

Квантовые вычисления и функциональное программирование

**Душкин Р. В.
roman.dushkin@gmail.com**

Москва, 2014

УДК 004.82 + 004.832.34
ББК 32.81
Д86

Душкин Р. В.

Д86 Квантовые вычисления и функциональное программирование. —
2014. — 318 с., ил.

В этой небольшой книге рассматриваются вопросы наиболее перспективной направления исследований в современных рамках информационно-коммуникационных технологиях — модели квантовых вычислений. Текст построен как можно более просто — главной задачей автор поставил для себя возможность чтения книги без наличия специальных знаний по квантовой механике и прочим подобным наукам, наполненным математическим анализом. Однако в силу определённых обстоятельств в качестве языка программирования, при помощи которого иллюстрируются многочисленные примеры, выбран функциональный язык Haskell, поэтому читатель должен владеть этим языком для полноценного чтения книги.

Книга будет интересна всякому, кто живо следит за новыми веяниями в области теории вычислений и смежных наук.

УДК 004.82 + 004.832.34
ББК 32.81

Все права защищены. Любая часть этой книги не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами без письменного разрешения владельцев авторских прав.

Материал, изложенный в данной книге, многократно проверен. Но, поскольку вероятность технических ошибок всё равно существует, издательство не может гарантировать абсолютную точность и правильность приводимых сведений. В связи с этим издательство не несёт ответственности за возможные ошибки, связанные с использованием книги.

© Душкин Р. В., 2014

Принимаются благодарности

Коллеги и симпатизанты! При подготовке этой книги было затрачено очень много времени и других ресурсов, материальных и нематериальных. Я стоял и крепко стою на позиции о том, что информация и знания должны быть свободны. Однако вкладываться на общественных началах в распространение знаний и повышение общего уровня квалификации наших соотечественников становится всё сложнее и сложнее.

Эта книга всё так же остаётся бесплатной для скачивания и распространения. Но здесь хотелось бы настоять на *обязательности* перечисления благодарности за этот материал, уникальный во всех отношениях для российского книжного рынка и электронного рынка информации. Ваша благодарность в обязательном порядке ожидается по следующим реквизитам:

- Яндекс.Деньги: **4100137733052**
- WebMoney: **R211895623295**
- PayPal: **darkus.14@gmail.com**

Следует напомнить, что на счета в этих платёжных системах благодарность можно перечислить и при помощи терминалов мгновенной оплаты.

Также лиц, заинтересованных в сотрудничестве по вопросам издания, распространения, написания новых книг и т. д., прошу обращаться по адресу **электронной почты** **roman.dushkin@gmail.com**.

*Книга посвящается всем людям
доброй воли с открытым разумом,
естественнонаучным мировоззрени-
ем и системным мышлением.*

Содержание

ВВЕДЕНИЕ	7
ГЛАВА 1. НАИВНОЕ ПОНИМАНИЕ МОДЕЛИ КВАНТОВЫХ ВЫЧИСЛЕНИЙ	14
КВАНТОВЫЕ СОСТОЯНИЯ И КУБИТЫ	18
НЕСКОЛЬКО КУБИТОВ	31
ГЕЙТЫ И КВАНТОВЫЕ СХЕМЫ	37
КВАНТОВАЯ СХЕМОТЕХНИКА	58
ПРИНЦИПЫ КВАНТОВЫХ ВЫЧИСЛЕНИЙ	65
ОБЩАЯ АРХИТЕКТУРА КВАНТОВОГО КОМПЬЮТЕРА	70
КРАТКИЕ ВЫВОДЫ	72
ГЛАВА 2. ФРЕЙМВОРК ДЛЯ КВАНТОВЫХ ВЫЧИСЛЕНИЙ	74
КВАНТОВЫЕ СОСТОЯНИЯ	75
КУБИТЫ	80
ГЕЙТЫ	90
КВАНТОВЫЕ ВЫЧИСЛИТЕЛЬНЫЕ СХЕМЫ	97
НЕКОТОРЫЕ ЗАДАЧИ И ИХ РЕШЕНИЕ	100
КРАТКИЕ ВЫВОДЫ	108
ГЛАВА 3. ЯЗЫК ПРОГРАММИРОВАНИЯ QUIPPER	111
НЕМНОГО О ЯЗЫКЕ QCL	114
ВВЕДЕНИЕ В ЯЗЫК QUIPPER	119
РЕШЕНИЕ НЕСКОЛЬКИХ ПРОСТЫХ ЗАДАЧ	121
ОТ ПРОСТОГО К СЛОЖНОМУ	131

Дополнительные возможности	142
Краткие выводы	145
ГЛАВА 4. ДЕТАЛЬНОЕ РАССМОТРЕНИЕ НЕКОТОРЫХ КВАНТОВЫХ АЛГОРИТМОВ	148
Алгоритм Гровера	150
Алгоритмы Дойча и Дойча-Йожи	163
Алгоритм Саймона	171
Алгоритм Шора	178
Краткие выводы	197
ГЛАВА 5. КВАНТОВЫЙ «ЗООПАРК»	201
Алгебраические и теоретико-числовые алгоритмы	203
Алгоритмы со специальными оракулами	212
Алгоритмы аппроксимации и эмуляции	250
Краткие выводы	265
ОБЗОР ЛИТЕРАТУРЫ О КВАНТОВЫХ ВЫЧИСЛЕНИЯХ	269
ОБЗОР ВИДЕОКУРСОВ ПО КВАНТОВЫМ ВЫЧИСЛЕНИЯМ И СМЕЖНЫМ ТЕМАМ	304
ЗАКЛЮЧЕНИЕ	313

Введение

Сегодня всё больше и больше теоретических работ и даже практических разработок появляется в области так называемых квантовых вычислений, то есть новой вычислительной модели, которая вместо давно известного понятия бита (который, в свою очередь, пришёл к нам из работ Алана Тьюринга и Джона фон Неймана), основана на понятии «кубита», то есть квантового бита.

Квантовый бит — это некая квантовая система, которая до измерения находится в произвольной линейной суперпозиции двух базисных квантовых состояний (то есть, по сути, может принимать бесконечно большое разнообразие возможных значений), а в результате измерения с той или иной вероятностью принимает одно из двух возможных значений. Поэтому он и называется битом, поскольку возможных значений два, но с другой стороны он квантовый, поскольку эти два значения находятся в суперпозиции друг с другом.

Эта вычислительная модель в последнее время привлекает к себе всё больше и больше внимания как учёных, так

и инженеров, поскольку её реализация, что называется, в «железе» откроет широчайшие возможности для решения задач, для которых в традиционной вычислительной модели не было вполне эффективных алгоритмов решения. Например, к таким задачам относится задача факторизации заданного числа, то есть поиска списка простых делителей этого числа (на гипотезе о невозможности быстро и эффективно найти разложение числа на простые множители основаны практически все современные методы криптографии). Есть ещё ряд задач такого же плана, решение которых в модели квантовых вычислений является полиномиальным, а не экспоненциальным, как в традиционной модели.

Это делает модель квантовых вычислений крайне интересной областью исследования. Однако, как это обычно бывает, основная масса исследований проводится за рубежом в университетах и исследовательских центрах западных стран. В России если и находятся энтузиасты, то все их работы «плетутся» в хвосте передовых исследований в данном перспективнейшем направлении. Связано это отчасти с тем, что методическая база для обучения специалистов у нас настолько устарела, что говорить о подготовке высококлассных специалистов в этой области просто не приходится.

Энтузиасты пользуются англоязычной литературой, которая часто настолько непроста для понимания просто в силу того, что западная мысль уже далеко ушла вперёд, что нашему исследователю сложно найти и ухватить нить повествования. Небольшое количество переводной литературы не спасает дело. А вот наличие некоторого количества монографий и учебников по квантовым вычислениям на русском языке скорее удручает, чем восхищает. Дело всё в том, что для того, чтобы читать и понимать всю эту литературу на русском языке, необходимо полноценно владеть теорией модели квантовых вычислений.

Получается замкнутый круг — научиться негде и не у кого, а для учёбы надо уже всё знать.

В какой-то мере разорвать этот порочный круг призвана данная небольшая книга, в которой даётся самое-самое введение в модель квантовых вычислений. Для её чтения необходимо лишь владеть острым умом, желанием научиться и идти дальше в самостоятельном поиске, иметь базовые знания в математике и понимание принципов функционального программирования. Желательно наличие знания языка Haskell, его синтаксиса и операционной семантики. Это поможет понимать примеры, которыми изобилует эта книга.

Я постарался как можно больше избегать формул, строгой математики и «жёсткого матана», насколько это вообще возможно при рассмотрении подобной темы. Все понятия объясняются на пальцах, как можно более наивно. Поэтому я сразу же прошу прощения у читателя, который знаком с моделью квантовых вычислений — многие объяснения могут показаться настолько упрощёнными и наивными, что будут балансировать на грани «ликбеза». Но именно такова цель этой книги — дать объяснение новой вычислительной модели как можно более широкому кругу читателей.

Да, для чтения книги желательно, чтобы читатель был знаком с языком функционального программирования Haskell. Это связано с несколькими причинами. Во-первых, функциональное программирование в целом и язык Haskell в частности являются моей областью интереса и исследований. Пропаганда и распространение информации о языке Haskell — это то, чем я занимаюсь уже на протяжении многих лет. К тому же, это — единственный язык программирования, на котором я могу реализовывать программы, поэтому все примеры в этой книге волей-неволей пришлось писать именно на нём.

Во-вторых, функциональное программирование как парадигма разработки программного обеспечения самым гармоничным образом легла на модель квантовых вычислений, поскольку понятие «функция» включает в себя понятие «унитарное преобразование», используемое в модели квантовых вычислений. Так что в рамках функционального программирования квантовые алгоритмы выражаются самым естественным образом. Поэтому нет никакого удивления в том, что современное предложение по реализации нового языка программирования Quipper, который используется для выражения квантовых вычислительных схем, полностью основано на языке Haskell.

Ну а в-третьих, есть одна важная особенность у модели квантовых вычислений. Разработать квантовый алгоритм — это значит настолько переформатировать себе мозги и «сдвинуть парадигму», что дальше уже просто нельзя. Квантовая механика по своей сути абсолютно контр-интуитивна, и у нас нет никаких эмпирических навыков работы с бесконечномерными гильбертовыми пространствами и траекториями в них. И я считаю, что именно функциональное программирование наиболее близко из других способов программирования к парадигме квантовых вычислений. Расстояние от функционального до квантового программирования несколько короче, чем от любого иного.

Данная книга разбита на пять глав. В первой главе заинтересованному читателю предлагается описание модели квантовых вычислений, выраженное самыми простыми словами, которые только нашлись у меня. Из этой главы можно будет узнать, что такое кубит, как кубиты можно связывать друг с другом для получения многокубитовых состояний, что такое квантовая вычислительная схема и т. д. Прочитав эту главу, читатель сможет уже обратиться к более сложным источникам информации по квантовым вычислениям.

Вторая глава посвящена разработке фреймворка для выполнения квантовых вычислений на языке Haskell. Этот фреймворк позволяет решать многочисленные задачи по квантовым вычислениям и квантовой механике в целом. Реализация такого фреймворка позволяет более глубоко проникнуть в понимание модели квантовых вычислений, вынести «на пальцах» основные объекты и операции модели квантовых вычислений. После чтения этой главы, а, особенно, после самостоятельной работы с реализованным фреймворком, читатель сможет легко оперировать понятиями квантовых вычислений на практическом уровне. Но сам фреймворк будет дополняться по мере продвижения по тексту книги, особенно в четвёртой главе.

В третьей главе описывается новый язык программирования Quipper, который был разработан на базе языка Haskell для реализации квантовых алгоритмов. Приводится описание языка, его синтаксис и несколько примеров реализации квантовых схем. После ознакомления с этой главой читатель будет знать об одной из самых современных задач в рамках дальнейшей разработки модели квантовых вычислений в практическом русле. Несмотря на то, что этот новый язык ещё не реализован в виде квантового компилятора, и до реализации в «железе» ещё довольно далеко, он может быть одной из перспективнейших идей, которая будет реализована в ближайшем будущем.

