извлекается число y), либо мы не получим никакого числа y (машина не останавливается или при ее остановке из полученной информации невозможно по правилам извлечь ответ). Тогда мы говорим, что машина ***вычисляет*** частич-

ную арифметическую функцию f, значение которой при аргументе x равно y, а в тех точках, где ответа не получено, функция считается неопределенной. С этой точки зрения каждая машина Тьюринга вычисляет соответствующую ей частичную арифметическую функцию.

Нам удобно перенумеровать все так получаемые функции в соответствии с нумерацией (1) машин Тьюринга

$$\varphi_0, \varphi_1, \varphi_2, \ldots , \qquad (2)$$

имея в виду, что φ_n — та арифметическая функция, которая вычисляется на машине Тьюринга M_n. Будем называть функции из класса $\{\varphi_n\}$ ***вычислимыми по Тьюрингу***.

По общим идеологическим установкам современной математики вычислимые по Тьюрингу функции φ_n — это как раз те функции, которые могут быть алгоритмически вычислены. Это – отражение ***тезиса Тьюринга***, который был сформулирован раньше.

В теории алгоритмов С.Клини [32] построил другим способом, не опирающимся на машины Тьюринга, класс ***частично рекурсивных функций***. То, что частично рекурсивные функции адекватны алгоритмически вычислимым функциям, составляет ***тезис А.Черча***. В свою очередь было доказано, что класс вычислимых по Тьюрингу функций совпадает с классом частично рекурсивных функций. С учетом этого результата можно сказать, что тезисы Тьюринга и Черча «равнообъемны».

Для других уточнений понятия алгоритма, о которых говорилось выше, также можно ввести соответствующие понятия вычислимых функций. Удивительный феномен состоит в том, что все так полученные классы функций совпадают с классом $\{\varphi_n\}$!

Функции $\{\varphi_n\}$ играют в математике исключительную роль. Учитывая вышесказанное, мы их будем называть ***частично вычислимыми*** (имея в виду, что «частично» понимается в смысле, что при некоторых значениях аргументов значение функции может быть неопределенным). Если функция φ_n всюду определена, то она называется ***вычислимой*** (прежнее название — ***рекурсивная***). Если мы будем ограничивать себя лишь вычислимыми функциями (т.е. каким-то регулярным способом выделять их из всего класса частично вычислимых функций), то потеряем многие замечательные свойства фундаментального класса $\{\varphi_n\}$.

Какие принципиальные трудности могут встретиться при вычислениях частично вычислимых функций? Поясним это на следующем.

Каждая частично вычислимая функция может быть представлена в так называемой ***нормальной форме Клини***

$$\varphi_i(x) = A[\mu y\, B(i,x,y) = 0].$$

Здесь A, B — совершенно конкретные функции, довольно простого вида, вычисление которых не доставляет каких-либо трудностей. Символ μy, называемый «***оператором наименьшего числа***», означает, что для данных i и x ищется наименьшее y такое, что $B(i,x,y) = 0$. Основная сложность всех вычислений заключена в нахождении

и вычислении именно такого наименьшего y. Итак, всевозможные вычисления могут быть стандартизированы, а все многообразие частично вычислимых функций получается варьированием числа i (номера функции).

Вычисляя последовательно

$$B(i, \boldsymbol{x}, 0),\ B(i, \boldsymbol{x}, 1),\ B(i, \boldsymbol{x}, 2),\ \ldots$$

(каждое такое вычисление вполне осуществимо в реальное время без особых трудностей), мы должны отыскать первое такое y, чтобы $B(i,x,y) = 0$. При этом может встретиться и такая ситуация – вычислено уже много элементов этой последовательности, а желаемого 0 все еще нет, и у нас (а тем более у машины) нет каких-либо гипотез или предположений о том, что же произойдет в дальнейшем.

Если бы мы еще до вычисления данной последовательности знали, что если требуемое y существует, то мы его должны обязательно обнаружить, проделав $\alpha(i,x)$ шагов для подходящей функции α, то наша задача сразу бы упростилась – проделай $\alpha(i,x)$ шагов вычисления и найди ответ. Но нахождение подобной функции α, которую саму можно было бы эффективно вычислять на подходящей машине Тьюринга, является труднейшей задачей. Более того, для многих вычислимых функций φ подобной функции α просто не существует.

Здесь мало, что зависит от величины значений функции $\varphi_i(x)$. Могло показаться, что вся трудность возникает из-за «больших задач», т.е. таких функций, значениями которых являются очень большие числа. Но оказывается можно ограничиваться функциями, принимающими лишь очень простые значения – 0 и 1. И вся трудность переносится на продолжительность счета таких функций. А все, что говорилось для произвольных функций, справедливо и для функций, имеющих значения {0,1}.

Как же можно сравнивать «трудность» или «сложность» вычислений для частично вычислимых функций?

Первое, что подсказывает интуиция, воспользоваться временными характеристиками. Обозначим через $T_i(x)$ — число шагов (тактов) машины Тьюринга M_i, которые ей понадобятся для вычисления $\varphi_i(x)$; такая функция $T_i(x)$ называется ***временной сигнализирующей*** для машины M_i. Если $\varphi_i(x)$ всюду определенная (вычислимая) функция, то и $T_i(x)$ — всюду определенная функция. Если $\varphi_i(x)$ неопределенна при каком-то значении x, то считаем и $T_i(x)$ неопределенным и пишем $T_i(x) = \infty$. Таким образом, $T_i(x)$ — частично вычислимая функция.

Будем говорить, что вычислимая функция φ_i ***не труднее*** функции φ_j, если $T_i(x) \le T_j(x)$ для почти всех x, т.е. это неравенство выполняется для всех значений x кроме конечного их числа, или, другими словами, оно выполняется для всех натуральных чисел, начиная с некоторого.

Это сравнение сложности вычислений двух функций интуитивно очень понятно и объяснимо. Задача, на которую требуется меньше времени, более простая.

Но посмотрим на эту ситуацию внимательнее. Заметим, что в последовательности (2) каждая отдельная функция $f(x)$ встречается, или, как мы будем говорить, реализуется, как функция $\varphi_i(x)$, бесконечное число раз. Действительно, конкретная функция может иметь несколько различных алгоритмов вычисления, да к тому же вычисление любой функции можно сделать как угодно сложным (длинным), даже просто добавляя несколько ничего не изменяющих «паразитных» шагов. Для каждой функции можно как угодно «удлинить» процесс ее вычисления.

Оказывается, что ситуация намного сложнее. Зафиксируем на дальнейшее некоторую вычислимую функцию $h(x)$.

