Квантовые вычисления
Глава 1

Сысоев Сергей Сергеевич

Санкт-Петербургский государственный университет
Математико-механический факультет
http://se.math.spbu.ru/

1 Введение

Математики и программисты привыкли воспринимать вычисления как абстрактный процесс, не имеющий физической основы. Тем не менее, любое устройство, выполняющее вычисление, основано на каком-либо физическом процессе. Принятие этого факта приводит к отказу от платоновского мира идей, зато открываются великолепные перспективы для экспериментов с различными физическими системами.

Поколения ЭВМ являются наглядной иллюстрацией истории таких экспериментов. До ЭВМ вычисления выполнялись механическими системами. Потом появились устройства, основанные на электромагнитных реле, радиолампах, транзисторах... Каждый новый тип устройства был эффективнее предыдущего – естественно было бы ожидать появления новых, еще более эффективных устройств.

Вместе с устройствами эволюционировали и алгоритмы. В 1936 году Алан Тьюринг предложил математическую модель вычислений, названную впоследствии машиной Тьюринга. Модель Тьюринга формализовала понятие алгоритма (способа вычисления некоторой функции на физическом устройстве), что позволило ввести понятие вычислимости и, в дальнейшем, сложности вычислений. Тогда же, в 1936 году, Тьюринг показал, что существуют невычислимые (undecidable) функции – такие функции, которые можно формально определить, но для вычисления которых невозможно предложить алгоритм.

Понятие сложности вычислений (количества ресурсов, необходимых для выполнения алгоритма) позволило выделить классы задач, названных неразрешимыми (intractable). В отличие от невычислимых функций, для них существуют алгоритмы, работающие за конечное время. Однако ресурсы, необходимые для выполнения этих алгоритмов, растут слишком быстро (например, экспоненциально) с увеличением размера входных данных.

С появлением неразрешимых задач возникла потребность в физических устройствах, вычислительные возможности которых зависели бы экспоненциально от размеров (ресурсов) устройства. В 1981 году нобелевский лауреат по физике Ричард Фейнман предложил построить вычислительное устройство на основе квантовой системы. Оптимизм Фейнмана был основан на

том факте, что простейшая квантовая система намного сложнее простейшей классической системы. Кроме того, линейный рост квантовой системы вызывает экспоненциальный рост сложности ее описания. Например, для классического описания состояния 10-кубитной системы потребуется 1024 комплексных числа, а для 30 кубит – уже более миллиарда комплексных чисел.

Уже в 1985-м году Дэвидом Дойчем была разработана математическая модель универсального квантового компьютера и показаны некоторые ее преимущества перед классической моделью. А в 1994-м Питер Шор взорвал научную общественность, опубликовав полиномиальный квантовый алгоритм разложения составного числа на множители. Результат Шора поставил под угрозу широко используемый алгоритм ассиметричного шифрования RSA.

Таким образом, квантовые вычисления имеют все шансы превзойти современные классические компьютеры сразу по двум напрявлениям – как более совершенный, быстрый и информационно емкий физический процесс, и как более перспективная в теоретическом смысле модель. Репертуар квантового компьютера шире, чем у классической машины Тьюринга (которая не может, например, генерировать случайные числа), а для многих сложных для классических вычислений задач уже существуют эффективные квантовые алгоритмы.

Цель этого курса – познакомить слушателя с моделью квантовых вычислений, структурой и примерами квантовых алгоритмов. Квантовые вычисления – сравнительно новая область науки, и поэтому она может быть интересной для молодых исследователей, желающих работать на самом острие современной научной мысли.

Развитие аппаратной базы квантового компьютера в скором времени может породить спрос на алгоритмистов, способных понимать существующие квантовые алгоритмы и разрабатывать новые. Создание фундамента подобных знаний у слушателей является главной задачей курса.

2 Информация и вычисления

Итак, перейдем к теме курса. Называется курс "Квантовые вычисления", и сначала мы должны разобраться с обоими словами, составляющими это название. Начнем со слова вычисление. Существует довольно много дополняющих друг друга определений этого понятия. Для наших целей подойдет самое общее определение, выделяющее важные для нас характеристики вычислений:

Вычисление – это конечный по времени физический процесс с фиксированным (не обязательно конечным) набором состояний, каждое из которых может быть описано в некоторой кодировке.

Выбор набора состояний и определение способа их описания зависят целиком от пользователя процесса. На одном и том же физическом явлении вы-

числения можно организовать по-разному. Рассмотрим в качестве примера пастуха, не умеющего считать, из популярной книги Дэвида Дойча "Начало бесконечности".

Каждое утро пастух выпускает своих овец из хлева на выпас, а по вечерам загоняет их обратно в хлев. В силу непреодолимых обстоятельств, не являющихся важными для читателя, пастух не умеет считать. Он не знает, сколько у него овец, и, по всей видимости, не считает эту информацию важной. Важно для него только то, чтобы количество овец в стаде оставалось неизменно в течение дня. Недостаток овец вечером необходимо отслеживать и, в случае возникновения этого недостатка, отправляться на поиски заблудившихся животных. Для решения этой задачи пастух использует вычислительное устройство - веревку, помогающую ему отслеживать неизменность количества овец утром и вечером. Утром, при выходе каждой овцы из хлева пастух наматывает один виток веревки на столб, стоящий неподалеку. Вечером, загоняя овцу обратно, он разматывает один виток со столба. Если все витки к концу процесса размотаны, то пастух вправе полагать, что все овцы на месте.

Подобный вычислительный процесс, включающий в себя веревку, столб и пастуха, хорошо демонстрирует данное выше определение.