Далее четвёртая и пятая главы посвящены описанию разработанных к настоящему времени квантовых алгоритмов. Не секрет, что в имеющейся литературе по квантовым вычислениям зачастую приводят описания двух или максимум трёх алгоритмов, которые разработаны к этому времени (алгоритм Дойча для распознавания сбалансированной функции, алгоритм Шора для факторизации и алгоритм Гровера для поиска — это тот самый максимум, что можно найти в современной литера-

туре). Однако на сегодняшний момент разработано уже несколько десятков квантовых алгоритмов, и число это с каждым месяцем и годом растёт. Наверняка к моменту выхода этой книги в свет число разработанных квантовых алгоритмов увеличится, и описание уже станет неполным. Однако, ознакомившись с приведёнными в книге описаниями, читателю станет намного проще ориентироваться в имеющемся разнообразии алгоритмов. Соответственно, в четвёртой главе кратко описываются алгоритмы Гровера, Дойча (и Дойча-Йожи), Саймона и два алгоритма Шора (включая алгоритм квантового преобразования Фурье), как наиболее простые алгоритмы квантовой модели вычислений. А в пятой главе приводятся краткие описания других алгоритмов.

Поскольку литература по рассматриваемому вопросу очень немногочисленна, то я считаю своим долгом произвести более или менее полноценный обзор всего, что имеется на текущий момент на русском языке. В разделе «Обзор литературы о квантовых вычислениях» приводятся ссылки на литературу и краткая аннотация к каждой книге или иному материалу. Этот раздел будет полезен тем из читателей, кто хочет углубить свои знания в этой важной теме.

Ну и немаловажным и небезынтересным будет смежный раздел «Обзор видеокурсов по квантовым вычислениям и смежным темам», поскольку сегодня со всё большим и большим внедрением в повседневную жизнь новых методов обучения (так называемые MOOC — *Massive Open Online Courses*, то есть «массовые открытые онлайн-курсы») любому человеку становятся доступны знания в виде видеолекций и интерактивных обучающих программ по практически любому направлению знаний, в том числе и по квантовым вычислениям. Для того чтобы ориентироваться и иметь представление о правильном выборе, читатель может обратиться к этому разделу.

К книге прикладываются многочисленные файлы с исходными кодами на языке Haskell, которые описываются по тексту книги. Читателю будет интересно разобраться самостоятельно с теми примерами, которые приводятся, и приложенные файлы помогут ему в этом. В случае если вы получили эту книгу без приложенных к ней файлов с примерами, вы всегда можете обратиться ко мне по электронной почте (roman.dushkin@gmail.com), чтобы получить их.

Структура и стиль этой книги устроены так, что описание многих понятий вводится постепенно, как бы исподволь. Понимание концепций модели квантовых вычислений будет накапливаться на читателя «волнами». И если в начале чтения книги может показаться, что что-то из написанного весьма непонятно, то дальше при изложении тех же понятий (возможно, иными словами и выражениями) понимание будет углубляться, расширяться и разясняться. Я заранее прошу прощения у тех из читателей, кто сложно воспринимает подобный стиль, однако по моему сугубому мнению в данном издании именно он будет наиболее применим.

Я искренне надеюсь, что этот мой скромный труд даст начало новому направлению работы в нашем научном сообществе. Я буду рад всем конструктивным отзывам, комментариям, замечаниям и, особенно, благодарностям моих читателей.

В добрый путь!

*Душкин Р. В.
Москва, 2014.*

Глава 1.

Наивное понимание модели квантовых вычислений

*Если вы думаете, что понимаете
квантовую механику, значит, вы её
не понимаете.*

Ричард Фейнман

Книгу, текст которой вы читаете или просматриваете в настоящий момент (вы же вряд ли держите в руках бумажный экземпляр, но наверняка читаете с экрана или компьютера, или электронной книгочиталки), уважаемый читатель, вряд ли можно назвать книгой по квантовой механике. Здесь нет практически ничего из того, что обычно изучают в университетах на строгих курсах по «квантам» — всех этих уравнений Шрёдингера, Гейзенберга, Паули и т. д., фотоэффектов, парадоксов типа ЭПР, элементарных частиц и так называемой Стандартной

Модели, квантования энергии и т. д., и т. п. Эта книга рассказывает о модели квантовых вычислений — новом способе осуществления вычислительных процессов таким образом, чтобы переложить на «квантовую природу» нашего мира возможность эффективной параллелизации вычислений. Квантовые вычисления — это пока лишь модель, которая ещё не нашла своего воплощения в «железе». Примерно так же воспринимали наши деды и прадеды вычислительную модель Тьюринга и фон Неймана, когда она только была разработана — до полноценной реализации в «железе» тогда было ещё очень далеко (впрочем, некоторые пытались её реализовать в «дереве»).

В рамках кибернетики давно известен принцип, гласящий, что лучшей моделью системы является сама система. Видимо, используя именно этот принцип, Ричард Фейнман как-то выразил мнение, что раз природа вокруг нас имеет квантово-механическую природу, то лучше всего её помогут смоделировать квантово-механические системы. Развивая эту мысль, можно сказать, что любое моделирование какого-либо физического процесса, выполненное при помощи современных средств вычислительной техники, будет неточным и неполным, но только лишь приближённым до какой-то степени точности, и у нас есть очень ограниченные средства повышения оной точности. В итоге любая такая модель так и останется неточной.

Однако если для моделирования использовать именно квантово-механические системы с внутренне присущей им неопределённостью и недетерминированностью, то такое моделирование позволит точно отразить те физические процессы, которые в своей основе являются абсолютно такими же — неопределёнными и недетерминированными. Это понимание и стало основой развития модели квантовых вычислений.

Другой момент, который необходимо отразить в преддверии описания модели квантовых вычислений, относится к пониманию математики, как научного языка. В самом общем понимании математика оперирует символами, при этом интерпретация символов всецело зависит от решаемой задачи, но не от самих символов. Математика просто описывает алфавит, из которого можно создавать символы, задаёт систему аксиом и правил преобразования, при помощи которых можно оперировать символами чисто синтаксическим порядком. Именно так работает классический компьютер — «он» несколько не понимает смысла производимых с битами (0 и 1) операций, «он» просто манипулирует символами. И даже наименования для битов «0» и «1» — это чистая условность, которая введена людьми для собственного удобства, поскольку компьютер манипулирует двумя различными состояниями.

Таким образом, математика — это система синтаксического манипулирования символами. Конечно, это несколько однобокое понимание, и в рамках математики есть системы, которые отрицают или расширяют такой подход. Но в целом именно этим и занимаются учёные-математики, и чем выше уровень абстракции символов, которыми они оперируют, тем меньше в них реального смысла и больше чистого синтаксиса.

Принимая во внимание оба описанных соображения, можно сказать, что квантовые вычисления — это некоторая математическая модель, формализм, помогающий в теории осуществить решение задач, которые сложно решить в традиционной модели вычислений за приемлемое время. Это достигается «автоматической» параллелизацией вычислительного процесса, что обусловлено квантово-механической природой модели.

Здесь необходимо отметить такой нюанс. Сама квантовая механика является всего лишь хорошо работающей теорией, которая позволяет прогнозировать (очень точно) результаты

экспериментов и решать прикладные задачи. Узнать, действительно ли эта теория описывает наблюдаемую нами объективную реальность, невозможно. Но теория очень хорошо согласуется с практикой, и сегодня нет никаких причин отказываться от этой прекрасной теории, поскольку ничего более точного не разработано.

Таким образом, модель квантовых вычислений является неким подмножеством квантовой механики, а сама квантовая механика является неплохо подтверждаемой практикой теорией, описывающей объективную реальность. Это наводит на мысль о том, что несмотря ни на что, в конце концов, модель квантовых вычислений можно будет реализовать в «железе», и все разработанные алгоритмы можно будет реализовать на квантово-механическом компьютере.

На сегодняшний день, к сожалению, реализации в «железе» ещё не существует. Есть несколько перспективнейших направлений исследований, однако большинство из них упирается пока ещё в непреодолимые технологические ограничения. Но, в конце концов, все ограничения будут сняты, ибо нет фундаментальных ограничений, и тогда модель квантовых вычислений будет реализована в виде работающего квантового компьютера. В этой книге мы не будем рассматривать исследования по реализации модели в «железе», но читатель должен помнить, что рано или поздно квантовый компьютер будет создан, и тогда все знания, полученные при помощи чтения этой и многих подобных книг, будут востребованы на вес золота.

Кроме того, большинство имеющихся на сегодняшний момент книг по квантовым вычислениям содержат описания физических экспериментов, всевозможные наукоёмкие перспективные технологии (ядерно-магнитные резонансные компьютеры, компьютеры на ионных ловушках, компьютеры на одиночных молекулах и др.), а вот простому описанию мо-

дели внимания уделяется мало, незаслуженно мало. И получается некоторый перекося — для физиков-теоретиков и физиков-экспериментаторов научной литературы по квантовым вычислениям довольно много, а для программистов нет вообще. А среднестатистический разработчик программного обеспечения даже в рамках классической вычислительной модели вряд ли знаком с физикой процессов, происходящих внутри процессора. Так и здесь, в этой новой области исследований требуется литература, которая будет готовить специалистов, которые смогут не физически реализовывать какие-либо устройства, а программировать их на логическом уровне.

Ну и теперь получается, что ознакомившись с информацией в этой книге вдумчивый читатель будет готов к переходу к более серьёзной литературе по теме. В соответствующем справочном разделе в конце книги всякий читатель найдёт для себя продолжение, если после прочтения возникнет непреодолимое желание погрузиться в тему дальше. Поэтому я, как автор, надеюсь, что мой скромный труд станет тем трамплином, который поможет вырастить мириады новых программистов.

Квантовые состояния и кубиты

Прежде всего, для понимания модели квантовых вычислений необходимо овладеть понятийным аппаратом, который используется для описания и работы. Вся терминология пришла напрямую из квантовой механики, поэтому для тех, кто вполне владеет квантово-механическим аппаратом, понимание модели квантовых вычислений будет простым (хотя и здесь придётся приложить усилия к изучению нескольких терминов, пришедших из теории информации и теории вычислений). Здесь будут даны самые простейшие определения терминов. Тем же чита-

телям, кто захочет полностью и глубоко понять математический аппарат, лежащий за описываемой моделью, имеет смысл обратиться к серьёзной литературе по квантовой механике.

Перед рассмотрением основного понятия в модели квантовых вычислений — кубита, необходимо изучить понятие квантового состояния. Квантовым состоянием будем называть совокупность из некоторого символа (наименования квантового состояния) и приписанного к нему коэффициента, причём этот коэффициент является комплексным числом. Квантовое состояние будет записываться как:

$$\alpha|s\rangle$$

где α — комплексночисленный коэффициент, а s — наименование квантового состояния. Последнее обычно состоит из одного символа, например: $\langle 0 \rangle$, $\langle 1 \rangle$, $\langle + \rangle$, $\langle - \rangle$. Так что, квантовыми состояниями, например, являются такие объекты, как $|0\rangle$ и $|1\rangle$. А можно придумать и более сложные квантовые состояния, например $\frac{1-i}{2}|\uparrow\rangle$.

Кубитом, в этом случае, называется просто список квантовых состояний. Это слишком общее определение, и в других книгах по квантовым вычислениям обычно даётся иное определение. Однако здесь мы заострим внимание на этом аспекте — кубит состоит из списка квантовых состояний, при этом есть одно ограничение — сумма квадратов модулей всех комплексночисленных коэффициентов обязательно должна равняться 1.

Обычно и всегда это введённое таким образом понятие ограничивают. Поскольку традиционный бит представляет собой возможность выбора из двух альтернатив (0 или 1), то и кубит ограничивают двумя квантовыми состояниями в списке. Далее станет понятно, почему именно так, а теперь имеет смысл перейти к рассмотрению понятия «базис».

Базисом называется набор кубитов, которые взаимно ортогональны друг другу. Если ограничивать рассмотрение кубитов двумя квантовыми состояниями, то, само собой разумеется, базис состоит из двух кубитов. Однако же что такое «ортогональность» в применении к кубитам? Если кубит — это всего лишь список комплексных чисел, к которым приписаны некоторые наименования квантовых состояний, что как могут быть ортогональны такие «именованные комплексные числа»?

Всё просто. Дело в том, что базис выбирается тоже произвольным образом. Для простоты понимания и соответствия традиционному пониманию бита базисом назван набор кубитов $|0\rangle$ и $|1\rangle$, после чего было введено векторное представление кубита. Для этих двух базисных кубитов векторное представление следующее:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

И, собственно, ортогональность в таком случае определяется как равенство нулю скалярного произведения двух векторных представлений кубитов. Но что из себя представляют эти числа 0 и 1 в векторах? Это ничто иное, как комплексно-численные коэффициенты α при базисных кубитах, то есть так и получается, что $|0\rangle = 1|0\rangle + 0|1\rangle$, а $|1\rangle = 0|0\rangle + 1|1\rangle$. Таким образом, произвольный кубит можно разложить в базисе, и такое разложение записывается как $|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle$, и оно называется *линейной суперпозицией базисных состояний*, и, соответственно, в виде вектора такой кубит представляется как:

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

где α и β — некоторые комплексные числа такие, что сумма квадратов их модулей равна строго 1.