Теорема об усложнении (Г.Цейтин [23], М.Рабин [38]). Возможно найти такую вычислимую функцию $f(x)$, что для всех ее реализаций

$$f(x) = \varphi_{i(0)}(\boldsymbol{x}) = \varphi_{i(1)}(\boldsymbol{x}) = \ldots\ldots$$

временная сигнализирующая $T_{i(j)}(\boldsymbol{x})$ сложнее, чем $h(\boldsymbol{x})$, т.е. $T_{i(j)}(\boldsymbol{x}) > h(\boldsymbol{x})$ для почти всех $\boldsymbol{x}$.

Оказывается, существуют такие «сложные» функции, для вычисления любой реализации которых необходимо как угодно много времени.

Еще более обескураживающий феномен доставляет следующая теорема.

Теорема об ускорении (М.Блюм [27]). Возможно найти такую вычислимую функцию $f(x)$, что для любой ее реализации $f(x) = \varphi_i(x)$ найдется другая ее реализация $f(x) = \varphi_j(\boldsymbol{x})$ такая, что $T_i(\boldsymbol{x})$ сложнее, чем $h\,T_j(\boldsymbol{x})$, т.е. $T_i(\boldsymbol{x}) > h\,T_j(\boldsymbol{x})$ для почти всех $\boldsymbol{x}$.

Оказывается, существуют такие функции, которые не имеют самого быстрого вычисления. Какой бы алгоритм ее вычисления не был указан, можно эффективно указать более быстрый алгоритм.

Для примера возьмем $h(x) = 2x$. Тогда теорема об ускорении говорит о том, что имеется такая специальная вычислимая функция $f(x)$, что по любому алгоритму ее вычисления можно сконструировать другой алгоритм для вычисления той же функции, но работающий в два раза быстрее.

Такая формулировка этой теоремы часто порождает различные недоумения. Возьмем какую-нибудь подобную вычислимую функцию $f(x) = \varphi_i(x)$ и построим для нее более быстрый (вдвое быстрее, начиная с некоторого значения аргумента) алгоритм $\varphi_j(\boldsymbol{x})$. Но для этого алгоритма $\varphi_j(\boldsymbol{x})$ есть еще более быстрый алгоритм $\varphi_k(x)$. И так можно строить бесконечную серию все более быстрых алгоритмов для функции $f(x)$. Но как же так может быть: для каждого конкретного значения аргумента n функция $T_i(n)$ равна некоторому числу m, и это число невозможно уменьшать в два раза бесконечное число раз!

Чтобы рассеять подобные недоумения, нужно понять, что в теореме об ускорении гарантируется «убыстрение» не для всех значений аргумента, а для ***почти всех***. Естественно, при первом переходе от φ_i к φ_j убыстрение происходило для аргументов, больших какого-то n. Но при следующем переходе от φ_i к φ_k для этого n может никакого убыстрения и не будет – граница, где происходит убыстрение могла сдвинуться к большему числу.

Возвращаясь к теореме об ускорении, хотим еще раз акцентировать внимание на том, что такое ускорение уже имеющегося вычисления возможно реализовать «в любое число раз» т.е. для любой заранее заданной функции $h(x)$!

Были исследованы и другие виды сигнализирующих функций. Например, $V_i(x)$ — объем памяти, т.е. число единиц памяти, использованных M_i для вычисления $\varphi_i(x)$. И для всех рассмотренных видов сигнализирующих функций удалось доказать и теоремы об усложнении и ускорении.

М.Блюм [27] сформулировал довольно простые требования для функций, которые естественно рассматривать как некоторые сигнализирующие. Он называет частично вычислимую функцию $S_i(x)$ сигнализирующей для $\varphi_i(x)$, если

1) для всех x функция $S_i(x)$ определена тогда и только тогда, когда функция $\varphi_i(x)$ определена;
2) следующая функция

$$M(i,n,m) = \begin{cases} 1, & \textit{если } S_i(n) = m, \\ 0, & \textit{в противном случае} \end{cases}$$

является вычислимой.

Удивительный факт состоит в том, что для всех таких сигнализирующих обе вышеизложенные теоремы оказываются справедливыми. Это говорит о том, что проблема введения понятия сложности для вычисления арифметических функций оказывается совершенно нетривиальной и сложной задачей.

Итак, идея связать вычисление функции с каким-либо параметром (сигнализирующей), имитирующим некую «сложность» этого вычисления, и по которому можно будет сравнивать вычисления, не приводит к достаточно объективной картине.

Еще несколько слов о нумерации (2) всех частично вычислимых функций, которая индуцирована нумерацией (1) машин Тьюринга. Отметим два важных обстоятельства, связанных с нумерацией (2):

1) если нам задана программа машины Тьюринга, вычисляющая функцию f, то можно эффективно указать номер этой функции в списке (2);
2) если нам задана функция φ_i, то можно эффективно указать программу машины Тьюринга, вычисляющую функцию φ_i.

Может показаться, что некий произвольный выбор подобных нумераций создает те трудности, которые возникли при определении сложности вычислений. В частности, то обстоятельство, что каждая функция имеет бесконечно много номеров в нумерации (2). Может быть, если такую нумерацию подбирать специально, то ряда отрицательных эффектов можно избежать. Например, можно ли построить ***однозначную*** нумерацию частично вычислимых функций, т.е. такую нумерацию, где каждая функция встречается ровно один раз?

Да, такая однозначная нумерация была построена Р.Фридбергом [31]! Но эта нумерация оказывается крайне сложной и не обладает многими свойствами, необходимыми

для доказательства важных теорем. Дело в том, что нумерация (2), в некотором смысле, содержит в себе все эффективные нумерации класса частично вычислимых функций; как говорят, она является ***главной нумерацией*** для этого класса. А нумерация Фридберга перестает быть главной для этого класса.

Теперь постараемся эффективно выбрать из всех частично вычислимых функций только вычислимые функции. Другими словами, будем искать такую вычислимую функцию α, чтобы класс $\{f_i(x)=\varphi_i(\alpha(x))\}$ содержал все вычислимые функции и только их. Если бы такая функция α нашлась, то функция $f_x(x)+1$ также была бы вычислимой функцией $f_j(x)$ для некоторого конкретного номера j. Но этого быть не может, так как в точке $x=j$ имеем $f_j(j)\neq f_j(j)+1$.

В следующем разделе мы увидим, что для множеств можно найти более адекватную возможность для понятия сложности решения проблем. В связи с этим часто заменяют функции их ***графиками***. А именно, графиком функции $f(x)$ назовем множество

$$\Gamma_f=\{c(x,y)/\ \ y=f(x)\},$$

где $c(x,y)$ — некоторая двуместная функция, кодирующая натуральными числами пары натуральных чисел.