Во-первых, процесс безусловно основан на физических системах. Все его составляющие присутствуют в наблюдаемой вселенной, а одна из них – пастух – даже преобразовывает в процессе вычислений энергию. Пастух, являясь одновременно частью вычислителя и его пользователем, различает только шесть состояний процесса:

1. Утро. Веревка полностью размотана. (**Начальное состояние)**
2. Утро. Веревка наматывается на столб. (Сохранение количества овец)
3. Вечер. Веревка разматывается со столба. (Считывание количества овец)
4. Вечер. Все овцы зашли, веревка размотана не полностью. (**Конечное состояние.** Недостаток овец)
5. Вечер. Веревка полностью размотана, но овцы зашли не все. (**Конечное состояние.** Избыток овец)
6. Вечер. Все овцы зашли, веревка размотана полностью. (**Конечное состояние.** Все овцы на месте)

Тот же самый фермер, прочитав учебник по арифметике, может существенно обогатить состояниями этот физический процесс, считая каждый отдельный виток веревки, как самостоятельное состояние, несущее информацию о количестве овец. Столб в этом случае становится инструментом "оцифровки" непрерывного параметра – длины веревки.

Пример этот, кроме прочего, показывает, что каждое состояние несет некоторую информацию (в заданной кодировке). Начальное состояние определяет входные данные процесса, конечное – выходные. Это позволит нам определить информацию, как описание состояния некоторой физической системы.

Информация – это описание в некоторой кодировке состояния физической системы (не обязательно выполняющей вычисления).

Подобный материалистический подход позволит нам пойти еще дальше и определить такие понятия как количество информации и количество вычислений.

Количество вычислений – количество смен состояний вычислительного процесса.

Для определения количества информации мы воспользуемся формулой Шеннона, предложенной американским криптоаналитиком Клодом Шенноном в 1948 году. Формула Шеннона определяет информационную емкость системы, имеющей n состояний, для каждого из которых определена его вероятность P_i:

$$I = -\sum_{i=1}^{n} P_i \cdot log_b(P_i)$$

Те, кто помнит школьную физику, заметят несомненное сходство информации в этой формуле с энтропией системы.

Если мы возьмем логарифм по основанию 2 ($b = 2$), то информационная емкость по формуле будет рассчитываться в битах. Бит – минимальная частичка информации, еще одно понятие, подаренное миру Клодом Шенноном.

Для системы, имеющей n равновероятных состояний, количество информации в битах по формуле Шеннона выглядит проще:

$$I = -\sum_{i=1}^{n} \frac{1}{n} \cdot lg\frac{1}{n} = lg(n)$$

Например, для простейшего тумблера (fig. 1), имеющего два равновероятных состояния, мы получаем по формуле ($I = lg2 = 1$) информационную емкость 1 бит. Получается, что такой тумблер может хранить (содержать) один бит информации.

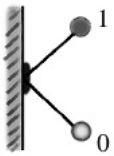

Fig. 1. Переключатель с двумя состояниями.

Если мы добавим тумблеру 3-е состояние, то его информационная емкость увеличится и станет равной $lg3 \approx 1.6$.

Оказывается, информация не только материальна, она еще и в некотором смысле эквивалентна энергии! Рассмотрим мысленный эксперимент, предложенный венгерским физиком Лео Сцилардом в 1929-м году.

В этом эксперименте рассматривается камера, ограниченная с двух сторон подвижными невесомыми поршнями и разделенная перегородкой. Перегородку можно убирать (fig. 2).

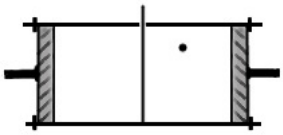

Fig. 2. Двигатель Сциларда.

В камере находится одна молекула идеального газа температуры T. В этой системе мы будем различать два состояния – молекула находится слева от перегородки (назовем это состояние 0) и молекула находится справа от перегородки – состояние 1. Информационная емкость по формуле Шеннона – 1 бит.

Допустим, что мы владеем этим битом информации, то есть мы знаем, с какой стороны от перегородки находится молекула. Тогда, мы можем пододвинуть невесомый поршень с другой стороны прямо к перегородке (fig. 3) и вынуть перегородку.

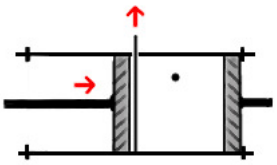

Fig. 3. Двигатель Сциларда. Стирание бита.

Идеальный газ (наша молекула), согласно законам термодинамики, расширится, толкая поршень обратно и совершит работу, равную $k_B \cdot T \cdot ln2$, где k_B – постоянная Больцмана, T – температура газа, а $ln2$ – логарифм отношения объемов газа, конечного и начального. Таким образом, владея информацией, мы заставили систему совершить работу!

Операция, которую мы совершили, называется Erase – стирание. Мы стерли один бит информации и получили работу ($I \to A$). Обратная операция называется Assign – присвоение.

Предположим, что мы хотим присвоить нашей системе, находящейся в неизвестном состоянии, например, значение 1. Тогда мы должны сжать идеальный газ левым поршнем до половины объема, затратив при этом ту же самую работу, которую мы получили при стирании бита. После этого мы вставляем перегородку и убираем обратно левый поршень. Работа (энергия) преобразовалась в информацию ($A \to I$).

Рассмотрим теперь простейший вычислительный гейт NOT, преобразовывающий 0 в 1, а 1 в 0. Применение этого гейта к нашему текущему состоянию можно выполнить следующим образом:

1. Пододвинуть левый поршень к перегородке ($A = 0$).
2. Убрать перегородку.
3. Одновременно, без изменения объема, передвинуть оба поршня влево ($A = 0$). (fig. 4)
4. Поставить перегородку и вернуть на место правый поршень.

Выполнение гейта NOT оказалось не связано с преобразованием энергии, так как количество информации в системе не изменилось.