Всё дело в том, что комплексный коэффициент при базисном кубите, или, что то же, при базисном квантовом состоянии, — это так называемая *амплитуда вероятности*. Это понятие из квантовой механики, и оно обозначает тот простой факт, что при измерении вероятность обнаружения кубита в этом квантовом состоянии равна квадрату модуля его амплитуды. Именно поэтому сумма квадратов модулей должна равняться строго единице, поскольку при измерении кубит будет обнаружен либо в том, либо в другом базисном квантовом состоянии.

Но что такое измерение? Это понятие пришло непосредственно из квантовой механики, где оно определяется достаточно сложно. Здесь же для упрощения рассмотрим такое определение. Измерение кубита — это попытка ответить на вопрос типа «Находится ли данный кубит в квантовом состоянии $|0\rangle$?» или «Каковы амплитуды вероятности нахождения данного кубита в квантовых состояниях, определяемых некоторым базисом?». Ответом на первый вопрос становится амплитуда вероятности нахождения кубита в запрашиваемом квантовом состоянии, а ответом на второй вопрос — набор амплитуд вероятностей, соответствующих базисным квантовым состояниям, сумма которых, конечно же, равна единице.

Необходимо отметить, что, как следует из положений квантовой механики, после измерения кубит переходит в какое-либо состояние из числа входящих в базис, в рамках которого производится измерение, при этом вероятность перехода кубита в это состояние равна как раз квадрату модуля амплитуды при этом состоянии в базисе. Понять это проще всего на нескольких незамысловатых примерах, однако перед их рас-

смотрением стоит более подробно изучить используемую нотацию.

Читатель уже заметил эти странные скобки — $|s\rangle$. Это так называемая «нотация Дирака». Придумана она давным-давно, однако именно такой её внешний вид позволяет с лёгкостью понимать смысл вычислений и применять модель квантовых вычислений. Прежде всего, надо отметить, что $|s\rangle$ называется «кет-вектором», а $\langle s|$, соответственно, «бра-вектором». Словечки «бра» и «кет» — это две части английского слова «bracket», которое переводится как «скобка». Соответственно, кет-вектор — это вектор столбец, которые мы уже видели при обсуждении векторного представления кубитов. А бра-вектор — это комплексно-сопряжённый с соответствующим кет-вектором вектор-строка. То есть, например, если у нас есть некоторый кубит

$$|\varphi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

то соответствующим бра-вектором будет

$$\langle\varphi| = (\alpha^* \ \beta^*)$$

где α^* и β^* — комплексные сопряжённые чисел α и β соответственно (для комплексного числа $a + bi$ комплексным сопряжённым будет комплексное число $a - bi$).

Из этого следует простое мнемоническое правило — если соединить бра- и кет-вектор, чтобы получился «бракет», то это будет *скалярным произведением* двух векторов:

$$\langle\varphi|\varphi\rangle = (\alpha^* \ \beta^*) \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \alpha^* \alpha + \beta^* \beta = |\alpha|^2 + |\beta|^2$$

(если комплексное число умножить на его комплексное сопряжённое, то, без всяких сомнений, получится квадрат модуля

этого числа, что каждый читатель может легко проверить при помощи формулы перемножения многочленов).

А что будет, если соединить бра- и кет-векторы в обратном порядке, то есть в виде «кетбра»? Это, само собой разумеется, записывается как $|\varphi\rangle\langle\varphi|$, а поскольку это векторы, то результатом такого их перемножения явится матрица. Если векторы имеют размерность 2, то такая матрица будет иметь размерность 2×2 :

$$|\varphi\rangle\langle\varphi| = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} (\alpha^* \ \beta^*) = \begin{pmatrix} \alpha\alpha^* & \alpha\beta^* \\ \beta\alpha^* & \beta\beta^* \end{pmatrix}$$

Это так называемая *матрица плотности* $\rho = |\varphi\rangle\langle\varphi|$. Этот термин ещё пригодится при детальном изучении модели квантовых вычислений.

Таким образом, есть два простых мнемонических правила:

1. $\langle\varphi|\varphi\rangle$ — скалярное произведение, которое называется «бракет» (от англ. *bracket*).
2. $|\varphi\rangle\langle\varphi|$ — матрица плотности, что проще всего запомнить в виде эдакого неформального преобразования $|\varphi\rangle\langle\varphi| \rightarrow |\varphi \times \varphi|$, то есть обращённые одна к другой угловые скобки как бы превратились в знак умножения.

Теперь можно более чётко понять, что такое измерение. Например, чтобы ответить на вопрос «Находится ли данный кубит $|\varphi\rangle$ в квантовом состоянии $|0\rangle$?», необходимо просто выполнить операцию $\langle 0|\varphi\rangle$, и ответом на вопрос станет квадрат модуля полученного в результате вычисления числа. Само собой разумеется, что для квантового состояния $|0\rangle$ это тривиально, но то же самое правило действует и для других квантовых состояний — необходимо просто посчитать скалярное произведение, взять модуль результата и возвести его в квадрат.

Всё это понять ещё легче, если обратиться к графическому представлению. Несмотря на то, что до этого момента для иллюстрации понятий квантовое состояние и кубит использовались некоторые символы и манипуляции ими, этим символам можно дать определённые интерпретации, в частности — графическую. Но для этого для начала необходимо несколько упростить задачу — пусть коэффициенты перед квантовыми состояниями будут не комплексными, а действительными. В этом случае всё очень просто: каждый кубит представляет собой единичный вектор, а его разложение в базисе — это всего лишь проекции данного вектора на произвольно выбранные ортогональные «оси»:

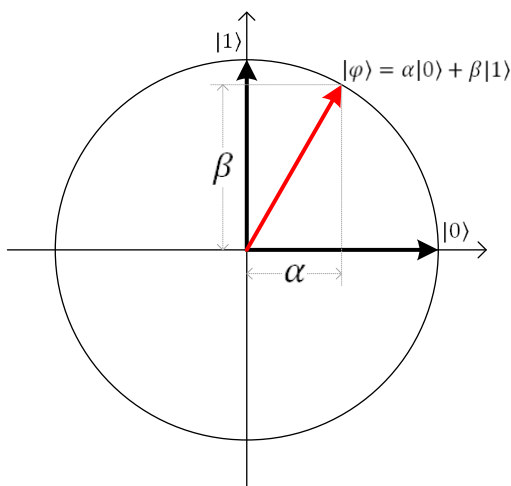


Рис. 1. Графическое представление разложения кубита в базисе в случае действительных коэффициентов

В этом случае становится понятным, почему $\alpha^2 + \beta^2 = 1$ — это всего лишь теорема Пифагора (конечно же, окружность в данном случае всегда имеет радиус 1).

Тут так же необходимо отметить, что «оси», то есть базисные квантовые состояния выбраны абсолютно произвольно — единственное условие, которому они должны удовлетворять, — это ортогональность. Просто так уж сложилось по традиции, что выбирают горизонтальный вектор $|0\rangle$ и вертикальный вектор $|1\rangle$. Но, само собой разумеется, это не единственный базис. Базисов может существовать бесконечное количество — любая пара ортогональных единичных векторов может служить базисом.

Например, другим часто используемым базисом является базис $\{|+\rangle, |-\rangle\}$:

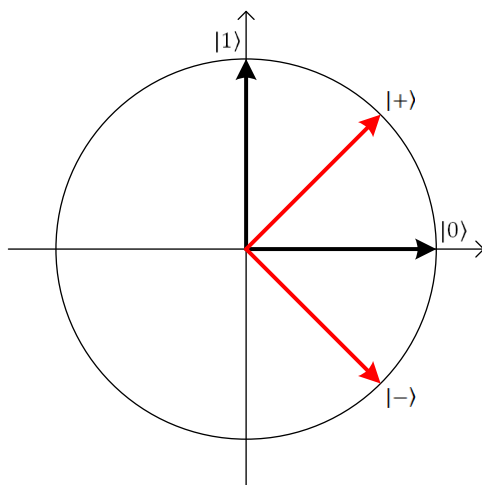


Рис. 2. Базис $\{|+\rangle, |-\rangle\}$ и его расположение относительно стандартного базиса

Естественно, что оба квантовых состояния этого базиса можно выразить через основной базис $\{|0\rangle, |1\rangle\}$, и сделать это довольно просто. Даже не вдаваясь в серьёзные расчёты (хотя и это можно было бы сделать), но используя только упомяну-

тую ранее теорему Пифагора, можно вычислить значения коэффициентов α и β , которые в данном случае равны. Это делается так (помним, что α в данном случае — действительное число, поэтому квадрат его модуля равен просто квадрату этого числа):

$$2\alpha^2 = 1 \Rightarrow \alpha^2 = \frac{1}{2} \Rightarrow \alpha = \frac{1}{\sqrt{2}}$$

то есть:

$$|+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$

$$|-\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$$

Этот базис также часто используется в модели квантовых вычислений, поэтому его хорошо бы постоянно держать в уме. По крайней мере, формулы пересчёта из одного базиса в другой полезно помнить наизусть.

Эти формулы помогут, к примеру, отвечать на такие вопросы, как «Если выполнить измерение заданного кубита в базисе $\{|+\rangle, |-\rangle\}$, то с какой вероятностью кубит примет значение $|+\rangle$?» и подобные. Ведь если есть кубит, для которого имеется выражение в стандартном базисе $|\varphi\rangle$, то для ответа на этот вопрос достаточно произвести операцию $\langle +|\varphi\rangle$, и в результате будет получена амплитуда искомой вероятности.

Например, если кубит $|\varphi\rangle$ разлагается в стандартном базисе как $|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle$, то для перевода его в базис $\{|+\rangle, |-\rangle\}$ можно воспользоваться следующей формулой:

$$|\varphi\rangle_{+|-} = \langle +|\varphi\rangle|+\rangle + \langle -|\varphi\rangle|-\rangle = \frac{\alpha + \beta}{\sqrt{2}}|+\rangle + \frac{\alpha - \beta}{\sqrt{2}}|-\rangle$$

Эта формула поясняется при помощи диаграммы на следующем рисунке:

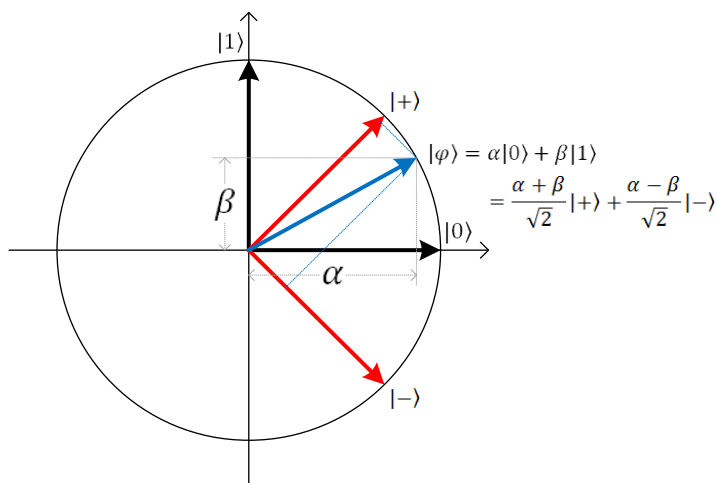


Рис. 3. Перевод кубита в базис $\{|+\rangle, |-\rangle\}$

Собственно, матричные операции дают возможность не только ответить на первый вопрос об измерении, но и на второй, то есть сразу узнать каковы амплитуды вероятности нахождения данного кубита в квантовых состояниях, определяемых некоторым базисом? Для этого надо перемножать не векторы, а сразу умножить сопряжённую матрицу, представляющую собой базис, на векторное представление кубита. В результате получится вектор, представляющий собой именно амплитуды вероятностей.

Например, матрица стандартного базиса выглядит как

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

и её комплексно-сопряжённая матрица выглядит точно так же. Если эту матрицу умножить на представление кубита в стандартном базисе, то в результате, как ни странно, получатся коэффициенты α и β , то есть амплитуды вероятности нахож-

дения кубита в базисных состояниях $|0\rangle$ и $|1\rangle$. А вот матрица базиса $\{|+\rangle, |-\rangle\}$ выглядит так:

$$\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$$

и её комплексно-сопряжённая матрица абсолютно такая же. Соответственно, можно произвести умножение комплексно-сопряжённой матрицы на кубит $|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle$:

$$\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \frac{\alpha + \beta}{\sqrt{2}} \\ \frac{\alpha - \beta}{\sqrt{2}} \end{pmatrix}$$

что полностью соответствует полученному ранее результату.