3. Вычисления, связанные с множествами.

Пусть $A\subseteq N$ — произвольное подмножество множества натуральных чисел. Для множества A можно сформулировать ***проблему принадлежности***: построить алгоритм (вычисление) отвечающий на любой вопрос вида «$x\in A$?». Часто эту проблему называют ***проблемой разрешимости*** для множества, однако это будет приводить к неудобным фразам типа «проблема разрешимости неразрешима» и пр.)

Для двух особых множеств — пустого множества $\varnothing$ и самого множества N, такие алгоритмы тривиальны:

для множества $\varnothing$ на любой вопрос «$x\in A$?» отвечай всегда «нет»;

для множества N на любой вопрос «$x\in A$?» отвечай всегда «да».

Если мы свяжем с множеством A так называемую ***характеристическую функцию***

$$\chi_A(x)=\begin{cases}1, & \text{если}\quad x\in A,\\ 0, & \text{если}\quad x\notin A,\end{cases}$$

то алгоритмическое решение проблемы принадлежности для множества A состоит в вычислении функции $\chi_A(x)$. Если функция $\chi_A(x)$ вычислима, то и множество A называется ***вычислимым*** (старое название ***рекурсивное***). Таким образом, для вычислимых множеств A проблема принадлежности алгоритмически разрешима.

Однако вычислимых множеств сравнительно мало, их счетное число (их не больше, чем всевозможных алгоритмов), в то время как имеется континуум различных подмножеств множества N. Итак, подавляющее число множеств являются невычислимыми. У нас остается единственная возможность — сравнивать «по сложности» вычислений характеристические функции множеств. Как это можно сделать?

Э.Пост [37] предложил следующий способ сравнения проблем принадлежности для множеств. Скажем, что множество A T***-сводится*** (Тьюрингово сводится) к множеству B (символически $A\leq_T B$), если можно построить алгоритм распознавания принадлежности чисел к множеству A, при условии, что нам известны ответы на любые вопросы вида «$y\in B$?». Проще говоря, если бы нам был известен алгоритм для решения проблемы принадлежности для множества B, то мы смогли бы построить алгоритм для решения проблемы принадлежности для множества A. Это неформальное пояснение можно сформулировать в точных терминах, связанных с характеристическими функциями χ_A,χ_B и частично вычислимыми функциями.

Естественно считать, что если $A\leq_T B$, то проблема принадлежности для A ***не труднее*** аналогичной проблемы для B.

Вполне могло случиться, что для конкретных множеств A и B имеются две сводимости: $A\leq_T B$ и $B\leq_T A$. В этом случае говорим, что множества A и B T***-эквивалентны*** (символически $A\equiv_T B$). Тогда T***-степенями*** называем классы эквивалентных между собой множеств. T-степени часто называют ***степенями неразрешимости***.

Тьюрингова сводимость частично упорядочивает T-степени по их сложности, достаточно объективно отражая существо предмета. Изучение структуры T-степеней привлекло внимание многих специалистов теории алгоритмов. На эту тему опубликовано значительное число статей и монографий, здесь решены многие трудные задачи, созданы оригинальные методы исследований. Однако ряд проблем в рассматриваемом круге вопросов еще ждут своего решения. Достаточно полные обзоры результатов по T-степеням имеются в [16], [19], [20].

Если мы возьмем любое вычислимое множество A и произвольное множество B, отличное от $\varnothing$ и N, то понятно, что $A\leq_T B$. Действительно, рассмотрим следующую функцию

$$h(x)=\begin{cases}b, & \text{если}\ \chi_A(x)=1,\\ c, & \text{если}\ \chi_A(x)=0,\end{cases}$$

где $b\in B, c\notin B$. Тогда $\chi_A(x)=\chi_B(h(x))$, или другими словами $x\in A\ \leftrightarrow\ h(x)\in B$ (символ $\leftrightarrow$ означает фразу «тогда и только тогда, когда»).

Это тривиальное утверждение как раз и означает очень простой вид T-сводимости A к B. Итак, вычислимое множество T-сводится к любому собственному подмножеству B множества N. Вычислимые множества не труднее любого множества. Все вычислимые множества составляют минимальную T-степень, которую обозначают через **0.** Множества $\varnothing$ и N сразу относим к степени **0**.

Строение множества T-степеней с отношением T-сводимости оказалось довольно сложным с алгебраической точки зрения. В частности, существуют «***несводимые***» (не-

сравнимые) друг к другу множества, т.е. такие множества A и B, что не выполняется ни $A \le_T B$, ни $B \le_T A$. Для несводимых множеств принято обозначение $A \mid B$.

Вопрос о существовании подобных несводимых множеств составлял в свое время так называемую ***проблему Поста***. Для ее решения Р.Фридбергом [30] и А.А.Мучником [13] был создан «***метод приоритета***», который в настоящее время стал одним из основных инструментов в теории алгоритмов.

Если провести некоторую аналогию с теоретическим программированием, то заменяя термин «T -сводится» термином «транслируется», можно сказать, что эти несводимые множества не транслируются между собой, т.е. ни одно из них не является «транслятором» другого. К тому же, если учесть, для данного множества B имеется не более счетного числа множеств A таких, что $A \le_T B$, то получается, что не существует «универсального транслятора», к которому транслируются все множества, которых континуальное число.

Более подробно рассмотрим другую сводимость множеств, которая является наиболее простым и естественным вариантом T -сводимости.

Скажем, что множество A m ***-сводится*** к множеству B (символически $A \le_m B$), если существует вычислимая функция $h(x)$ такая, что для всех натуральных чисел x

$$x \in A \leftrightarrow h(x) \in B$$

Э.Пост называл такую сводимость ***много-односводимостью*** (many-one reducibility); отсюда в обозначении и появляется индекс m. Ранее, когда мы T -сводили вычислимое множество A к любому $B \neq \varnothing, N$, там в действительности имели m -сводимость.

При m -сводимости $A \le_m B$ для решения распознавания принадлежности любого x множеству A нам нужно решать аналогичную задачу о принадлежности к B всего одного числа $h(x)$, которое получается по x с помощью вычислимой функции. И в этом случае можно считать, что множество A ***не труднее*** множества B.

Если $A \le_m B$ и $B \le_m A$, то множества A и B называются m ***-эквивалентными*** по m -сводимости (символически $A \equiv_m B$). По мысли Э.Поста m -эквивалентные множества в соответствующем закодированном виде содержат одинаковые массивы информации.