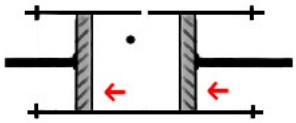

Fig. 4. Двигатель Сциларда. Гейт NOT.

Получается, что обратимые преобразования (вроде гейта NOT) в идеале не требуют от нас затрат энергии. Забегая вперед, отметим, что все преобразования в квантовых вычислениях обратимы, и значит, теоретически, квантовые вычисления можно производить без затрат энергии. Конечно, современное состояние области далеко от этого идеала.

3 Характеристики вычислительной системы

Подошло время нам перейти к описанию характеристик вычислительных систем, чтобы мы могли сравнивать эти системы друг с другом.

Мы договорились, что вычисление – это всегда физический процесс. Но физические процессы бывают разные. Нам нужно выделить какие-нибудь важные характеристики, которые позволят нам сравнивать эти процессы и предпочитать один другому для организации вычислений.

Среди наиболее важных характеристик выделим:

1. Информационную емкость процесса (по Шеннону).
2. Инерционность, или скорость смены состояний.
3. Универсальность.

Назовем (неформально) репертуаром вычислителя все множество задач, которые он может решать.

Информационная емкость влияет на репертуар вычислительного процесса (например, с помощью одного тумблера с двумя состояниями нельзя считать овец, если их больше одной.)

Инерционность или скорость смены состояний влияет на потребительские характеристики вычислителя и, в практическом смысле, так же на репертуар. Всего за несколько десятилетий скорость смены состояний в ЭВМ выросла в миллиарды раз, и многие современные задачи были бы невычислимы в практическом смысле для первых ЭВМ, так как для их решения потребовались бы миллионы лет.

Универсальность – характеристика, стоящая немного в стороне от первых двух, непосредственно задает репертуар вычислительного процесса.

Вспомним пастуха, использующего веревку для контроля неизменности количества овец в стаде. У веревки есть непрерывный параметр – ее длина. С помощью веревки пастух может, например, запоминать и сравнивать длины предметов, измеряя их веревкой и делая отметки. Непрерывность параметра очень удобна для этой задачи, так как не возникает проблем соизмеримости сравниваемых предметов и "делений" измерителя.

Веревка без делений может служить и для сравнения количеств. Вместо наматывания веревки на столб пастух мог бы просто откладывать от мотка кусок, равный, например, длине своей руки, при выходе овец из хлева. "Подсчет" овец при обратном процессе заключался бы в обратной операции – сматывании веревки в моток на одну длину руки при возвращении каждой овцы. При таком подходе у пастуха нередко возникали бы ситуации с накоплением ошибки. Считать ли конечным состояние, в котором несмотанным оказался кусок веревки меньший, чем длина руки фермера? Какое решение необходимо принять в такой ситуации?

Фермер может «оцифровать» веревку, завязав на ней узелки. Таким образом он сразу решит проблему накопления ошибки. Вообще, процесс оцифровки часто служит именно для цели контроля и исправления ошибок. Кроме того, оцифрованная веревка более универсальна в задачах подсчета чего бы то ни было. Наличие узелков увеличивает ее репертуар. С помощью такой веревки можно складывать, вычитать и даже умножать и делить с остатком.

Универсальность – свойство цифровых, дискретных вычислений. И именно такие вычисления мы будем рассматривать в дальнейшем.

Оставив пока в стороне универсальность, сосредоточимся на первых двух характеристиках – емкости и скорости. Эволюция вычислительных устройств, известная нам как "поколения ЭВМ" – это последовательный подбор все

более быстрых и емких физических процессов, лежащих в основе этих устройств.

Инженеры идут по пути миниатюризации базового элемента вычислительного процесса, начиная от сравнительно больших и инертных радиоламп и кончая технологией упаковки огромного количества p-n переходов в одном квадратном сантиметре кристалла кремния. Нам нужны базовые элементы все меньшего размера. Почему?

1. Для того, чтобы увеличить емкость при снижении энергопотребления.
2. Для того, чтобы уменьшить инерционность. Чем меньше элемент, тем меньше его, например, индуктивность или масса, тем выше скорость переключения состояний.

В настоящее время основным базовым элементом вычислений является p-n переход. Современные нам технологии производства процессоров приближаются к отметке 10нм, что всего в 200 раз больше атома водорода! Понятно, что у этого пути есть конец, и каждый новый шаг в его направлении дается все с большим трудом. На отдельном атоме p-n переход уже не организовать, а значит, от технологии транзисторов придется отказаться. Кроме того, при таких размерах базового элемента уже нельзя пренебрегать квантовыми эффектами. Получается, что в том или ином виде, квантовый компьютер неизбежен!

4 Вычислимость и алгоритм

Физические ограничения реального мира – не единственное препятствие к безграничному росту наших вычислительных возможностей. Чтобы разобраться с этим вопросом, определим цель вычислений.

Вычисления всегда реализовывают алгоритмически (если они цифровые) некоторую функцию. Любой алгоритм имеет множество (иногда пустое) входных данных – параметры функции, и выходные данные для каждого набора входных. Иными словами, алгоритм является отображением из множества входных данных в множество выходных. Данные являются описанием состояния физической системы в выбранной нами кодировке. Если мы выберем числовое представление (например, двоичное), то можно утверждать, что любой алгоритм является отображением множества натуральных чисел в множество натуральных чисел:

$$A : \mathbb{N} \to \mathbb{N}$$

Можем ли мы утверждать, что верно и обратное утверждение – любая функция представима в виде алгоритма?

Для того, чтобы ответить на этот вопрос, нам необходимо строгое математическое определение понятия "алгоритм".