Все эти примеры, которые рассматривались выше, имеют один нюанс — коэффициенты в квантовых состояниях кубитов в них действительные. Честно говоря, это нисколько не уменьшает силу квантовой вычислительной модели, однако в объективном мире уж так устроено, что квантовая механика оперирует с комплексными коэффициентами. Такова модель описания реальности, принятая в квантовой механике, а потому она перенесена и в вычислительную модель, поскольку в конце концов кубиты будут реализованы в «железе», и такая реализация кубитов будет иметь дело именно с комплексными коэффициентами. Поэтому здесь надо рассмотреть дополнительную визуализацию модели при помощи диаграммы, но уже с квантовыми коэффициентами.

Если действительные коэффициенты при двух базисных квантовых состояниях приводили к появлению одномерного пространства состояний (окружность), то резонно предпопо-

жить, что использование комплексных коэффициентов приведёт к появлению двумерного пространства — сферы. Так оно и есть, и такая визуализация называется *сферой Блоха*.

Тут есть один тонкий момент (те, кто не любит формул, могут сразу же перейти к следующей странице и посмотреть на приятную взору диаграмму). Поскольку каждый кубит нормирован, его длина всегда равна единице (то есть, напомним, $|\alpha|^2 + |\beta|^2 = 1$), то разложение в стандартном базисе можно записать как

$$|\varphi\rangle = e^{i\gamma} \left(\cos \frac{\theta}{2} |0\rangle + e^{i\omega} \sin \frac{\theta}{2} |1\rangle \right)$$

Это выражение для кубита $|\varphi\rangle$ осуществляется на основании широко известной формулы Эйлера ($e^{ix} = \cos x + i \sin x$). Поскольку коэффициенты α и β являются комплексными числами, то ничто не мешает записать их как систему уравнений:

$$\begin{aligned} \alpha &= e^{i\gamma} \cos \frac{\theta}{2} = \cos \gamma \cos \frac{\theta}{2} + i \sin \gamma \cos \frac{\theta}{2} \\ \beta &= e^{i\gamma} e^{i\omega} \sin \frac{\theta}{2} = e^{i(\gamma+\omega)} \sin \frac{\theta}{2} \\ &= \cos(\gamma + \omega) \sin \frac{\theta}{2} + i \sin(\gamma + \omega) \sin \frac{\theta}{2} \end{aligned}$$

Так уж получилось, что в квантовом мире множитель $e^{i\gamma}$ можно опустить, поскольку он не приводит к каким-либо наблюдаемым эффектам — это просто некоторый фазовый коэффициент, который при проведении измерения никак не влияет на результаты. Это значит, что в данном случае приведённые выше уравнения можно переписать в виде:

$$\begin{aligned} \alpha &= \cos \frac{\theta}{2} + i \sin \frac{\theta}{2} \\ \beta &= \cos \omega \sin \frac{\theta}{2} + i \sin \omega \sin \frac{\theta}{2} \end{aligned}$$

Так что кубит описывается двумя угловыми параметрами — θ и ω , а они приводят к чёткой геометрической интерпретации кубита с комплексными коэффициентами в виде точки на сфере:

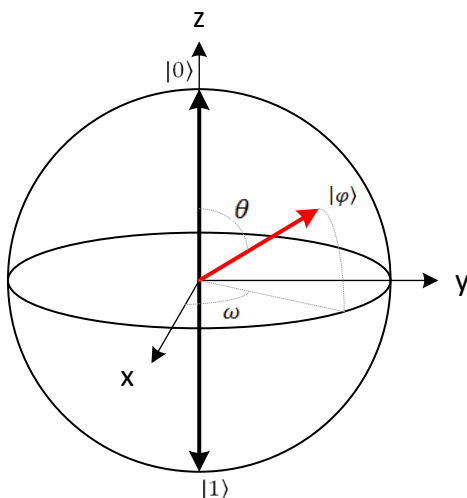


Рис. 4. Кубит как точка на сфере Блоха

Как видно, в данном представлении базис представляется парой квантовых состояний, не ортогональных друг другу, но разнонаправленных и лежащих на одной прямой. На диаграмме показан стандартный базис, но вдумчивый читатель может решить вышеприведённые уравнения для базиса $\{|+\rangle, |-\rangle\}$ и отобразить его на сфере Блоха.

И теперь осталось упомянуть об одной важной особенности операции измерения в модели квантовых вычислений, которая, если можно так выразиться, контр-интуитивна. Измерение — это дорога в один конец. Другими словами, если мы произвели измерение кубита и получили какой-то конкретный результат (с определённой вероятностью), то сам кубит принял значение,

полученное в результате измерения, и вся суперпозиция базисных состояний была потеряна. Восстановить её невозможно.

Таким образом, после измерения вся квантовая информация, которая хранилась посредством суперпозиции базисных состояний с комплексными коэффициентами, теряется, а остаётся только одна из двух альтернатив. Именно поэтому кубит и называется «битом» — несмотря на то, что потенциально в нём в «скрытом виде» хранится бесконечное количество информации (произвольная суперпозиция двух ортонормированных базисных состояний), при измерении из него можно выудить только один бит информации — либо одно, либо другое состояние в базисе измерения.

Несколько кубитов

Итак, из рассмотренного до сего времени сложно сделать выводы о том, что модель квантовых вычислений даёт какое-то преимущество по сравнению с традиционной вычислительной моделью. Ну да, один кубит содержит в скрытом виде бесконечное количество информации, но мы никак не можем этим воспользоваться. Это именно скрытая информация, которая используется внутри кубита неизвестным для нас способом. В чём же дело?

Интересная особенность квантовых вычислений проявляется тогда, когда на сцену выходят многокубитовые системы. Дело в том, что правила комбинации кубитов разительно отличаются от правил комбинации битов. Проще всего рассмотреть вариант комбинации двух кубитов.

В квантовой механике и при обычном рассмотрении модели квантовых вычислений считается, что любые два кубита всегда находятся в разных пространствах состояний, поэтому даже если обозначения их базисов одинаковые, то всё равно считается, что базисы разные (и для этого иногда используют индексы,

которые тут же начинают опускать). Однако это — усложнение рассмотрения, поэтому далее будет считаться, что все кубиты находятся в одном и том же пространстве состояний и раскладываются на суперпозицию в одинаковых базисах.

Так что пусть есть два кубита $|\varphi\rangle$ и $|\psi\rangle$. Они представляют собой две суперпозиции базисных состояний, скажем, в стандартном базисе: $|\varphi\rangle = \alpha_1|0\rangle + \beta_1|1\rangle$ и $|\psi\rangle = \alpha_2|0\rangle + \beta_2|1\rangle$ — это просто два обычных кубита. Но что будет, если их соединить в двухкубитовую систему? Всё просто:

$$|\varphi\rangle|\psi\rangle = \alpha_1\alpha_2|0\rangle|0\rangle + \alpha_1\beta_2|0\rangle|1\rangle + \beta_1\alpha_2|1\rangle|0\rangle + \beta_1\beta_2|1\rangle|1\rangle$$

Для упрощения обозначений запись $|\varphi\rangle|\psi\rangle$ сокращают до $|\varphi\psi\rangle$, и это так называемое *тензорное произведение* кубитов или, ещё шире, тензорное произведение квантовых систем. Внезапно система из двух кубитов представляет собой суперпозицию четырёх базисных состояний (а то, что эти состояния — $|00\rangle$, $|01\rangle$, $|10\rangle$ и $|11\rangle$ являются базисными, мы скоро убедимся). Вдумчивый читатель уже понял, что добавление третьего кубита в квантовую систему приводит к появлению суперпозиции восьми состояний. Так что система из n кубитов представляет собой суперпозицию из 2^n квантовых состояний. Не это ли экспоненциальное увеличение мощности вычислительной модели?

Квантовая модель вычислений говорит, что вычисления с кубитом производятся одновременно со всеми базисными состояниями, на которые кубит разложен. Поскольку для одного кубита имеет место два базисных состояния, при работе с одним кубитом вычисления одновременно производятся с двумя квантовыми состояниями. Таким образом, при работе с многокубитовой системой вычисления одновременно производятся с экспоненциальным числом квантовых состояний, как показано в предыдущем абзаце.

Учёные говорят, что во всей Вселенной всего имеется около 10^{80} всевозможных атомов, а это число примерно равно 2^{266} , то есть если сделать нелепый традиционный компьютер из всей Вселенной, в котором каждый атом будет представлять один бит информации, то этот компьютер сможет успешно смоделировать только 266 кубита. Ну вот как-то так можно представить себе вычислительную мощность модели квантовых вычислений.

Таким образом, многокубитовая система представляет собой не просто набор кубитов, а нечто большее, поскольку кубиты взаимосвязаны друг с другом, и суперпозиция образуется на всю систему, а не на каждый кубит в отдельности. Это значит, что для многокубитовой системы при разложении на суперпозицию базисных состояний должен определяться новый базис. Предыдущее рассмотрение уже показало, что в таком базисе должно быть 2^n базисных состояний для n -кубитовой системы.

Что такое тензорное произведение векторов? Проще всего понять суть выражения $|00\rangle$ следующим образом. Необходимо вспомнить векторное представление квантового состояния $|0\rangle$, после чего произвести операцию тензорного произведения:

$$|00\rangle = |0\rangle|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ 0 \begin{pmatrix} 1 \\ 0 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Точно таким же образом выходит следующее представление других квантовых состояний двухкубитовой системы:

$$|01\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

$$|10\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

$$|11\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Можно простейшими вычислениями убедиться, что эти четыре вектора образуют ортонормированный базис в четырёхмерном комплекснозначном пространстве. И, соответственно, матрица этого базиса выглядит следующим образом:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Несложные умозаключения позволяют понять, что для системы из n -кубитов стандартным базисом является единичная матрица размера $2^n \times 2^n$. Но, само собой разумеется, что базисов может быть бесконечное число: единственное ограничение — это ортонормированность всех векторов.

А что, если в формуле разложения в стандартном базисе для двух кубитов, выполнить некий финт ушами? Ведь ничто не запрещает нам сделать вот какую-то такую суперпозицию квантовых состояний:

$$|\varphi\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$$

Эта суперпозиция отвечает главному свойству — её коэффициенты, то есть амплитуды вероятности, соответствуют условию равенства единице суммы квадратов модулей. Так что такое разложение системы из двух кубитов вполне может существовать. Однако как оно получено? Ведь действительно,

можно легко показать, что не существует таких двух кубитов, чьё тензорное произведение давало бы такой результат. Достаточно рассмотреть систему уравнений:

$$\begin{cases} \alpha_1 \alpha_2 = \frac{1}{\sqrt{2}} \\ \alpha_1 \beta_2 = 0 \\ \beta_1 \alpha_2 = 0 \\ \beta_1 \beta_2 = \frac{1}{\sqrt{2}} \end{cases}$$

чтобы понять, что у неё нет решения. Так что такая квантовая суперпозиция была получена как-то иначе (её реально можно получить на физических объектах — элементарных частицах, так что это не игра разума теоретиков). Действительно, это так называемое *запутанное состояние*, и у таких состояний есть множество применений в рамках модели квантовых вычислений. В частности, квантовая телепортация, о которой можно прочесть в большинстве источников по квантовым вычислениям (поэтому мы не будем её здесь описывать), работает именно с запутанными состояниями.

В практических целях выделяют четыре подобных состояния, которые называются состояниями Белла по имени одного из исследователей парадокса Эйнштейна — Подольского — Розена, в котором формулируются вопросы, на которые отсутствуют ответы в рамках классической физики. Джон Стюарт Белл придумал чрезвычайно хитроумный способ доказать при помощи эксперимента, что в квантовой механике нет так называемых «скрытых параметров», и что сама природа мира оперирует неопределённостью, которая «схлопывается» в момент измерения. Но эта занимательная история выходит за рамки данной книги, поэтому заинтересованного читателя можно отослать к более детальной литературе по квантовой механике.

Итак, есть четыре состояния Белла, которые, кстати, образуют ортонормированный базис (что легко показывается — скалярное произведение любой пары этих состояний равно 0). Вот они:

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$$

$$|\Phi^-\rangle = \frac{1}{\sqrt{2}}|00\rangle - \frac{1}{\sqrt{2}}|11\rangle$$

$$|\Psi^+\rangle = \frac{1}{\sqrt{2}}|01\rangle + \frac{1}{\sqrt{2}}|10\rangle$$

$$|\Psi^-\rangle = \frac{1}{\sqrt{2}}|01\rangle - \frac{1}{\sqrt{2}}|10\rangle$$

В общем, эти состояния рекомендуется выучить и запомнить — они понадобятся во многих приложениях модели квантовых вычислений. Очень многие квантовые алгоритмы используют их. Более того, именно наличие таких запутанных состояний позволяет осуществлять некоторые «фишки» квантовых вычислений, недоступные в классической вычислительной модели, поскольку в ней попросту нет аналога этому явлению.