Как и раньше, m ***-степенями*** называем классы эквивалентных между собой множеств, их мы будем обозначать полужирными строчными буквами типа **a, b, c, …**.

m -степени частично упорядочиваются так: $\mathbf{a} \le \mathbf{b} \leftrightarrow A \le_m B$ для $A \in \mathbf{a}$, $B \in \mathbf{b}$. Если **a** и **b** не сводятся друг к другу, то называем их ***несводимыми*** (***несравнимыми***) и обозначаем это через $\mathbf{a} \mid \mathbf{b}$. Все вычислимые множества лежат в минимальной m -степени **0**. В эту степень **0**, как и раньше, включаем и степени множеств $\varnothing$ и N. Далее мы будем часто опускать приставку m -, имея в виду, что далее везде будем говорить о m – сводимости, m -степенях и т.п. Мы увидим, что m -степени являются удобными вариантами оценок сложности множеств**.**

Введем в рассмотрение один новый класс множеств, играющий в теории алгоритмов такую же фундаментальную роль, как и частично вычислимые функции. Эти множества

будут называться ***вычислимо перечислимыми*** (прежнее из название – ***рекурсивно перечислимые множества***).

Отличительной особенностью вычислимо перечислимых множеств является то, что для них существует алгоритм их пошагового ***перечисления*** (порождения). Если мы возьмем какую-нибудь вычислимую функцию $f(x)$, то процесс перечисления конкретного множества A можно истолковывать, как последовательное вычисление значений $f(0), f(1), f(2), \ldots$ В этом случае считаем

$$A = \{f(0), f(1), f(2), \ldots\}.$$

Можно перечислять множество A и с помощью частично вычислимой функции φ, имея в виду, что x будет отнесено к множеству A в тот момент, когда оно получится как значение функции φ. Легко доказать, что перечисление этого же множества можно организовать и с помощью подходящей вычислимой функции. Конечно, это справедливо лишь для непустых множеств. Но множество $\varnothing$ мы также будем сразу относить к вычислимо перечислимым множествам.

Таким образом, можно считать, что вычислимо перечислимые множества являются областями значений частично вычислимых функций. В частности, пустое множество $\varnothing$ — область значений функции, которая всюду неопределенна.

Такое определение вычислимо перечислимых множеств приводит к их нумерации, индуцированной нумерацией (2) частично вычислимых функций:

$$W_0, W_1, W_2, \ldots, \tag{3}$$

где W_i считается областью значений функции φ_i.

Заметим, что нумерация (3) также как и нумерация (2) неоднозначна, одно и тоже множество W_i будет повторяться бесконечное число раз. Р.Фридберг [31] построил однозначную нумерацию и для класса всех вычислимо перечислимых множеств. Но и эта нумерация не является главной для этого класса всех вычислимо перечислимых множеств.

Ясно, что всякое вычислимое множество A и его дополнение $\overline{A}$ являются вычислимо перечислимыми: надо для x вычислять $\chi_A(x)$, и если это значение получится равным 1, то перечислить x в A; если же это значение получится равным 0, то отнести x в $\overline{A}$.

Но не всякое вычислимо перечислимое множество является вычислимым. Здесь справедлива теорема, принадлежащая Э.Посту [37].

Теорема Поста. Множество вычислимо тогда и только тогда, когда оно само и его дополнение вычислимо перечислимы.

Для доказательства рассмотрим два перечисления — самого множества и его дополнения:

$$A = \{f(0), f(1), f(2), \ldots\},$$
$$\overline{A} = \{g(0), g(1), g(2), \ldots\}.$$

Тогда алгоритм вычисления характеристической функции $\chi_A(x)$ состоит в следующем:

1) перечисляй A и $\overline{A}$ до тех пор, пока в одном из этих перечислений не встретится x - это обязательно случится, так как любое число попадет в одно из этих множеств;

2) если x встретится в перечислении множества A, то $\chi_A(x)=1$;

3) если x встретится в перечислении множества $\overline{A}$, то $\chi_A(x)=0$.

Если же про множество A известно только, что оно вычислимо перечислимо, то вычисление $\chi_A(x)$, используя его перечисление $A=\{f(0),f(1),f(2),...\}$, столкнется с трудностями. Мы будем иметь как бы «половину вычисления» характеристической функции $\chi_A(x)$: она эффективно вычислится при $x\in A$. Но если $x\notin A$ то, вычислив как угодно длинный кусок нашего множества $A^s=\{f(0),f(1),...,f(s)\}$, мы там не обнаружим x, и у нас все время будет сохраняться сомнение – а не появится ли это x на последующих шагах перечисления? В этом случае мы так и не дождемся момента, чтобы положить $\chi_A(x)$ =0, таким образом функция $\chi_A(x)$ в такой точке останется неопределенной.

По этой причине удобнее для множества A рассматривать ***частично характеристическую функцию***

$$\chi_A^*=\begin{cases}1, & \text{если } x\in A\\ \text{неопределено} & \text{в противном случае.}\end{cases}$$

В соответствии с этим замечанием можно дать другое определение вычислимо перечислимых множеств: множество вычислимо перечислимо, если его частично характеристическая функция частично вычислима.

Процесс перечисления вычислимо перечислимых множеств напоминает процесс постепенного доказательства теорем из некоторого набора аксиом, например, в евклидовой геометрии, или другой аксиоматической теории. Аксиоматический метод – один из самых излюбленных математиками способов построения своих теорий.

Обычно изучение аксиоматических теорий ведется в языке исчисления предикатов и функций первого порядка (ИПФ), внелогические символы которого фиксируют обозначения соответствующих для этой теории основных предикатов и функций. Если использовать подходящую нумерацию всех допустимых формул в таком логическом языке (такие нумерации получили название ***геделевских***, по имени К.Геделя, который эффективно применил подобные нумерации при доказательстве самой знаменитой его ***теоремы о неполноте арифметики Пеано натуральных чисел*** [29]), то возможно на логические теории перенести основные понятия теории алгоритмов.

В соответствии с этим мы будем говорить о степенях логических теорий. Если эта степень равна **0** (т.е. множество номеров всех верных (выводимых) в этой теории формул вычислимо), то такая теория называется ***разрешимой***, в противном случае – ***неразрешимой.*** Позднее мы приведем примеры как разрешимых, так и неразрешимых теорий.

Вычислимо перечислимые множества по отношению к сводимости составляют обособленное замкнутое семейство: ясно, что если $A\le_m B$ и B вычислимо перечислимое

множество, то таким же будет и множество A: надо перечислять множество B и вычислять функцию $f(x)$, и как только какое-то $f(x)$ перечислится в B, число x надо перечислить в множество A.