Наиболее известным и общепринятым определением считается модель устройства, предложенная в 1936-м году английским ученым Аланом Тьюрингом. Машина Тьюринга M – это упорядоченная шестерка:

$$M = \{Q, \Gamma, b, \delta, q_0, F\} \tag{1}$$

где

- Q – конечное непустое множество состояний,
- Γ – конечный непустой алфавит,
- $b \in \Gamma$ – символ алфавита, для которого допустимо появление на ленте бесконечное число раз,
- $\delta : Q \setminus F \times \Gamma \to Q \times \Gamma \times \{\leftarrow, \rightarrow\}$ – набор правил,
- $q_0 \in Q$ – начальное состояние,
- $F \subset Q$ – множество конечных состояний.

Машина Тьюринга представляет собой бесконечную ленту, разделенную на ячейки, в каждой из которых записан один символ алфавита Γ, и читающую / пишущую головку, расположенную на каждом шаге над некоторой ячейкой ленты.

Машина начинает работу в начальном состоянии. На каждом шаге головка считывает символ, записанный на ленте. После этого из множества правил выбирается то, которое соответствует текущему состоянию и прочитанному символу. Правило определяет, какой символ нужно записать на ленту, в какое перейти состояние и куда (влево или вправо) передвинуть головку.

Работа заканчивается в одном из конечных состояний либо не заканчивается никогда (машина входит в бесконечный цикл).

Репертуаром машины Тьюринга назовем все множество функций, для которых существует конкретный набор вида (1), который для любых входных данных функции, действуя по правилам машины Тьюринга приходит в конечное состояние, написав на ленте значение функции. Функции, вычисление которых входит в репертуар машины Тьюринга, называются вычислимыми.

Тогда же, в 1936-м году, независимо Алонзо Черчем и Аланом Тьюрингом был сформулирован вопрос – возможен ли цифровой вычислительный процесс с большим, чем у машины Тьюринга репертуаром. Ответ на этот вопрос известен как тезис Черча-Тьюринга, и этот ответ отрицателен. Одна из множества эквивалентных формулировок тезиса звучит так: **все алгоритмически вычислимые функции входят в репертуар машины Тьюринга.** В этой формулировке термин "алгоритмически вычислимые" подразумевает интуитивное понинмание алгоритма, предполагающее механическое выполнение конечного числа шагов для достижения детерминированного (одинакового для одного и того же набора входных данных) результата. На сегодняшний момент тезис не доказан и не опровергнут. Если предположить, что тезис верен, то получится, что даже вычислительные возможности человека подчиняются тем же правилам и ограничениям, что и машина Тьюринга.

Что же это за ограничения?

Давайте посчитаем, сколько всего различных машин Тьюринга.

Каждая машина Тьюринга может быть закодирована строкой из нулей и единиц конечной длинны, что соответствует двоичному представлению некоторого натурального числа. Так же, мы можем интерпретировать любое достаточно большое натуральное число как закодированное представление машины Тьюринга. Множество всех натуральных чисел счетно, следовательно, множество различных машин Тьюринга тоже счетно.

А сколько всего функций вида $f : \mathbb{N} \to \mathbb{N}$?

Для ответа на этот вопрос давайте рассмотрим произвольное число $a \in [0, 1]$. Запишем это число, например, в двоичной системе счисления. Получится бесконечная строка из нулей и единиц вида $a = 0.a_1a_2a_3 \ldots a_k \ldots$.

Теперь определим функцию f_a:

$$f_a(n) = a_n$$

Эта функция определена на всем натуральном ряде и возвращает 1 бит ($f_a : \mathbb{N} \to \{0, 1\}$). Очевидно, что разные значения a задают таким образом разные функции такого вида. Разных чисел на отрезке [0,1] континуум, а значит, и разных функций из $\mathbb{N}$ в $\mathbb{N}$ – минимум континуум (на самом деле – ровно континуум, поскольку подобным образом можно каждой такой функции сопоставить ровно одно вещественное число).

Таким образом, получается, что функций больше, чем алгоритмов! Причем не просто больше. Функции, которые можно вычислять на машине Тьюринга имеет меру нуль на общем множестве функций! В этом богатом и разнообразном мире мы не можем вычислить практически ничего!

Но, может быть и не надо? Может быть все эти невычислимые функции нам совсем не интересны? Оказывается нет. В той же работе, в которой Тьюринг определил математическую модель вычислений, он привел пример полезной невычислимой функции. Функция эта получила название «Проблема останова» (Halting problem).

Тьюринг показал, что не существует такого алгоритма A, который бы предсказал поведение всех возможных алгоритмов, например, таким образом:

$$A(x, y) = \begin{bmatrix} 0, \text{X(y) halts} \\ 1, \text{X(y) hangs} \end{bmatrix}$$

А принимает на вход двоичное представление алгоритма x и входные данные к нему – y и возвращает 1, если X на этих входных данных зависает, и 0, если нет.

Для доказательства того, что такого алгоритма не существует, Тьюринг воспользовался модификацией диагонального аргумента Кантора:

Предположим, что алгоритм A существует. Тогда мы можем определить алгоритм $\hat{A}$ - анализатор, действующий следующим образом:

$$\hat{A}(x, y) = \begin{bmatrix} 0, & \text{X(y) hangs} \\ \text{hang}, & \text{X(y) halts} \end{bmatrix}$$

Ясно, что имея алгоритм A, алгоритм $\hat{A}$ реализовать нетрудно. Теперь определим алгоритм D – диагонализатор, который вызывает анализатор:

$$D(x) = \hat{A}(x, x)$$

Посмотрим, что получится, если мы вызовем Диагонализатор от двоичного представления самого диагонализатора:

$$D(d) = \hat{A}(d, d) = \begin{bmatrix} 0, & \text{D(d) hangs} \\ \text{hang}, & \text{D(d) halts} \end{bmatrix}$$

Получается, что $D(d)$ возвращает 0, если $D(d)$ зависает. Это противоречие указывает на ошибочность самой первой предпосылки – существование алгоритма A.