Вот появились некие свойства модели квантовых вычислений, которые, вроде бы, повышают эффективность этой модели по сравнению с классическими вычислениями. Однако здесь не всё так просто. Дело в том, что сама теория сложности вычислений является областью настолько неразведанной, что сказать абсолютно точно об эффективности квантовых вычислений по сравнению с вычислениями обычными ничего нельзя. Да, разработаны квантовые алгоритмы, которые для некоторых задач показывают кардинальное увеличение эффективности. Например, для задачи разложения числа на простые множители пока не существует эффективного алгоритма в классической

модели вычислений, а в модели квантовых вычислений такой алгоритм есть. Но никто пока не смог доказать, что такой же эффективный алгоритм в конце концов не будет найден и для классических вычислений. Так что вопрос о том, является ли модель квантовых вычислений эффективнее традиционной модели, остаётся открытым.

Область эта — всё ещё *terra incognita*, и нас ждёт множество открытий в теории сложности вычислений. Но пока мы даже не знаем, равны ли классы сложности **P** (алгоритмы, работающие за полиномиальное время) и **NP** (алгоритмы, результат работы которых можно проверить за полиномиальное время). Если окажется так, что равны, то вряд ли можно сказать, что модель квантовых вычислений эффективнее классической вычислительной модели.

Гейты и квантовые схемы

Теперь мы готовы перейти собственно к вычислениям. Всё рассмотренное перед этим разделом касалось базиса и кубитов (ну и многокубитовых состояний), однако о вычислениях пока речи не шло. Сейчас же переходим к вычислениям и определяем, как и какие операции можно производить над кубитами.

Читатель, знакомый со схемной нотацией вычислительных процессов над классическими битами, должен вспомнить, что вычисление в традиционной вычислительной модели может быть определено при помощи нескольких равнозначных формализмов — машины Тьюринга (в самых разных её вариантах), лямбда-исчисления и, наконец, при помощи булевых схем. Этот способ мы сейчас и вспомним.

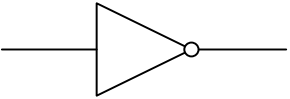
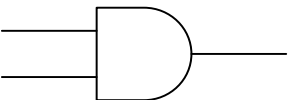
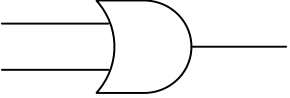
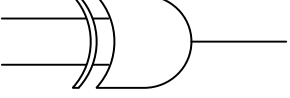
Вычислительный процесс представляет собой функцию, которая в общем случае определяется следующим образом:

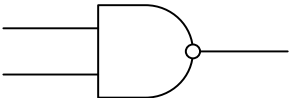
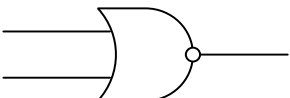
$$f: \{0,1\}^n \rightarrow \{0,1\}^m$$

то есть функция является отображением упорядоченного набора из n битов в упорядоченный набор из m битов.

Собственное, такие функции можно «конструировать» из мелких блоков, которые традиционны для схемотехники. Вот, к примеру, некоторые из традиционно использующихся блоков:

Таблица 1. Основные схемотехнические элементы

Функция	Схемотехническое обозначение	Таблица истинности
НЕ		$f: \{0,1\} \rightarrow \{0,1\}$ $f\ 0 = 1$ $f\ 1 = 0$
И		$f: \{0,1\}^2 \rightarrow \{0,1\}$ $f\ 0\ 0 = 0$ $f\ 0\ 1 = 0$ $f\ 1\ 0 = 0$ $f\ 1\ 1 = 1$
ИЛИ		$f: \{0,1\}^2 \rightarrow \{0,1\}$ $f\ 0\ 0 = 0$ $f\ 0\ 1 = 1$ $f\ 1\ 0 = 1$ $f\ 1\ 1 = 1$
Исключающее ИЛИ		$f: \{0,1\}^2 \rightarrow \{0,1\}$ $f\ 0\ 0 = 0$ $f\ 0\ 1 = 1$ $f\ 1\ 0 = 1$ $f\ 1\ 1 = 0$

Функция	Схемотехническое обозначение	Таблица истинности
НЕ-И		$f: \{0,1\}^2 \rightarrow \{0,1\}$ $f \ 0 \ 0 = 1$ $f \ 0 \ 1 = 1$ $f \ 1 \ 0 = 1$ $f \ 1 \ 1 = 0$
НЕ-ИЛИ		$f: \{0,1\}^2 \rightarrow \{0,1\}$ $f \ 0 \ 0 = 1$ $f \ 0 \ 1 = 0$ $f \ 1 \ 0 = 0$ $f \ 1 \ 1 = 0$

Как видно, это обычные логические операции — отрицание, конъюнкция, дизъюнкция, исключающее ИЛИ и т. д. Они используются в совершенно разных областях деятельности, но были введены именно в рамках традиционной вычислительной модели как полноценная замена формализмам машины Тьюринга и лямбда-исчисления.

Эти функции или схемотехнические элементы также могут составлять базис в том смысле, что имеются некоторые наборы (в том числе, минимальные по количеству элементов), через элементы которых можно выразить произвольную функцию любой размерности. Приведённый в таблице 1 набор элементов является базисным (но не минимальным — в нём есть избыточные элементы, которые могут быть выражены через другие элементы; например, элемент ИЛИ может быть выражен при помощи комбинации двух элементов НЕ и одного элемента НЕ-И посредством известного правила де Моргана). При помощи него, например, можно составить следующую безынтересную функцию:

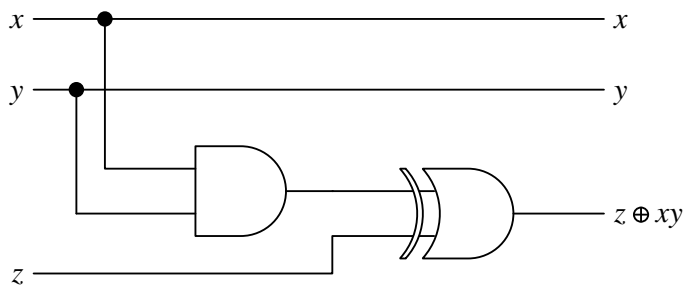


Рис. 5. Схемное представление элемента Тоффоли

Это так называемый *элемент Тоффоли* (или, как ещё называют, «вентиль Тоффоли»), замечательная функция, которая одна сама по себе образует минимальный базис — через неё одну можно выразить любую другую функцию общего вида. Её таблица истинности выглядит следующим образом:

Таблица 2. Результат работы элемента Тоффоли

Вход			Выход		
x	y	z	x	y	$z \oplus xy$
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

Действительно, если мы определим эту функцию следующим образом (и здесь мы начинаем использовать для пояснения язык программирования Haskell):

```
toffoli :: Int -> Int -> Int -> (Int, Int, Int)
toffoli 1 1 0 = (1, 1, 1)
toffoli 1 1 1 = (1, 1, 0)
toffoli x y z = (x, y, z)
```

```
getX, getY, getZ :: (Int, Int, Int) -> Int
getX (x, _, _) = x
getY (_, y, _) = y
getZ (_, _, z) = z
```

тогда все другие базовые функции (из таблицы 1) можно определить в точности через неё и только через неё одну (с учётом вспомогательных функций «добывания» компонентов тройки):

```
not :: Int -> Int
not = getZ . toffoli 1 1

and :: Int -> Int -> Int
and a b = getZ $ toffoli a b 0

nand :: Int -> Int -> Int
nand a b = getZ $ toffoli a b 1

or :: Int -> Int -> Int
or a b = not $ and (not a) (not b)

nor :: Int -> Int -> Int
nor a b = not $ or a b

xor :: Int -> Int -> Int
xor a b = getZ $ toffoli 1 a b
```

Несколько некорректными выглядят определения функций `or` и `nor`, поскольку они выражены не напрямую через функцию `toffoli`, а при помощи использования правил булевой логики. Тем не менее, такой подход тоже возможен. Ну а вдумчивому читателю предлагается поразмыслить и попробовать выразить

эти функции через функцию `toffoli` при помощи фиксации тех или иных аргументов, либо при помощи каких-либо иных способов.

При этом надо отметить, что сама по себе функция `not` является обратимой (то есть взаимно однозначной, по значению её выхода можно доподлинно восстановить значение входа), поэтому её выражение через элемент Тоффоли вообще является очень неэффективным и неэкономным. Это значит, что при разработке алгоритмов для реализации функций в квантовой вычислительной модели разработчик всегда должен обращать внимание на свойства функции. Есть такие функции, которые обратимы сами по себе (как функция `not` или дискретное преобразование Фурье, о котором много будет рассказано далее), и их можно реализовать вообще без потерь энергии.

Кроме этого интересного элемента есть ещё один широко описываемый в литературе — это так называемый *элемент Фредкина*. Его отличие от только что рассмотренного элемента заключается в том, что из трёх его входных битов на выходе меняются два, а не один, как у элемента Тоффоли. Вот соответствующая таблица истинности:

Таблица 3. Результат работы элемента Фредкина

Вход			Выход		
C	I ₁	I ₂	C	O ₁	O ₂
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0

1	0	1	1	1	0
1	1	0	1	0	1
1	1	1	1	1	1

То есть видно, что изменения происходят на предпоследней и третьей с конца строках. На самом деле, изменения производятся тогда, когда бит C равен 1, и изменения эти заключаются в том, что значения двух других битов меняются местами. То есть, если $C = 1$, то $O_1 = I_2$, а $O_2 = I_1$. Когда $I_1 = I_2$ изменений, конечно же, не видно.

Этот элемент обладает теми же свойствами, что и предыдущий рассмотренный — он один образует базис функций, так как при помощи комбинаций из одного лишь этого элемента можно построить любую другую функцию. Для читателя будет интересным попробовать сделать это, а также выразить элемент Тоффולי через элемент Фредкина и наоборот.

Если внимательно посмотреть на элементы Тоффולי и Фредкина, то будет видна одна особенность. На поверхности видно то, что оба этих элемента имеют одинаковое количество входных и выходных битов. Но если взглянуть поглубже, то окажется, что это неспроста. По результату, то есть по выходным битам элемента можно однозначно восстановить все входные биты. Этого нельзя сказать ни об одном элементе, который приведён ранее в таблице, кроме элемента НЕ.

Это свойство называется *обратимостью*, и это очень важное свойство. В 1961 году Рольф Ландауэр сформулировал принцип и вывел формулу, оцениваю-



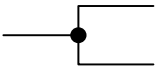
Toffoli.hs

К книге прилагается файл
с исходным кодом на языке
Haskell для демонстрации
свойств элементов Тоффо-
ли и Фредкина

щую нижнюю оценку количества выделяемой теплоты (то есть рассеяния энергии в окружающее пространство) при потере одного бита информации. Это был важный шаг, который позволил связать воедино два вида энтропии — термодинамическую и информационную. И этот принцип сегодня подтверждается необходимостью иметь огромные охлаждающие мощности для серверных в центрах обработки данных — классические вычисления постоянно теряют информацию бит за битом миллиарды раз в секунду, что приводит к выделению большого количества теплоты.

Здесь также будет уместным упомянуть, что эти два рассмотренных элемента также могут быть использованы для реализации так называемого элемента FANOUT, который дублирует поданный на его вход бит. В классической логике наличие этого элемента подразумевается как бы «по умолчанию» (на рис. 5 элемент FANOUT обозначен жирной точкой на линии), однако в обратимых вычислениях эта операция не может быть осуществлена просто так, поскольку если взять её именно такой, какой она подразумевается в классической модели, она будет необратимой. Естественно, что выражение при помощи элементов Тоффоли и Фредкина использует один из выходных битов для того, чтобы дублировать значение одного из входных битов. Например, если на первый вход элемента Тоффоли подать 1, а на третий — 0, то на третьем выходе будет дубликат второго входного бита (который также будет присутствовать на втором выходе). Примерно так же реализуется элемент FANOUT при помощи элемента Фредкина (читателю, как всегда для тренировки ума предлагается подумать, как именно; но в любом случае любознательный читатель сможет найти пример реализации в прилагаемом к книге файле).

Таблица 4. Элемент FANOUT

Функция	Схемотехническое обозначение	Таблица истинности
FANOUT		$f: \{0,1\} \rightarrow \{0,1\}^2$ $f\ 0 = (0, 0)$ $f\ 1 = (1, 1)$

Модель квантовых вычислений основана на квантовой механике, одним из положений которой является обратимость её законов во времени. Любая эволюция квантовой системы обратима во времени, и поэтому данное свойство должно быть непременно отражено в модели квантовых вычислений. А это означает, что любой вычислительный процесс в рамках квантовых вычислений должен быть обратимым.