Среди вычислимо перечислимых множеств Э.Пост [37] открыл замечательный класс множеств и назвал эти множества ***креативными*** (***творческими***). Все креативные множества лежат в одной степени, которую мы обозначим через **1**. К этой степени **1** сводятся все другие степени вычислимо перечислимых множеств. Таким образом, степень **1** является самой максимально трудной среди таких степеней, а в любое креативное множество можно универсально транслировать произвольное вычислимо перечислимое множество («***универсальный транслятор***»).

Построить какое-нибудь креативное множество не так трудно. Пусть $c(x,y)$ — уже упоминавшаяся функция, кодирующая натуральными числами пары натуральных чисел. Рассмотрим следующее множество:

$$K_0=\{c(x,y)/\ x\in W_y\}.$$

Ясно, что это множество вычислимо перечислимо: надо одновременно перечислять все множества W_i, и как только мы увидим, что $x\in W_y$, число $c(x,y)$ - номер пары $<x,y>$, перечислить в множество K_0. Вместе с тем множество K_0 не вычислимо, т.к. его дополнение не является вычислимо перечислимым. Также ясно, что к множеству K_0 сводятся все вычислимо перечислимые множества. Действительно, возьмем любое вычислимо перечислимое множество A. Оно в нумерации (3) имеет некоторый номер n (т.е. $A=W_n$). Но тогда имеем

$$x\in W_n\leftrightarrow c(x,n)\in K_0,$$

что означает сведение множества W_n к множеству K_0 с помощью функции $c(x,n)$.

Первый пример неразрешимой проблемы был указан самим А.Тьюрингом [45]. Это так называемая ***проблема остановки*** (***halting problem***). Один из вариантов проблемы остановки состоит в следующем. Пусть H означает множество номеров машин Тьюринга в нумерации (1), которые обязательно остановятся, начав работу с пустой информации (в памяти машины ничего не записано). Ясно, что множество H вычислимо перечислимо. Но оказывается, что H - креативное множество. В частности, это множество не вычислимо – его дополнение не является вычислимо перечислимым множеством.

Обозначим через AP множество номеров всех верных теорем классической ***арифметики Пеано натуральных чисел*** со сложением и умножением. Множество AP креативно, что свидетельствует о том, что теория чисел – наука чрезвычайно трудная (J.B.Rosser). С другой стороны, арифметика натуральных чисел только с одним сложением является разрешимой теорией (M.Presburger).

Аналогично, оказались креативными арифметики целых и рациональных чисел (в качестве основных операций здесь снова берутся сложение и умножение) (A.Tarski, A.Mostowski, J.Robinson). Однако, арифметика действительных чисел имеет степень **0**, она разрешима (A.Tarski). Это несколько противоречит нашей интуиции – действитель-

ные числа нам всегда представляются как очень трудные объекты для изучения. Но такой феномен может быть объяснен и так: в логическом языке для теории действительных чисел невозможно выразить наиболее сложные утверждения об этих числах. Здесь мешает некоторая ограниченность логических языков того уровня, который обычно допускается. Например, здесь нет адекватных средств, с помощью которых можно было бы говорить о мощностях множеств. Логический же язык для теории натуральных чисел затрагивает необычайно богатый мир понятий и результатов в этой области.

Итак, арифметика действительных чисел разрешима. Алгоритм для этой теории, позволяющий для любого утверждения в соответствующем языке узнать, верно оно или нет для действительных чисел, разработал A.Tarski [43].

Аналогичная ситуация и с теорией комплексных чисел – она разрешима. Далее, известно, что элементарная геометрия с помощью введения координат может быть рассмотрена внутри теории действительных чисел. Это означает, что алгоритм А.Тарского будет успешно работать и в геометрии. То же самое можно сказать об элементарной гиперболической геометрии (A.Tarski, W.Schwabhäuser).

Были проанализированы на предмет их разрешимости или неразрешимости многие классические математические теории. Обзоры полученных результатов в этом направлении имеются в [44] и [5]. Мы приведем здесь наиболее впечатлительные результаты на эту тему, и если о них говорится в обзоре [5], то будем указывать лишь его авторство, отсылая за ссылками к этому обзору. Также приведем небольшое число примеров элементарных теорий, для которых окончательные решения об их разрешимости или неразрешимости получены после выхода обзора [5], т.е. после 1965 г.; здесь дадим соответствующие ссылки.

Например, теория групп оказывается креативной (A.Tarski). Неразрешимой оказывается теория конечных групп (А.И.Мальцев). Заметим, что данная теория не является аксиоматизированной теорией. Аналогичное положение с теориями полугрупп, колец, структур – все эти теории, как и теории их конечных моделей, неразрешимы (A.Tarski, А.И.Мальцев).

Очень неожиданным оказалось, что теория абелевых групп разрешима (W.Szmielew). Алгоритм для этой теории очень сложен.

Теория полей неразрешима (J.Robinson), но Ю.Л.Ершов [3], J.Ax и S.Kochen [24], [25] доказали, что теория конечных полей разрешима.

Теории одноместных предикатов, равенства, эквивалентности, одноместных функций, линейно упорядоченных множеств, вполне упорядоченных множеств, булевых алгебр разрешимы (L.Löwenheim, A.Janiczak, H.Läuchli, A.Ehrenfeucht, A.Tarski, A.Mostowski).

В противовес этому теория двухместных предикатов, частично упорядоченных множеств, симметрично-рефлексивных предикатов, как и подобные теории для конечных моделей, неразрешимы (A.Church, L.Kalmar, A.Tarski, R.L.Vaught, H.Rogers,

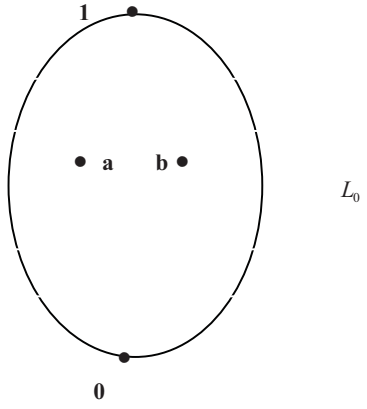
Б.А.Трахтенброт, М.А.Тайцлин, И.А.Лавров). Отсюда, в частности следует, что теория графов и теория конечных графов также алгоритмически неразрешимы.

Отметим еще несколько результатов, относящихся к нелогическим направлениям математики. Проблема тождества слов в группах алгоритмически неразрешима (П.С.Новиков [14]). Неразрешимой оказывается проблема гомеоморфизма в топологии (А.А.Марков [10]).

Еще один пример из алгебры. Проблема: для данного конечного множества 3×3 матриц с целыми коэффициентами узнать, будет ли некоторое конечное произведение элементов этого множества равно нулевой матрице. Эта проблема также алгоритмически неразрешима (M.S.Paterson, [35]).