5 Сложность вычислений

Существование невычислимых функций накладывает серьезные ограничения на классическую модель вычислений. Но и для тех функций, которые являются формально вычислимыми для машины Тьюринга, в практическом плане существуют проблемы.

Дело в том, что вычислимые функции в модели Тьюринга имеют важную характеристику – сложность вычислений. Строго говоря, сложность вычислений – это характеристика не функции, а алгоритма, ее вычисляющего. Поскольку для одной и той же функции существует бесконечно много вычисляющих ее алгоритмов, нам необходимо выбрать, сложность какого из этих алгоритмов мы будем считать сложностью функции.

Мы не будем формально определять сложность вычислений и классы сложности, поскольку это – тема отдельного курса. Неформально, сложность алгоритма – это характер роста какого-то ресурса (например, времени), необходимого для его выполнения, в зависимости от размера задачи.

Для определения сложности задачи (функции) было бы естественно выбрать сложность наилучшего вычисляющего ее алгоритма. Но, выбор наилучшего элемента в бесконечном множестве не всегда осуществим. Рассмотрим, например, последовательность чисел $\{a_n\}_{n=1}^{\infty}$, $a_n = \frac{1}{n}$. В этой последовательности нет наименьшего элемента. Точно так же, теоретически, возможны задачи, для которых нет наилучшего алгоритма. Однако, чаще всего, препятствием для определения сложности задачи является то обстоятельство, что лучший алгоритм неизвестен.

Приведем несколько примеров.

При сложении чисел в столбик, мы выполняем вдвое больше элементарных (табличных) операций сложения, чем разрядов (цифр) в складываемых числах. Если n – количество цифр, то сложность сложения в столбик – $O(n)$.

При сортировке массива из n элементов методом пузырька мы выполняем $O(n^2)$ операций сравнения. Сортировка массива – очень хорошая функция. Для нее нам известен теоретический предел сложности – $O(n \cdot ln(n))$, и

алгоритмы, реализующие этот предел. Таким образом, у задачи сортировки есть сложность, и она известна.

Задачи такого рода, для которых сложность оценивается некоторым полиномом от размера входных данных, считаются "хорошими". Все вместе они составляют класс P – бесконечный класс задач, для которых существуют полиномиальные алгоритмы (fig. 5).

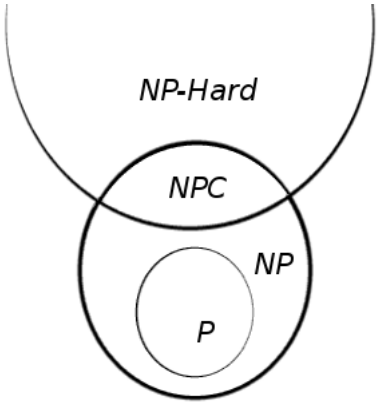

Fig. 5. Диаграмма классов сложности при предположении, что $P \neq NP$.

Есть еще класс "условно хороших" задач – класс NP. Для задач этого класса существуют полиномиальные алгоритмы проверки решения. К таким задачам, например, относятся:

1. Поиск гамильтонова пути в графе
2. Разложение чисел на множители
3. И.. сложение в столбик!

Все задачи из класса P входят в класс NP ($P \subset NP$), так как мы можем проверить решение, просто решив задачу. Если у нас для этого есть полиномиальный алгоритм, то он и будет процедурой проверки решения.

Обратное включение ($NP \subset P$) – один из самых важных нерешенных вопросов современной теории вычислений. На настоящий момент существуют важные задачи из класса NP, для которых неизвестен полиномиальный алгоритм.

Например, разложение числа на множители (факторизация). Представим, что некоторое число N является произведением двух больших простых множителей – p и q ($N = p \cdot q$). Если простые числа p и q нам даны, то мы можем "легко" (полиномиально по времени) их перемножить и посмотреть, получится ли число N. Если же эти числа неизвестны, то в настоящий момент для классических вычислений не существует эффективной процедуры их получения.

Представьте, что вы придумали два разных огромных простых числа p и q, в каждом порядка 10^4 знаков. Эффективная процедура поиска больших простых чисел существует. Вы перемножили эти числа, сохранили (или даже опубликовали) их произведение N, а потом забыли, потеряли p и q. Теперь даже за все время жизни вселенной эти числа невозможно восстановить. Даже, если искать их будет компьютер, размером со всю вселенную, при том, что произведение у всех на виду!

Если гипотеза $P \neq NP$ верна, то класс NP на (fig. 5) больше класса P, а задача факторизации, вероятно, присутствует в их разности $NP \setminus P$.

В 1994-м году американский математик Питер Шор предложил полиномиальный алгоритм для квантового компьютера, позволяющий раскладывать числа на множители. И невыполнимая для классических вычислений задача стала легко выполнимой для квантовых.

На (fig. 5) присутствует так же класс NP-трудных задач. Это такие задачи, к которым полиномиально сводятся все задачи из класса NP. Иными словами, если у нас есть волшебный ящик, умеющий решать какую-то задачу из этого класса, то с помощью этого ящика мы сможем эффективно решать любую задачу из NP. В 1972-м году Стивен Кук показал, что пересечение классов $NP-Hard$ с NP не пусто. Задачи из этого пересечения называются NP-полными или $NP-Complete$, NPC.