Так что если в природе имеется какой-либо обратимый процесс, который неким способом производит аналоговые вычисления элемента Тоффоли или элемента Фредкина, то при помощи этого процесса можно построить универсальное вычислительное устройство, которое является обратимым. Собственно, в рамках квантовой механики такие процессы уже найдены. Но перейдём к математическому описанию, поскольку вопросы физической реализации модели квантовых вычислений в этой книге не рассматриваются.

Теперь посмотрим, что это означает с точки зрения математики (а мы в самом начале договорились, что математику мы привлекаем в качестве метанауки, позволяющей манипулировать символами без необходимости вникания в смысл этих символов). В модели квантовых вычислений каждое вычисление осуществляется при помощи так называемых *гейтов*. Гейт в терминах квантовой механики — это «унитарное преобразование», то есть некоторая квадратная матрица U , размерность

которой соответствует количеству базовых квантовых состояний, подчиняющаяся простому свойству:

$$U^\dagger U = U U^\dagger = I$$

где U^\dagger есть эрмитово-сопряжённая матрица к U . Эрмитовое сопряжение является операцией над квадратными матрицами с комплексными числами. Для того чтобы получить эрмитово-сопряжённую матрицу, необходимо транспонировать матрицу и заменить все числа в ней на их комплексно-сопряжённые. Например:

$$\begin{pmatrix} 1 & -3i \\ 2+i & 3 \end{pmatrix}^\dagger = \begin{pmatrix} 1 & 2-i \\ 3i & 3 \end{pmatrix}$$

Ну, в общем-то, представленный пример показывает просто эрмитово-сопряжённую матрицу, но она не является унитарным преобразованием, которое может использоваться в рамках модели квантовых вычислений, поскольку если перемножить эти две матрицы, то в результате точно не получится единичная матрица I (кто из читателей хочет, может проверить и сравнить свой результат с тем, что приведён на врезке на следующей странице).

Понятно, что такие преобразования являются обратимыми в модели квантовых вычислений, поскольку если мы применим к какому-либо кубиту некоторое унитарное преобразование, то для получения первоначального состояния кубита мы можем применить эрмитовое сопряжение этого унитарного преобразования, в результате чего будет получен первоначальный кубит:

$$U^\dagger(U|\varphi\rangle) = U^\dagger U|\varphi\rangle = (U^\dagger U)|\varphi\rangle = I|\varphi\rangle = |\varphi\rangle$$

Можно рассмотреть несколько важных примеров унитарных преобразований (или, как их ещё называют, *унитарных операторов* в гильбертовом пространстве), действующих на одном кубите. К примеру, самым простым, тривиальным оператором является единичная матрица, поскольку эрмитово-сопряжённой

к ней является она сама же, а произведение двух единичных матриц есть единичная матрица.

А вот ещё несколько замечательных матриц, у которых даже есть собственные названия, поскольку они очень часто используются в квантовых алгоритмах. Это так называемые *матрицы Паули*, роль которых в модели квантовых вычислений заключается в поворотах кубитов вокруг различных осей в двумерном пространстве. Вот эти матрицы (единичная матрица I также часто включается в этот набор в качестве матрицы с индексом 0):

$$\begin{pmatrix} 10 & 2 - 10i \\ 2 + 10i & 14 \end{pmatrix}$$

*Результат перемножения
матриц с предыдущей
страницы*

$$\sigma_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \sigma_2 = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \sigma_3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

В рамках модели квантовых вычислений для этих матриц используются более подходящие обозначения: X , Y и Z . Можно посмотреть, что получается, если эти операторы применять к кубитам в стандартном вычислительном базисе. Например, изучим действие матрицы X :

$$X|0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle$$

$$X|1\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle$$

$$X|\varphi\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \beta \\ \alpha \end{pmatrix}$$

Этот унитарный оператор можно считать квантовым аналогом логического элемента НЕ, поскольку он переводит кубит $|0\rangle$ в $|1\rangle$ и наоборот. У суперпозиции базовых вычислительных состояний этот оператор просто меняет местами коэффициенты при базисных состояниях. В общем-то, это можно считать квантовой операцией отрицания.

Теперь рассмотрим действие оператора Y . Примерно то же самое:

$$\begin{aligned} Y|0\rangle &= \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ i \end{pmatrix} \\ Y|1\rangle &= \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} -i \\ 0 \end{pmatrix} \\ Y|\varphi\rangle &= \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} -\beta i \\ \alpha i \end{pmatrix} \end{aligned}$$

Это так называемый оператор сдвига фазы, и у него нет аналогов в традиционной логике. А оператор Z просто меняет знак у коэффициента при базовом состоянии $|1\rangle$ — для того чтобы понять это, не надо даже перемножать матрицы, достаточно просто взглянуть на матрицу этого оператора.

Четыре матрицы Паули (I , X , Y и Z) образуют полный базис унитарных операторов в двумерном гильбертовом пространстве. У них есть ещё множество интересных и важных свойств, которые более подробно можно изучить в специализированной литературе. Ну а читателю в качестве упражнения для разминки мозгов рекомендуется умышленно понять, какие преобразования осуществляют эти операторы над точкой (кубитом) на сфере Блоха.

Теперь можно рассмотреть гейт Адамара, поскольку он очень примечателен тем, что переводит кубиты из стандартного вычислительного базиса в базис $\{|+\rangle, |-\rangle\}$ и обратно. Этот гейт обозначается как H и имеет следующую матрицу:

$$H = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$$

Легко понять, что кубит $|0\rangle$ переводится в $|+\rangle$, а кубит $|1\rangle$ — в $|-\rangle$, и обратно. Этот унитарный оператор часто используется в квантовых алгоритмах, поэтому читателю рекомендуется хорошенько его запомнить, а также для тренировки преобразовать несколько разных кубитов при помощи этого гейта.

Гейты бывают не только однокубитовые (все рассмотренные ранее гейты являются именно однокубитовыми). Само собой разумеется, что унитарное преобразование может воздействовать на систему кубитов произвольного размера. В этом случае размерность матрицы унитарного преобразования равно размерности базиса, то есть 2^k для k кубитов. Вот, к примеру, наиболее важный двухкубитовый гейт, который называется *CNOT* («контролируемое НЕ»):

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Действие этого гейта понять просто — он применяет ко второму кубиту гейт X тогда и только тогда, когда первый кубит равен $|1\rangle$. То есть имеет место система равенств:

$$CNOT|00\rangle = |00\rangle$$

$$CNOT|01\rangle = |01\rangle$$

$$CNOT|10\rangle = |11\rangle$$

$$CNOT|11\rangle = |10\rangle$$

Этот гейт осуществляет, по сути, квантовую условную операцию — условием является значение первого кубита, и если оно равно $|1\rangle$, ко второму кубиту применяется гейт X (квантовое НЕ). Вообще говоря, любое унитарное преобразование U можно сделать контролируемым CU . Для этого достаточно добавить ещё один кубит, от значения которого зависит, приме-

няется ли преобразование U или нет. Если значение контрольного кубита равно $|0\rangle$, то преобразование U не применяется, и все кубиты проходят через него без изменений. В противном случае преобразование U применяется, и результатом является применение этого преобразования к своему множеству кубитов. В виде формул этого записывается следующим образом (\otimes — операция тензорного произведения):

$$CU(|0\rangle \otimes |\varphi\rangle) = |0\rangle \otimes |\varphi\rangle$$

$$CU(|1\rangle \otimes |\varphi\rangle) = |1\rangle \otimes (U|\varphi\rangle)$$

Матрица такого контролируемого унитарного преобразования U выглядит следующим образом (обобщённый вид):

$$CU = \begin{pmatrix} I & 0 \\ 0 & U \end{pmatrix}$$

где I — единичная матрица $2^n \times 2^n$, где n — количество кубитов в регистре $|\varphi\rangle$.

Есть ещё множество интересных гейтов как для одного кубита, так и для многих. Все их описания могут быть найдены в соответствующей литературе, а в этой книге в случае необходимости специальные гейты будут описываться непосредственно в рамках описания квантовых алгоритмов, в которых используются. Теперь же имеет смысл рассмотреть то, как гейты обозначаются. Всё просто — большинство гейтов обозначаются прямоугольником с входами и выходами, причём число выходов в точности равно числу входов (вычисления обратимы). В самом прямоугольнике указывается наименование гейта:

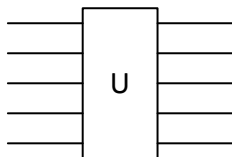


Рис. 6. Условное обозначение произвольного гейта U на квантовых схемах

Если необходимо указать, какие кубиты подаются на тот или иной вход, то эти кубиты указываются непосредственно около входа. Обычно для запуска процесса квантовых вычислений все входные кубиты инициализируются в значение $|0\rangle$, и если около входа не указан кубит (как на рисунке вверху), то считается, что на этот вход подаётся именно значение $|0\rangle$.

Собственно, взглянув на рис. 6 и становится понятным, почему именно функциональное программирование стоит ближе всего к модели квантовых вычислений, чем любая другая парадигма программирования. Гейт U на рис. 6 имеет вполне определённый смысл в рамках функционального программирования — это функция, которая возвращает ровно столько же данных, сколько принимает на вход. То есть гейт — это некий достаточно ограниченный вид функций.

Особое обозначение имеется у гейта $CNOT$, как у «квантового условного оператора». Он обозначается так, как указано на следующем рисунке:

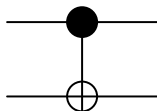


Рис. 7. Условное обозначение гейта $CNOT$

Соответственно, чёрный кружок обозначает контролирующий кубит, то есть в зависимости от значения которого

к контролируемому кубиту применяется квантовая операция отрицания. Соответственно, контролируемый кубит обозначается кружком с крестом, и это легко запомнить, поскольку в логике символом \oplus обозначают логическую операцию «Исключающее ИЛИ» (XOR), а именно так можно определить результат контролируемого кубита: $|b'\rangle = |a\rangle \oplus |b\rangle$, где a — контролирующий кубит, а b — контролируемый. При этом необязательно, что контролирующий кубит находится сверху, на некоторых схемах его удобнее рисовать снизу.

Контролируемый гейт CU обозначается схожим образом:

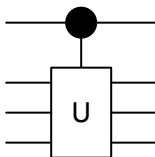


Рис. 8. Условное обозначение гейта CU

Часто в квантовых алгоритмах используются так называемые *оракулы*, которые являются квантовыми аналогами чёрных ящиков. Если есть некоторая функция на двоичных строках $f: \{0,1\}^n \rightarrow \{0,1\}^m$, то соответствующим оракулом для неё будет являться некоторое унитарное преобразование $U_f: \{0,1\}^{n+m} \rightarrow \{0,1\}^{n+m}$, которое считается заданным и вычисляющим эту функцию f . Считается как бы, что есть некоторый физический процесс, который обратимым образом вычисляет эту функцию, осуществляя квантово-механическое унитарное преобразование. Это кажется несколько странным, но для некоторых классов функций такие физические процессы найдены. Обозначаться такой оракул будет следующим образом:

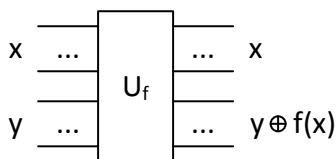


Рис. 9. Условное обозначение оракула U_f

Здесь регистр x является входными данными для функции f и этот параметр занимает n кубитов. Регистр y является вспомогательным (и обычно, но не обязательно, проинициализированным в строку нулей) и занимает m кубитов. На выходе результат вычисления функции складывается по модулю 2 с этим регистром.

С другой стороны квантовая модель вычислений обладает не меньшей вычислительной силой по сравнению с классическими вычислениями. Это значит, что любая функция, которая может быть вычислена при помощи классической схемы, также может быть вычислена и при помощи последовательности унитарных преобразований. Имеется квантовый аналог элемента Тоффоли, и, соответственно, любая бинарная функция может быть построена при помощи преобразования её логической схемы к представлению в виде элементов Тоффоли (ниже будет приведён пример подобного представления). Таким образом, в общем случае для построения квантового оракула для произвольной функции $f: \{0,1\}^n \rightarrow \{0,1\}^m$ можно применить метод её конструирования из базисных элементов. В итоге получается обратимое унитарное преобразование $U_f: \{0,1\}^{n+m} \rightarrow \{0,1\}^{n+m}$, которое можно использовать в алгоритме.