История теории чисел во многом связана с решением диофантовых уравнений, т.е. нахождением целочисленных решений уравнения $p(x_1, x_2, ..., x_n) = 0$, где p - многочлен с целыми коэффициентами. Для ряда конкретных диофантовых уравнений такие решения полностью описаны. Однако даже в настоящее время для не очень сложных многочленов p многочисленные попытки описать решения соответствующего диофантового уравнения не увенчались успехом. В числе знаменитых проблем Д.Гильберта десятая проблема посвящена диофантовым уравнениям: указать алгоритм для решения произвольного диофантового уравнения. Ю.В.Матиясевич [11] доказал, что данная проблема неразрешима.

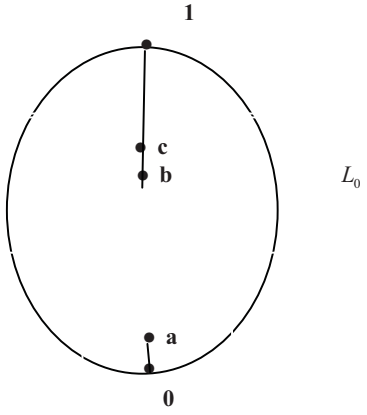
Возвратимся к степеням вычислимо перечислимых множеств. Проведены довольно детальные исследования частично упорядоченного множества L_0 степеней вычислимо перечислимых множеств. Мы приведем лишь небольшое число результатов в этом направлении.



Между **0** и **1** лежит бесконечное множество несводимых между собой степеней. Более того, для каждой степени $\mathbf{a} \in L_0$, отличной от **0** и **1**, существует степень **b** такая, что $\mathbf{a} \mid \mathbf{b}$.

При отыскании вычислимо перечислимых множеств, отличных от вычислимых и креативных, Э.Пост [37] открыл ряд удивительных классов множеств. Приведем лишь один пример на этот счет.

Если множество A креативно, то оно само вычислимо перечислимо, но его дополнение $\overline{A}$ хотя и бесконечно, но не является вычислимо перечислимым. Вместе с тем в $\overline{A}$ можно эффективно выделить бесконечное вычислимо перечислимое подмножество. Вычислимо перечислимые множества, которые Э.Пост почему-то назвал ***простыми*** (simple) таковы, что в их бесконечном дополнении не содержится ни одного бесконечного вычислимо перечислимого множества. В дополнении $\overline{A}$ простого множества A нельзя эффективно указать бесконечную серию чисел. И таких множеств и их различных разновидностей довольно много.

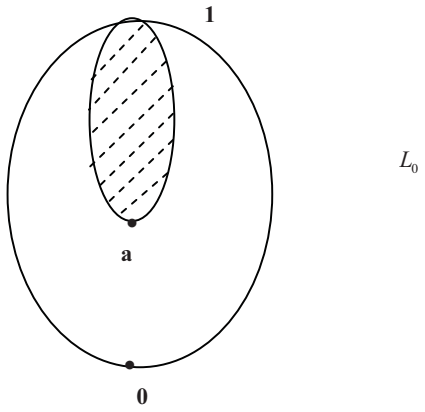


Имеются минимальные степени **a**, т.е. такие степени **a**, что **0 < a**, но между **0** и **a** нет других степеней. С другой стороны, под степенью **1** нет максимальной: если **b < 1**, то существует степень **c** (а значит таких степеней бесконечно много) такая, что **b < c < 1.**

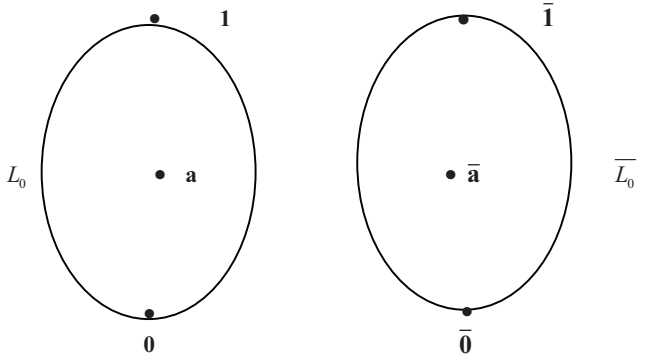
Много усилий было потрачено для того, чтобы каким-то регулярным способом описать ***начальные отрезки*** L_0. Цель была достигнута в [33], [34], [4], [1] – дано вполне удовлетворительное, но сложное алгебраическое описание таких начальных отрезков.

Из этого описания, в частности, следует, что если **a < 1**, то кусок **[a,1]** = ={ **b** / $\mathbf{a} \le \mathbf{b} \le \mathbf{1}$} в точности похож (изоморфен) на все L_0=[**0**,**1**]. Это напоминает эффект голографии.

По-другому можно сказать и так: если бы в самом начале теории алгоритмов в качестве вычислимых (разрешимых) множеств был избран другой вариант – «вычислимыми множествами» считались бы множества из степени $\mathbf{a} \neq \mathbf{1}$, то качественная картина семейства степеней осталась бы прежней.



Если мы обозначим через $\overline{\mathbf{a}}$ степень таких множеств A, дополнение которых $\overline{A}$ принадлежит степени **a**, то строение двух таких семейств L и $\overline{L_0}$ будут совершенно идентичны (изоморфны) и будут соприкасаться лишь в степени **0** (так как $\mathbf{0}=\overline{\mathbf{0}}$).

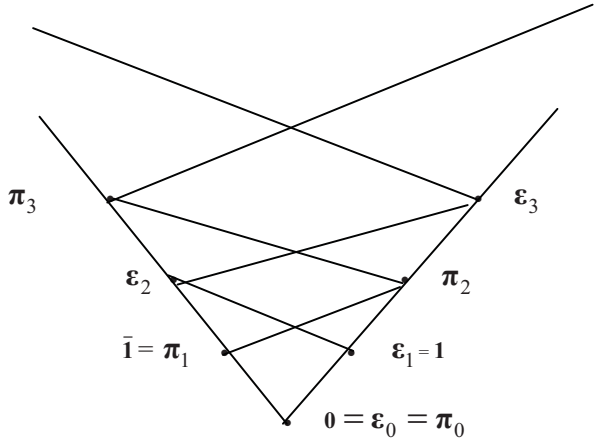


Если брать степени не только вычислимо перечислимых множеств и их дополнений, то картина уровней степеней окажется еще более сложной. Не смотря на это, удается оценить (а иногда и точно локализовать) сложность многих классических проблем и задач.