Для всех задач из NP квантовые вычисления в общем виде дают лучший результат, чем известные в настоящее время классические алгоритмы. К сожалению, этого нельзя сказать про класс NP-трудных задач. Ближе к концу курса мы разберем одну из этих задач и покажем, что квантовые вычисления для нее дают не лучший результат, чем классические.

6 Квантовые вычисления

С одной стороны, уменьшение базового элемента в вычислительном процессе приводит к тому, что мы уже не можем пренебрегать квантовыми эффектами. С другой стороны, для многих полезных задач, время их решения растет экспоненциально от размера задачи. Одной из таких задач является моделирование квантовых систем.

В 1982 году нобелевский лауреат по физике Ричард Фейнман предложил убить сразу двух зайцев, реализовав вычислительный процесс на базе квантовой системы. Таким образом, мы смогли бы с пользой эксплуатировать "вредные" квантовые эффекты, и получить вычислители, мощность которых растет экспоненциально с ростом количества базовых элементов.

Уже через три года (1985 г.) англичанин Дэвид Дойч предложил математическую модель квантовых вычислений и рассмотрел простую задачу (известную как "задача Дойча"), для которой он продемонстрировал преимущества квантовых вычислений над классическими.

За этой работой последовало несколько довольно экзотических результатов и, наконец, в 1994-м году был опубликован алгоритм Питера Шора,

показавший, что некоторые пока не разрешимые в практическом смысле задачи могут иметь эффективное решение на квантовом компьютере.

В 1996-м году Лов Гровер предложил квантовый алгоритм, решающий любую задачу из класса NP в общем виде с квадратичным по сравнением с классической моделью ускорением.

Преимущества квантовых вычислений над классическими стали очевидны сразу по двум направлениям – с точки зрения миниатюризации базового элемента вычислительного процесса, и с точки зрения теоретических преимуществ модели.

Настало время и нам перейти от слова "вычисления" в названии курса к слову "квантовые", и разобраться, сначала, с квантовыми эффектами, о которых мы упоминали ранее.

"Эксперимент" с лягушкой.
Представим себе лягушку, смотрящую на включенный и направленный на нее фонарик. Лягушка видит фонарик, как яркое пятно света. Мы начинаем удалять лягушку от фонарика. Что видит лягушка?

1. Пятно становится меньше.
2. Яркость пятна падает.

Пропадет-ли пятно совсем? И да и нет! Начиная с некоторого момента точка света, наблюдаемая лягушкой, начнет мигать. И начиная с этого момента, яркость вспышек будет неизменной (у лягушки очень чувствительный глаз – она будет их видеть). Но вспышки эти будут происходить все реже по мере отдаления. Реже, но не прекратятся никогда.

Эксперимент этот показывает, что свет не "размазывается" по поверхности распространения бесконечно тонким слоем и, по всей видимости, состоит из частичек, атомов света – фотонов. Именно эти частички воспринимает своим чувствительным глазом лягушка. Сначала их много, и лягушка видит яркое пятно, но по мере ее отдаления от источника, все меньшая доля фотонов устремляется точно в ее направлении.

Разумеется, ни одной лягушки при проведении подобного эксперимента не пострадало. Вместо лягушек использовали высокочувствительную пленку, а вместо расстояния – затемняющие фильтры.

Опыт Юнга
Итак, свет состоит из частичек – атомов света, фотонов. Рассмотрим следующий эксперимент:

Вертикально, друг напротив друга расположены два экрана, в первом из которых прорезаны две щели (fig. 6). В некоторой точке напротив первого экрана, строго между щелями находится пушка, стреляющая в сторону экранов картечью.

Большая часть картечи попадет в первый экран, но будет маленькая часть, которая пройдет сквозь щели и долетит до второго экрана (fig. 7). Картинка, нарисованная картечью на втором экране, будет по форме напоминать щели в первом. Это – своеобразное рисование по трафарету.

Fig. 6. Эксперимент с двумя щелями. Установка.

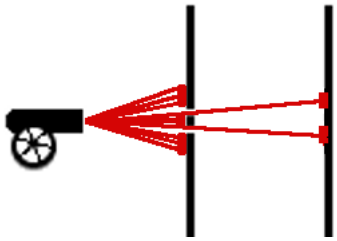

Fig. 7. Эксперимент с двумя щелями. Картечь.

Изменим установку. Экраны остаются, но теперь они поставлены в воду, а в точке, где была раньше пушка, теперь происходят колебания, создающие на поверхности воды волны (fig. 8).

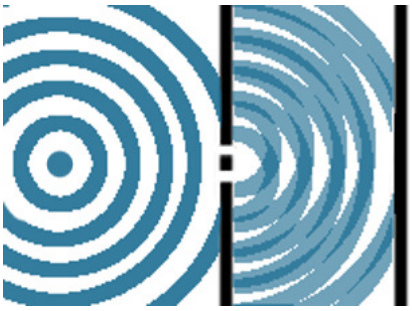

Fig. 8. Эксперимент с двумя щелями. Волны.

В этом случае щели в первом экране становятся вторичными источниками волн, и на экране мы видим уже не образ двух щелей, а интерференционную картину этих волн. Например, в точку, находящуюся строго посередине между щелями, волны прибывают в одинаковой фазе, и их амплитуды складываются. Но есть также и точки, в которые волны приходят в противофазе, и амплитуда в этих точках – 0.

Какую картину мы увидим, если в первой установке мы вместо пуль будем выпускать фотоны? Мы только что убедились, не без помощи лягушки, что свет состоит из неделимых частичек – фотонов, и, следовательно, следует ожидать ситуацию рисования по трафарету.

Не тут-то было! Еще в 1803 году английский физик Томас Юнг показал, что в этом эксперименте свет ведет себя как волна, и вместо рисования по трафарету мы видим интерференционную картину! Более того, интерференционная картина возникает даже если мы выпускаем фотоны по одному в час!