Наконец, процесс (или операция) измерения кубита обозначается стилизованным изображением измерительного прибора:



Рис. 10. Условное обозначение операции измерения

Само собой разумеется, что эти обозначения гейтов являются как бы строительными блоками при рисовании так называемых *квантовых схем*, то есть диаграмм для визуализации квантовых алгоритмов. Вот пример одной квантовой схемы:

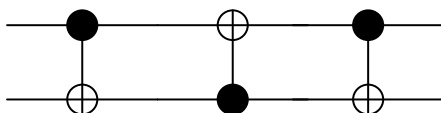


Рис. 11. Пример квантовой схемы

Эта квантовая схема примечательна тем, что меняет местами квантовые состояния двух кубитов, поданных на её вход. Это можно легко проверить для базисных состояний двухкубитовой квантовой системы $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$ при помощи следующей таблицы:

Таблица 5. Результат работы квантовой схемы с рис. 11

Входные кубиты	Результат первого гейта	Результат второго гейта	Результат третьего гейта
$ 00\rangle$	$ 00\rangle$	$ 00\rangle$	$ 00\rangle$
$ 01\rangle$	$ 01\rangle$	$ 11\rangle$	$ 10\rangle$
$ 10\rangle$	$ 11\rangle$	$ 01\rangle$	$ 01\rangle$
$ 11\rangle$	$ 10\rangle$	$ 10\rangle$	$ 11\rangle$

То есть матричное представление этой квантовой схемы выглядит так (надеюсь, читатель к этому моменту в книге уже

вполне может оперировать матричными представлениями кубитов и многокубитовых систем):

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Этот же результат можно получить, если перемножить три матрицы тех унитарных операторов, которые представлены на этой квантовой схеме. Необходимо при этом помнить, что квантовая схема читается слева направо, то есть самый левый гейт применяется к входным кубитам первым, а значит перемножать соответствующие матрицы необходимо как бы в обратном порядке (если три гейта на рис. 11 обозначить слева направо буквами A , B и C , то результат получится при выполнении перемножения $C \times (B \times A)$, или, поскольку операция умножения матриц ассоциативна, просто $C \times B \times A$). Для выполнения этой проверки вдумчивому читателю необходимо очень внимательно выписать матрицу B , поскольку первая мысль, которая может прийти в голову, скорее всего, неправильная.

Также читателю рекомендуется проверить, что данная схема делает с произвольной двухкубитовой системой, находящейся в некоторой суперпозиции четырёх указанных в первом столбце таблицы 5 базисных квантовых состояний.

А вот ещё один пример несложной квантовой схемы:

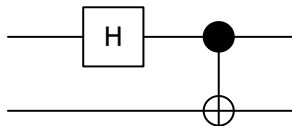


Рис. 12. Ещё один пример квантовой схемы

Эта квантовая схема получает на вход два кубита, которые запутывает между собой так, чтобы на выходе получилось одно из четырёх состояний Белла. Результат работы этой квантовой схемы для базисных состояний отражён в следующей таблице:

Таблица 6. Результат работы квантовой схемы с рис. 12

Входные кубиты	Результат на выходе	Обозначение состояния Белла
$ 00\rangle$	$\frac{1}{\sqrt{2}} 00\rangle + \frac{1}{\sqrt{2}} 11\rangle$	$ \Phi^+\rangle$
$ 01\rangle$	$\frac{1}{\sqrt{2}} 01\rangle + \frac{1}{\sqrt{2}} 10\rangle$	$ \Psi^+\rangle$
$ 10\rangle$	$\frac{1}{\sqrt{2}} 00\rangle - \frac{1}{\sqrt{2}} 11\rangle$	$ \Phi^-\rangle$
$ 11\rangle$	$\frac{1}{\sqrt{2}} 01\rangle - \frac{1}{\sqrt{2}} 10\rangle$	$ \Psi^-\rangle$

Другими словами, квантовые схемы с точки зрения функционального программирования являются ничем иным, как *композицией* специального вида функций. Здесь название операции композиции выделено курсивом специально, поскольку это не совсем та композиция, которая известна в языке Haskell как оператор $(.)$ — здесь имеет место последовательное выполнение унитарных преобразований (то есть в терминах функционального программирования — применение специального рода функций) над кубитами.

Относительно квантовых схем осталось рассмотреть ещё один аспект. На двух предыдущих рисунках приведены квантовые схемы, в которых используются два кубита. Само собой разумеется, что в квантовых схемах может быть задействовано произвольное количество кубитов, и обычно весь набор кубитов называется *квантовым регистром*. Соответственно, неко-

торые унитарные операции действуют на отдельные кубиты в составе регистра, некоторые на часть, а некоторые гейты — на весь квантовый регистр в целом. В этом вопросе у разработчика алгоритма есть самая широкая свобода выбора.

Если на несколько кубитов в квантовом регистре одновременно действуют несколько же гейтов, то такие гейты можно объединять в один, действующий на все кубиты первоначальные сразу. Новый гейт получается просто как тензорное произведение всех гейтов, причём, поскольку тензорное произведение матриц некоммутативно, произведение начинается с самых «верхних» (на схеме) кубитов. Это положение следует из линейности модели квантовых вычислений.

Это положение можно пояснить при помощи следующей диаграммы:

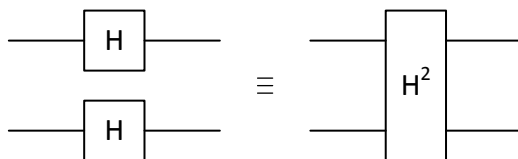


Рис. 13. Тожественность унитарных преобразований

На рис. 13 под меткой « H^2 » имеется в виду тензорное произведение $H \otimes H$ (обычно записываемое как $H^{\otimes 2}$), которое, как и в случае векторов, вычисляется следующим образом. Пусть, как на вышеприведённом рисунке, есть два гейта Адамара, тогда тензорное произведение их матриц будет вычисляться как:

$$\begin{aligned}
& \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 1 & 1 \\ \sqrt{2} & -\sqrt{2} \end{pmatrix} \otimes \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 1 & 1 \\ \sqrt{2} & -\sqrt{2} \end{pmatrix} \\
&= \begin{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 1 & 1 \\ \sqrt{2} & -\sqrt{2} \end{pmatrix} & \frac{1}{\sqrt{2}} \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 1 & 1 \\ \sqrt{2} & -\sqrt{2} \end{pmatrix} \\ \frac{1}{\sqrt{2}} \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 1 & 1 \\ \sqrt{2} & -\sqrt{2} \end{pmatrix} & -\frac{1}{\sqrt{2}} \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 1 & 1 \\ \sqrt{2} & -\sqrt{2} \end{pmatrix} \end{pmatrix} \\
&= \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{pmatrix} \\
&= \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}
\end{aligned}$$

Данный гейт ($H^{\otimes 2}$) преобразовывает пару кубитов, подаваемых к нему на вход, из стандартного вычислительного базиса в базис $\{|+\rangle, |-\rangle\}$. Соответственно, гейт $H^{\otimes n}$ переводит в такой базис n кубитов в стандартном вычислительном базисе, представляя собой матрицу размеров $2^n \times 2^n$.

Квантовая схемотехника

Квантовая схемотехника — это, по сути, методология анализа и, главное, синтеза квантовых схем, реализующих те или

иные алгоритмы (в общем понимании, не только квантовые). Обобщённо любой вычислительный процесс представляется в виде тройки (вход, процесс преобразования, выход). Принимая во внимание это соображение, задачами квантовой схемотехники можно назвать:

1. Прямой анализ. При наличии схемы входа и описания вычислительного процесса определить схему выхода.
2. Обратный анализ. При наличии описания вычислительного процесса и схемы выхода определить схему входа.
3. Синтез. При наличии схем входа и выхода построить описание вычислительного процесса.

К сожалению, в имеющейся литературе по квантовым вычислениям данные вопросы практически не находят своего отражения (от силы есть пара источников на русском языке, где кратко рассматриваются некоторые из них), а именно они являются краеугольным камнем прикладного программирования. Поэтому далее в этом разделе мы постараемся в полной мере раскрыть все три аспекта квантовой схемотехники.

Прямой анализ. С математической точки зрения эта задача решается достаточно просто. Входная схема — это описание множества возможных значений, которые могут быть поданы на вход квантовому вычислительному процессу. Поскольку квантовый регистр, то есть набор кубитов, может быть представлен в виде вектора, а каждый гейт в квантовой схеме представляет собой унитарную матрицу, то задача прямого анализа сводится к последовательному умножению матриц на вектор. А можно сначала перемножить все матрицы в правильном порядке, а затем итоговую матрицу умножить на входной вектор.

Проведя эту процедуру на всех возможных значениях входа (либо применив иные техники анализа), можно получить все выходные значения, после чего объединить их в схему. Данный вид анализа затрудняется тем, что при увеличении количества

кубитов размерность векторов и матриц увеличивается экспоненциально. И если для квантовых регистров, состоящих из 1 — 3 кубитов такую процедуру ещё можно провести, то ручной анализ для четырёх кубитов уже затруднителен, а для десяти кубитов и больше уже сложно проводить и перемножение матриц на компьютере.

Обратный анализ. В рамках модели квантовых вычислений задача обратного анализа тривиально сводится к задаче прямого анализа. Поскольку вычисления являются обратимыми, а матрицы всех гейтов — унитарными, то для обратного анализа заданной квантовой схемы достаточно обратить её, то есть перекоммутировать гейты в обратном порядке, сделав выход входом и наоборот, а сами гейты преобразовать в эрмитово-сопряжённые. После этого проводится описанная ранее процедура прямого анализа.

Само собой разумеется, что вышеприведённые рассуждения применимы только к квантовым схемам, в которых нет операции измерения, которая является необратимой. С другой стороны, измерение обычно применяется в самом конце квантовых вычислений, когда необходимо получить классический результат, поэтому обращение схемы можно осуществить, отбросив операции измерения. И есть совсем немного квантовых алгоритмов, в которых измерение производится в середине процесса вычислений (например, квантовая телепортация), и к таким алгоритмам и их квантовым схемам этот метод обратного анализа не подходит, и надо использовать иные (если они вообще существуют — в общем случае обратная задача неразрешима).

Синтез. Задача построения квантовой схемы по заданным входу и выходу усиливается, по сравнению с классическим случаем, необходимостью обеспечить обратимость вычислений. В общем случае произвольный вычислительный процесс может быть описан как двоичная функция, принимающая

на вход n битов и возвращающая m битов. Такая функция в классическом варианте может быть построена при помощи базисного (универсального) набора логических элементов.

Наличие классической схемы из универсального набора строительных блоков для заданной функции переводит задачу синтеза в задачу построения квантового оракула. Как уже было указано выше, квантовый оракул строится из классического выражения функции следующим образом:

1. Квантовый оракул будет иметь по $(n + m)$ входов и выходов. Первые n входов принимают входные параметры функции, а следующие m входов инициализируются в $|0\rangle$. Соответственно, первые n выходов возвращают входные данные, а следующие m выходов получают сумму по модулю 2 (операция Исключающее ИЛИ) значения функции на заданном входе с m входными данными.
2. Все элементы классической схемы преобразуются в соответствующие квантовые гейты (например, можно воспользоваться универсальным набором из гейтов H и $CNOT$, либо квантовым аналогом элемента Тоффоли). Это может привести к появлению огромного количества так называемых мусорных кубитов.
3. Построенная схема из универсальных квантовых элементов подвергается процессу оптимизации. Это довольно нетривиальный процесс, и в каждом случае применяются свои методы и техники. Цель этого процесса — свести количество мусорных кубитов к минимуму, желательно вообще их исключить.
4. Если от мусорных кубитов избавиться не удалось (пусть их осталось l), то все они добавляются к входным и выходным (стало быть, входов и выходов становится $(n + m + l)$). Но эти кубиты должны быть вначале инициализированы в $|0\rangle$, а по окончании вычислений они также должны быть

установлены в $|0\rangle$. Это следствие законов квантовой механики.

Таким образом, будет построена квантовая схема классической функции. Однако, это не единственный способ. Другим вариантом является построение одной унитарной матрицы для представления полной классической функции. Эта задача может быть успешно разрешена при помощи решения системы уравнений, получаемой из произведения матрицы на вектор. Проблема лишь в том, что количество уравнений и неизвестных растёт экспоненциально от количества кубитов. И если количество входов и выходов равно $(n + m)$, то количество уравнений и неизвестных в них будет равно, как полагается, $2^{2(n+m)}$. Решать такую систему просто для небольших чисел, а вот для больших уже проблематично.