Дадим лишь два примера на этот счет. Проблема принадлежности к множеству $Fin = \{ x / W_x$ - конечное множество$\}$ имеет так называемую сложность $\boldsymbol{\varepsilon}_2$, а проблема принадлежности к множеству $Infin = \{ x / W_x$ - бесконечное множество$\}$ имеет сложность $\boldsymbol{\pi}_2 = \overline{\boldsymbol{\varepsilon}_2}$. Позднее мы объясним, почему применяем здесь такие обозначения. Кстати, степень **1** при таких обозначениях имеет сложность $\boldsymbol{\varepsilon}_1$, степень $\bar{\mathbf{1}}$ — сложность $\boldsymbol{\pi}_1$, а степень **0** — сложность $\boldsymbol{\varepsilon}_0 = \boldsymbol{\pi}_0$.

Обе эти степени $\boldsymbol{\varepsilon}_2$ и $\boldsymbol{\pi}_2$ — проблемы узнавания про вычислимо перечислимое множество, является ли оно конечным или бесконечным, не сводимы друг к другу и более сложные, чем **1** и $\bar{\mathbf{1}}$. Можно строить еще более сложные степени $\boldsymbol{\varepsilon}_3, \boldsymbol{\pi}_3, \boldsymbol{\varepsilon}_4, \boldsymbol{\pi}_4, \ldots$

Взаимоотношения между такими степенями можно схематически изобразить следующим образом:



Попробуем записать логическую формулу, задающую множество $Fin : x \in Fin \leftrightarrow \exists y \forall z [y < z \rightarrow z \notin W_x]$ (в формуле говорится, что, начиная с некоторого места, все числа не принадлежат W_x).

В теории алгоритмов известно, что $a \in W_b \leftrightarrow \exists c R(a,b,c)$ для некоторого вычислимого предиката R, т.е. такого предиката, что множество $\{c_3(a,b,c) / R(a,b,c)\}$ вычислимо. Здесь c_3 — вычислимая функция, взаимно однозначно нумерующая натуральными числами всевозможные тройки натуральных чисел.

Легкие преобразования написанной выше формулы для Fin приведут нас к формуле $x \in Fin \leftrightarrow \exists y \forall z Q(x,y,z)$ для некоторого вычислимого предиката Q. Последняя формула записана в так называемой пренексной нормальной форме, имеющей такую структуру: в начале (приставка формулы) идут кванторы, после кванторной приставки записан некоторый вычислимый предикат. Оценка ***типа*** подобных формул вычисляется так – это будет $\sum_n$ или $\prod_n$ в зависимости от того, с какого квантора $\exists$ или $\forall$ начинается кванторная приставка, а число n означает число перемен разноименных кванторов. Для множества Fin тип формулы получается $\sum_2$. А степень этого множества $\boldsymbol{\varepsilon}_2$ — максимальная по сводимости степень множеств типа $\sum_2$. Заметим, что в приставке $\exists\forall$ имеется две перемены кванторов – сперва идут $\exists$. потом $\forall$.

Аналогичные рассуждения свидетельствует, что множество $Infin$ имеет тип $\prod_2$ и его степень $\boldsymbol{\pi}_2$ максимальна среди степеней множеств типа $\prod_2$.

Все множества типов $\sum_n$ и $\prod_n$ называются ***арифметическими***, для них можно построить подобные формулы с соответствующими кванторными приставками. Конечно, арифметических множеств счетное число, поэтому большинство подмножеств множества натуральных чисел носят не арифметический характер и их степени никак не отмечены в нашей схеме степеней.

Отметим еще одно обстоятельство, навеянное множествами Fin и $Infin$. Эти множества обладают следующим свойством: если какой-то номер (индекс) n множества A попадает в такое множество, то туда же попадут и все другие номера (индексы) этого же множества. В устоявшейся терминологии теории алгоритмов множества с таким свойством называются ***индексными***.

Пусть $\mathfrak{A}$ – некоторое семейство вычислимо перечислимых множеств. Тогда

$$\theta\,\mathfrak{A} = \{ x / W_x \in \mathfrak{A} \}$$

называется ***индексным множеством*** семейства $\mathfrak{A}$. Проблему принадлежности к индексному множеству $\theta\,\mathfrak{A}$ некоторого семейства $\mathfrak{A}$ можно истолковывать как ***проблему распознавания*** определенного свойства. Другими словами под проблемой распознавания понимаем задачу построения алгоритма, который по любому числу x отвечает, обладает или нет множество W_x свойством, характеризующим семейство $\mathfrak{A}$.

Так семейство Fin индуцирует следующую проблему распознавания: является или нет множество W_x конечным? Семейство же $Infin$ связано с проблемой распознавания

бесконечности множества W_x. Мы уже знаем, что обе эти проблемы не являются алгоритмически разрешимыми, их степени $\boldsymbol{\varepsilon}_2$ и $\boldsymbol{\pi}_2$.

Для двух семейств, $\mathfrak{A}_1$ – пустого семейства, т.е. не содержащего ни каких множеств, и $\mathfrak{A}_2$ полного семейства, т.е. содержащего все вычислимо перечислимые множества, проблемы распознавания очевидно разрешимы. Эти два семейства и индуцированные ими свойства называются тривиальными. Как следует из следующей теоремы, эти семейства совершенно уникальны по отношению к проблеме распознавания.

Теорема Райса.[39] Всякое нетривиальное свойство не распознаваемо.

Когда мы рассматриваем множество W_x, то его номер – это номер программы машины Тьюринга, порождающей это множество. А номер программы – это по существу сама программа, конкретно выписанная. Впечатление о том, что, зная программу машины, мы в принципе можем отвечать на многие вопросы о самом процессе вычисления и его результатах, оказывается полностью дискредитированным. Имея перед собой такую программу, мы не сможем ответить ни на один общий разумный вопрос.

Конечно, на ряд вопросов для конкретных программ могут быть получены правильные ответы, но общая картина здесь совершенно безнадежная.

В [40] дано достаточно ясное описание семейств вычислимо перечислимых множеств, индексные множества которых вычислимо перечислимы (множества типа $\sum_1$): это эффективные объединения открытых множеств типа $\{W_x \,/\, D \subseteq W_x\}$, где D – конечное множество, в топологии на классе всех вычислимо перечислимых множеств (теорема Райса-Шапиро).

Из-за простой связи множеств типа $\prod_1$ и $\sum_1$ можно получить и описание семейств вычислимых перечислимых множеств, индексные множества которых являются дополнениями вычислимо перечислимых множеств.

К сожалению, не смотря на многочисленные попытки, в настоящее время нет удовлетворительного описания (в терминах как в теоремах Райса и Райса-Шапиро) семейств вычислимо перечислимых множеств, индексные множества которых имеют типы $\sum_n$ и $\prod_n$ при $2 \le n$. В работах [17], [18] приведен ряд доводов, свидетельствующих о трудности подобных проблем, особенно в случае $n = 2$.