Если мы закрываем одну из щелей, мы видим образ второй щели – рисование по трафарету. Если же мы открываем вторую щель, в тех местах, где раньше был свет, появляются тени интерференционной картины. Мы добавили света, а получили тень!

Если же мы будем "подглядывать" за фотонами, пытаясь определить, через какую щель они прошли, то интерференционная картина немедленно исчезает. Подглядывать можно, например, по разному изменяя поляризацию фотонов в разных щелях. Фотоны не любят, когда за ними подглядывают. Пока мы на них не смотрим, они как-бы проходят сразу через две щели, но стоит нам установить наблюдение – они тут же становятся нормальными классическими частицами, проходящими ровно через одну щель.

И за подобным поведением замечены не только фотоны. Так же ведут себя и многие маленькие частички, например электроны. Для маленьких частиц нашего мира свойственно находится сразу в двух состояниях. Сразу на двух орбитах атома, или сразу в двух щелях интерферометра Юнга. Или одновременно иметь разный спин. Очень неудобное обстоятельство для организации классических вычислений на подобных элементах!

7 Многомировая интерпретация квантовой механики

Было бы жестоко познакомить вас с этими чудесами совсем без объяснения.

В квантовой механике разработан строгий математический аппарат, позволяющий описывать все перечисленные явления. Согласно этой теории, положение фотона или электрона описывается не его координатой, а некоторой функцией вероятности, "размазанной" по определенной области пространства – волновой функцией. Наблюдение частицы вызывает коллапс этой волновой функции. Иными словами, в эксперименте с двумя щелями вероятности прохождения через каждую из щелей равны, а значит и фотон проходит сразу через обе щели. Только это не совсем фотон, а его волновая функция, которая моментально становится фотоном, если мы проведем измерение.

В таком виде объяснение представляется не слишком понятным, да и вообще не является объяснением. Это скорее описание.

Первое понятное и математически строгое объяснение квантовых эффектов дал в своей диссертации в 1956 г. американский ученый Хью Эверетт III.

Вспомним всеми любимый мысленный эксперимент с котом Шредингера. Кот заперт в ящике с адской машиной, поведение которой зависит от микроскопической частицы – например, от пути фотона в интерферометре Юнга (fig. 9).

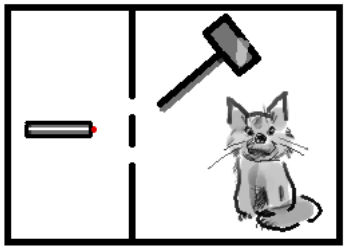

Fig. 9. Кот Шредингера.

Этот путь (состояние электрона) описывается суперпозицией некоторых базисных состояний. Например – верхняя и нижняя щели, с равными вероятностями ($|\uparrow\rangle + |\downarrow\rangle$). Если электрон проходит через верхнюю щель, адская машина приходит в действие, и кот умирает. Через нижнюю щель – кот остается жив. В некотором смысле, кот первым проводит процесс измерения.

Проблема в том, что до того как мы измерили (пронаблюдали) кота, система в ящике для наблюдателя не подвергалась измерению, и, согласно законам квантовой механики должна была развиваться линейно и унитарно (fig. 10). Коллапса волновой функции этой системы не произошло, и, следовательно, она находится в состоянии

$$(|\uparrow\rangle |Cat = Dead\rangle + |\downarrow\rangle |Cat = Alive\rangle) |H1\rangle .$$

Здесь $H1$ – волновая функция наблюдателя 1. Получается, что для каждого из исходов мы получаем реально существующего, соответствующего этому исходу кота.

Добавим на картинку второго наблюдателя (fig. 11). Пусть H1, находясь в закрытой комнате, в 12-00 открыл коробку и посмотрел на кота. Тем самым он сформировал некоторый субъективный чувственный опыт, осознав, например, что кот мертв. Система для него коллапсировала в состояние

$$|\uparrow\rangle |Cat = Dead\rangle |H1 = SAD\rangle$$

Но для $H2$ измерение комнаты с $H1$ еще не происходило. Он знает, что в 12-00 $H1$ откроет коробку, но волновая функция всей комнаты коллапсировать не должна. Для $H2$ эта волновая функция выглядит так:

$$|\uparrow\rangle |Cat = Dead\rangle |H1 = SAD\rangle + |\downarrow\rangle |Cat = Alive\rangle |H1 = GLAD\rangle$$

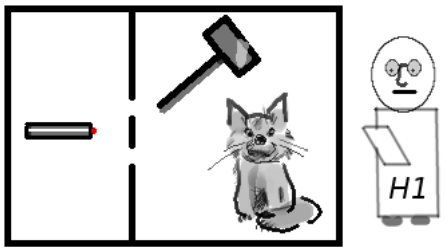

Fig. 10. Кот Шредингера и Наблюдатель 1.

Получается, что когда субъективно $H1$ расстроился из-за потери кота, объективно существует второй $H1$, который увидел кота живым, и для H2 оба они реальны!

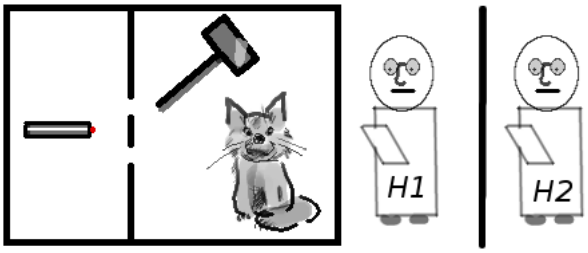

Fig. 11. Кот Шредингера, Наблюдатели 1 и 2.

Где же существуют эти реальные копии наблюдателей? Для того, чтобы разобраться с этим вопросом, представим эту ситуацию в пространстве-времени.

В пространстве-времени есть "снимок" вселенной в момент t_0, когда фотон испускается в направлении интерферометра Юнга (fig. 12). Далее, для момента t_1 физически возможно существование "снимков", на которых фотон проходит через верхнюю или нижнюю щель (fig. 13). Для каждого из этих "снимков" предыдущим по времени является "снимок" 1. И Эверетт предположил, что раз оба эти снимка физически возможны и нет никакого способа предпочесть один другому, то оба они существуют в пространстве-времени!

Таким образом, наша вселенная, согласно интерпретации Эвереттa, сложнее, чем кажется. В ней могут существовать копии моментов времени, содержащих различные варианты одних и тех же событий. Для каждой точки во

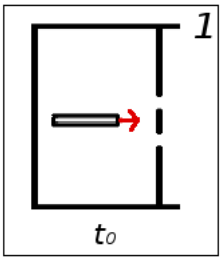

Fig. 12. Снимок 1. Фотон направлен в интерферометр.

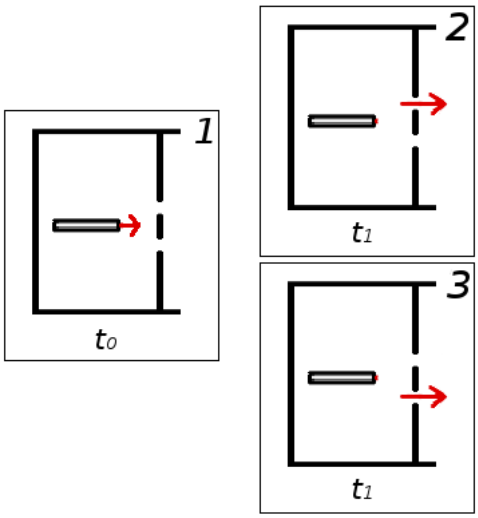

Fig. 13. Снимки 2 и 3. Фотон проходит через верхнюю и через нижнюю щели.

времени существует множество различных потомков этой точки, и обитатели "параллельных" снимков не могут воспринимать соседние снимки. Но если параллельные снимки в пространстве-времени не могут взаимодействовать, то откуда же берется интерференционная картина?

Рассмотрим темную полосу в интерференционной картине (fig. 14). В точки этой полосы фотоны из разных щелей интерферометра приходят в противофазе. При этом, при попадании фотона в экран уже нельзя сказать, из какой щели он пришел (и какой из снимков – 2 или 3 был его предшественником). Получается, что оба снимка являются предысторией снимка 4, но этого не может быть, поскольку в этом случае фотон должен находится в противофазе с самим собой. Это противоречит здравому смыслу, а значит и снимка 4, в котором фотон приходит в точку черной полосы в пространстве-времени не существует.

Если же в точке экрана мы можем сказать, откуда пришел фотон, например запутав его позицию в интерферометре с поляризацией, то противоречие исчезает, и снимок 4 имеет право на существование, что приводит к исчезновению интерференционной картины.

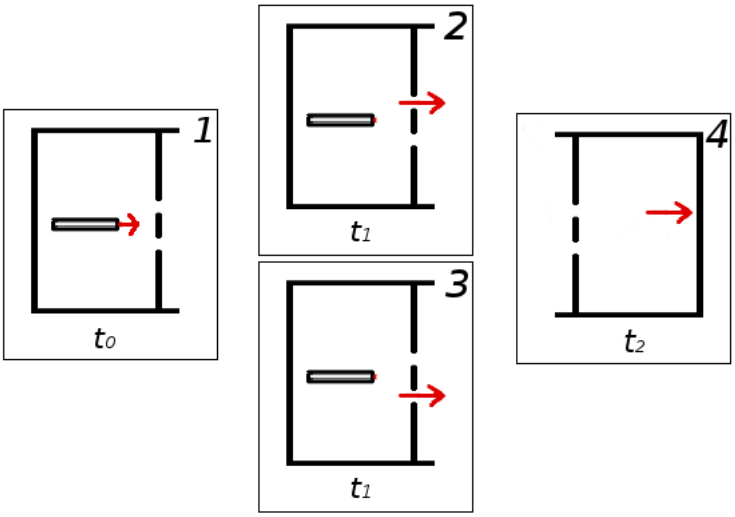

Fig. 14. Снимки 2 и 3. Фотон попадает в "черную" полосу на экране.

Объяснение Эверетта – это просто объяснение, не меняющее математического аппарата теории. С точки зрения математической модели не важно, как мы понимаем коэффициенты в состоянии $\frac{1}{\sqrt{3}}|0\rangle + \frac{\sqrt{2}}{\sqrt{3}}|1\rangle$ – как параметры волновой функции или как доли вселенных, в которых реализовано каждое из базовых состояний. Но подобное объяснение позволяет иногда лучше понять тот математический аппарат, которым нам предстоит пользоваться дальше.

8 Дополнительные материалы

1. **Michael R. Garey, David S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness.** W.H. Freeman and Co. (1979). ISBN 0-7167-1045-5
2. **Дэвид Дойч. Структура реальности. Наука о параллельных вселенных.** Альпина нон-фикшн, Москва (2015). David Deutch. The Fabric of Reality. The Science of Parallel Universes and its Applications. Penguin Books. (1997). ISBN 978-5-91671-693-1, 978-5-91671-346-6
3. **Дэвид Дойч. Начало бесконечности. Объяснения, которые меняют мир.** Альпина нон-фикшн, Москва (2015). David Deutch. The Beginning of Infinity. Penguin Books. (2011). ISBN 978-0-7139-9274-8