Литература.

[1] Денисов С.Д., Строение верхней полурешетки рекурсивно перечислимых множеств m m -степеней и смежные вопросы. I, Алгебра и логика, 1978, 17, 6, 643-683.

[2] Ершов Ю.Л., Теория нумераций, М.: Наука, 1977.

[3] Ершов Ю.Л., Об элементарных теориях локальных полей, Алгебра и логика, 1965. 4, 2, 5-30.

[4] Ершов Ю.Л., Верхняя полурешетка нумераций конечного множества, Алгебра и логика, 1975Ю 14, 3, 258-284.

[5] Ершов Ю.Л., Лавров И.А., Тайманов А.Д., Тайцлин М.А., Элементарные теории, УМН, 1965, 20, 4, 37-108

[6] Колмогоров А.Н., Три подхода к определению «количества информации», Проблемы передачи информации, т.1, в.1, 3-11

[7] Мальцев А.И., Алгоритмы и рекурсивные функции, М.: Наука, 1986.

[8] Мальцев А.И. Конструктивные алгебры, I, УМН, 1961, 16, 3, 3-60.

[9] Марков А.А., Теория алгорифмов, Труды математического института АН СССР, 1951, 38, 176-184; 1954, 42.

[10] Марков А.А., Неразрешимость проблемы гомеоморфизма, Proc. Inter.Congress Math., London, Cambridge, University Press,1958, 300-306.

[11] Матиясевич Ю.В., Диофантовость перечислимых множеств, ДАН СССР, 1970, 191, 279-282.

[12] Минский М. (Minsky M.L), Вычислимость и автоматы, М.: Мир, 1971.

[13] Мучник А.А., Неразрешимость проблемы сводимости теории алгоритмов, ДАН СССР, 1948, 108, 194-197.

[14] Новиков П.С., Об алгоритмической неразрешимости проблемы тождества слов в теории групп, Труды математического института АН СССР, 1955, 44.

[15] Рогожин Ю., Семь универсальных машин Тьюринга, Матем.исследования, 1982, 69, 76-90.

[16] Роджерс Х. (Rogers H.), Теория рекурсивных функций и эффективная вычислимость, М.:Мир. 1972.

[17] Селиванов В.Л., Об индексных множествах в иерархии Клини-Мостовского, Труды математического института, Новосибирск, 1982, 2, 135-158.

[18] Селиванов В.Л., Индексные множества в гиперарифметической иерархии, СМЖ, 1984, 25, 164-181.

[19] Соар Р. (Soare R.I.) Вычислимо перечислимые множества и степени, Казань, 2000.

[20] Справочная книга по математической логике, ч.III, Теория рекурсии, ч. III, М.: Наука, 1982.

[21] Успенский В.А., Лекции о вычислимых функциях, М.: Физматгиз, 1960.

[22] Успенский В.А., Семенов А.Л., Теория алгоритмов: основные открытия и приложения, М.: Наука, 1987.

[23] Цейтин Г., Оценка числа шагов при применении нормального алгоритма, Математика в СССР за сорок лет, т.1, М, 1959, 44-45

[24] Ax J., The elementary theory of finite fields, Ann.Math., 1968, 88, 239-271.

[25] Ax J., Kochen S., Diophantine problems over local fields, Amer.J.Math., I-II, 1965, 87, 605-648

[26] Ax J., Kochen S., Diophantine problems over local fields III, Ann. Math. 1966, 83, 437-456.

[27] Blum M., Independent theory of the complexity of recursive functions, Jour. Assoc. Comput. Math., 1967, 14, 322-336. (Русский перевод в книге «Проблемы математической логики», М.: Мир, 1970, 401-422.)

[28] Church A., On unsolvable problem of elementary number theory, Amer.jour.math. 1936, 58, 345-348.

[29] Gödel K., Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme, I Monatsh. Math. und Phyc., 1931,38, 173-198.

[30] Friedberg R.M., Two recursively enumerable sets of incomparable degrees of unsolvability (solution of Post's problem 1944), Proc. Nat. Acad.Sci., 1957, 43, 236-238.

[31] Friedberg R.M., Three theorems on recursive functions: I Decomposition. II Maximal sets. III Enumeration without duplication, Jour.Symb.logic, 1958, 23, 309-316.

[32] Kleene S.C., General recursive functions of natural numbers, Math.Ann., 1936, 112, 727-742

[33] Lachlan A., Initial segments of many-one degrees, Canad. Jour.Math., 1970, 22, 1, 75-85.

[34] Lachlan A., Recursively enumerable many-one degrees, Алгебра и логикаб 1972б 11б 3б 326-358.

[35] Paterson M.S., Unsolvability in 3×3 matrices, Studies in Appl.Math., 1970, 49, 105-107.

[36] Post E.L., Finite combinatory processes – formulation I, Jour.Symb.logic, 1936, 1, 103-105. (Русский перевод в книге В.А.Успенского «Машины Поста», М.:Наука, 1979.)

[37] Post E.L. Recursive enumerable sers of positive integers and their decision problems, Bull. Amer. Math. Soc., 1944, 50, 284-316.

[38] Rabin M.O., Real-time computation, Israel Jour. Math., 1963, 1, 203-211.

[39] Rice H.G., Classes of recursively enumerable sets and their decision problems, Tras. Amer. Math. Soc., 1953, 74, 358-366.

[40] Rice H.G., On completely recursively enumerable classes and their key arrays, Jour. Symb.logic, 1956, 304-308.

[41] Rogozhin J., Small universal Turing machines, Theoretical Computer Science, 1996, 168, 215—240.

[42] Shannon C.E., A universal Turing machine with two internal states, Automata Studies, Princeton, 1956. (Русский перевод в книге «Автоматы», М.: ИЛ, 1956.)

[43] Tarski A., A decision method for elementary algebra and geometry, Berkeley, Los Angeles, 1951.

[44] Tarski A., Mostowski A., Robinson R., Undecidable theories, 1953, North-Holland Publishing Company, Amsterdam.

[45] Turing A. M., On computable numbers, with an application to the Entscheidungsproblem, Proc. London Math.Soc., ser.2, 1936, 42, 230-265;.1937, 43, 544-546.

[46] Watanabe S., 5-symbol 8-state and 5-symbol 6-state universal Turing machines, Jour. Assoc. Comp. Math., 1964, 8, 4, 476-483. (Русский перевод в книге «Кибернетический сборник», М.:ИЛ. 1963, 6.)