В качестве примера можно рассмотреть построение квантового аналога классической логической операции И. Функция, реализующая эту операцию, имеет 2 входа и 1 выход, следовательно у квантового оракула будет по 3 входа и выхода. Следующая таблица показывает 8 возможных вариантов поведения квантового оракула.

**Таблица 7. Входы и выходы квантового оракула
для операции И**

X_1	X_2	Y	$X_1 \& X_2$	$(X_1 \& X_2) \oplus Y$
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	0	1
1	0	0	0	0
1	0	1	0	1

1	1	0	1	1
1	1	1	1	0

Следовательно, система уравнений, которую надо решить, выглядит следующим образом:

$$I \times \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} & a_{17} & a_{18} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} & a_{27} & a_{28} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & a_{36} & a_{37} & a_{38} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} & a_{47} & a_{48} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & a_{56} & a_{57} & a_{58} \\ a_{61} & a_{62} & a_{63} & a_{64} & a_{65} & a_{66} & a_{67} & a_{68} \\ a_{71} & a_{72} & a_{73} & a_{74} & a_{75} & a_{76} & a_{77} & a_{78} \\ a_{81} & a_{82} & a_{83} & a_{84} & a_{85} & a_{86} & a_{87} & a_{88} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Её решение тривиально, и результатом будет следующая матрица:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Эта матрица и есть квантовый оракул в матричном представлении. Теперь необходимо убедиться, что она унитарна. Для этого надо умножить её на эрмитово-сопряжённую. Поскольку эта матрица является самосопряжённой, то её надо просто возвести в квадрат. Несложные умозаключения дают

понять, что результатом будет единичная матрица, то есть найденная матрица является унитарной.

Краткие пояснения. Единичная матрица в исходном уравнении получается из восьми строк таблицы 8, в которых выделены столбцы 1, 2 и 3. Каждая из этих восьми строк преобразуется в вектор-столбец (здесь ради экономии показаны векторы-строки, но надо помнить, что это векторы-столбцы). Строка «0 0 0» представляет собой вектор (1 0 0 0 0 0 0 0), строка «0 0 1» представляет вектор (0 1 0 0 0 0 0 0) и т. д. до строки «1 1 1», которая есть вектор (0 0 0 0 0 0 0 1). Результирующая матрица строится по тому же принципу, однако теперь из таблицы 8 берётся строки со столбцами 1, 2 и 5. Как видно, столбец 5 отличается от столбца 3 только в 7-ой и 8-ой строке, отсюда и перемена 7-го и 8-го столбцов в результирующей матрице.

Другими словами, для получения результирующей матрицы квантового оракула достаточно собрать векторы-столбцы по тому же принципу, как это описано в предыдущем абзаце. Для n входных кубитов берётся 2^n строк, в каждой из которых $(n + 1)$ столбец. Первые n столбцов принимают все возможные значения входных кубитов, а последний столбец — значение $(y \oplus f(\vec{x}))$. Для каждой строки ставится в соответствие вектор-столбец в вычислительном базисе, а потом все столбцы собираются в матрицу.

А можно воспользоваться и более простым методом. Для этого каждой строке матрицы сопоставляется входной кубит (сверху вниз), а каждому столбцу матрицы сопоставляется выходной кубит (слева направо). В самой матрице значение 1 ставится на тех местах, которые обозначают преобразование входного кубита в выходной, а на остальных позициях ставится значение 0. Для быстрого построения оракулов этот метод

проще всего. Ну и можно при желании доказать его тождественность предыдущему методу.

Задача решена. Хорошим упражнением для заинтересованного читателя будет построение таких же матриц для других пятнадцати двоичных функций от двух переменных.

При всём описанном необходимо понимать, что синтез квантовой схемы — это не построение квантового алгоритма. Показанная здесь техника позволяет решать на квантовом компьютере произвольную вычислительную задачу. А разработка квантового алгоритма — это дело настолько необычное и нетривиальное, что чаще помогает озарение, нежели скрупулёзное построение шаг за шагом. На текущий день все разработанные квантовые алгоритмы базируются на нескольких базовых, которые были найдены именно при помощи озарения.

Принципы квантовых вычислений

Теперь осталось перейти к перечислению основополагающих принципов, на которых зиждется модель квантовых вычислений. Знание и отчётливое понимание сути этих принципов позволит в дальнейшем легко и бегло знакомиться с более глубокой литературой по квантовым вычислениям.

Обратимость вычислений — главный принцип, из которого, в общем-то, следуют все остальные. Квантовые вычисления обратимы во времени, а это значит, что по результату любого вычисления можно восстановить исходные данные. Более того, для любой квантовой схемы можно построить дуальную схему, которая будет получать на вход результат работы первоначальной схемы и возвращать исходные данные для неё. Для этого достаточно всего лишь перевернуть схему справа налево, а все унитарные преобразования заменить на эрмитово-сопряжённые.

Поскольку все унитарные преобразования являются обратимыми, все они имеют ровно столько же выходов, сколько и входов — не меньше, но и не больше (если бы было больше, то эрмитово-сопряжённое преобразование имело бы меньше выходов, а, значит, теряло бы информацию и, стало быть, не было бы обратимым).

Из этих рассуждений следует, что квантовые вычисления страшно *избыточны*. Ведь, например, самые простые логически обратимые элементы Тоффоли и Фредкина имеют по три входа и три выхода. Если эти элементы используются для получения, скажем, элемента НЕ, то по два входа и выхода будут незадействованными. А для элементов И, ИЛИ и других неиспользуемыми будет один вход и два выхода. Для элемента FANOUT наоборот два входа использоваться не будут, равно как и один выход.

Это можно проиллюстрировать следующей диаграммой. На рис. 14 показано выражение элемента Тоффоли через элемент Фредкина. Из всего огромного множества входных и выходных битов используются только по три с каждой стороны.

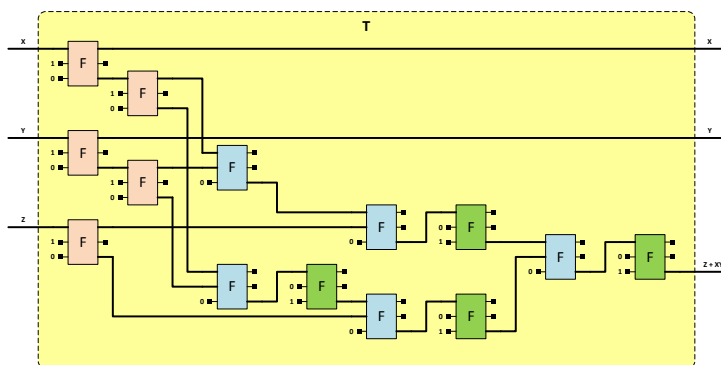


Рис. 14. Диаграмма, показывающая один из возможных вариантов выражения элемента Тоффоли через элемент

Фредкина (красные элементы — FANOUT, синие — И, зелёные — НЕ)

Данный принцип влечёт, что в процессе работы квантовых алгоритмов будет накапливаться огромное количество «мусорных» битов. Если их начать уничтожать (например, при помощи своеобразной «сборки мусора»), то начнёт выделяться огромное количество теплоты, а вычисления станут необратимыми.

Собственно, на рис. 14 видно, что для выражения одного элемента с тремя входами и тремя выходами по обратимой схеме пришлось задействовать 14 базовых элементов, и общее число входов и выходов на схеме стало равным по 26, из которых только по 3 входа и выхода используются, а остальные являются вспомогательными и, по сути, «мусорными».

Есть мнение, что за экспоненциальное ускорение решения некоторых задач, которое даёт модель квантовых вычислений, придётся заплатить экспоненциальным увеличением размера задействованной памяти, и ситуация с выражением элемента Тоффли через элемент Фредкина это отчасти иллюстрирует. Впрочем, для решения этой проблемы был разработан метод, который позволяет уничтожать промежуточные «мусорные» биты без потери обратимости вычислений. Метод этот математически достаточно прост, и заинтересованный читатель сможет найти его описание в специализированной литературе.

Однако, конечно же, при более вдумчивом подходе, когда инженер или программист включает своё воображение и техническую смётку, можно получить более экономные способы выражения одних элементов через другие. В частности, рассмотренный пример можно преобразовать таким образом, что для выражения элемента Тоффли через элементы Фредки-

на потребуется всего 4 последних. Из этих четырёх элементов формируется схема, в которой есть 6 выходов, однако все они могут быть задействованы. Три выхода являются непосредственным решением задачи, а три других могут использоваться в дальнейшем, поскольку тем или иным способом преобразуют входные биты. Вдумчивому читателю предлагается самостоятельно найти такую схему выражения элемента Тоффоли (это просто).

Всё это обозначает, что в квантовых схемах *нет и не может быть циклов и возвратов назад*. Кубиты как бы двигаются по «проводам», проходя через гейты и преобразовываясь в соответствии с предназначением гейта. Поток управления квантовой программой движется от начала алгоритма только вперёд. Унитарное преобразование и контролируемое унитарное преобразование — вот единственные способы выполнения вычислений. Измерение — единственный способ получения результата, который, к тому же, «схлопывает» суперпозицию, в которой находятся кубиты.

По поводу обратимых вычислений остаётся отметить, что одним из немаловажных аспектов является обработка ненужных выходов. Если просто уничтожать результаты их вычислений, то такое обращение ничем не будет отличаться от обычной классической вычислительной модели, поскольку при уничтожении информации выделяется тепло (энергия, задействованная в процессе хранения и передачи информации рассеивается). Выделение тепла делает обратимые вычисления необратимыми, а саму модель квантовых вычислений бесполезной. Поэтому при разработке квантовых схем используются специальные методы, которые собирают все неиспользованные выходы и преобразуют их специальным образом так, чтобы они использовались (например, для вычисления обратной функ-

ции), а весь процесс вычислений и его квантовая схема были полностью обратимыми.

Следующим принципом является *квантовый параллелизм*. Система кубитов в процессе прохождения через гейты унитарных преобразований параллельно решает одну и ту же задачу для огромного, экспоненциально большого количества данных из-за того, что кубиты находятся в суперпозиции базисных состояний. Поскольку с ростом числа кубитов размерность базиса растёт в степенной зависимости, квантовый параллелизм обладает огромной вычислительной мощностью. Главное — уметь правильно пользоваться этим принципом.

С параллелизмом тесно связана *интерференция*. Некоторые алгоритмы используют интерференцию квантовых состояний для того, чтобы взаимно усилить требуемый результат и наоборот сгладить результат нежелательный. Повторяя несколько раз последовательность параллельной обработки кубитов с интерференцией их состояний, разработчик алгоритма может так усилить амплитуду искомого состояния, что дальнейшее измерение даст требуемый результат с высокой вероятностью (варьируя количество повторений шага с интерференцией, можно управлять уровнем вероятности, доводя его до любого заданного значения).

Квантовая запутанность — ещё один принцип, причём наименее изученный, равно как и наименее поддающийся рациональному осмыслению. Учёные уже больше века ломают головы и копы на тему запутанности квантовых систем (взять, хотя бы, спор Н. Бора с А. Эйнштейном на эту тему). Но именно квантовая запутанность позволяет решить многие хитрые задачи, являясь ключевым фактором многих квантовых алгоритмов.

Общая архитектура квантового компьютера

Было бы странным полагать, что квантовые компьютеры, когда они будут реализованы в «железе», будут представлять собой некие устройства, похожие на компьютеры современные. Уже тот принцип, который говорит о недопустимости циклов и возвратов потока управления программой, намекает на то, что квантовый компьютер будет каким-то иным устройством. А есть ещё взаимодействие с пользователем, работа с периферийными устройствами, работа по сети с другими компьютерами — всё это вряд ли возможно будет реализовать в рамках модели квантовых вычислений, да и не будет такой необходимости, поскольку классические компьютеры очень неплохо справляются с этими задачами.

Скорее всего, все разработанные и реализованные к моменту создания квантового компьютера алгоритмы будут упакованы в некий «Фонд алгоритмов и программ», библиотеку квантовых алгоритмов, которая будет связана с устройством, выполняющим квантовые вычисления. Вполне вероятно, что это будет некий «квантовый сопроцессор», которым управляет обычный процессор при помощи каких-либо хитроумных устройств, которые позволяют выполнять две основные задачи: подготовка необходимых унитарных преобразований для выбранного квантового алгоритма и инициализация входных кубитов. На выходе этого «квантового сопроцессора» будут производиться измерения, а их результат сообщаться основному процессору, производящему вычисления.

Так что от взаимодействия с внешним миром, циклов и возвратов назад по некоторым условиям никуда не деться — всё это очень мощные идиомы программирования, чтобы от них отказываться. Но выполнять эти операции будет класси-

ческий компьютер, а квантовому сопроцессору останется только по управляющему вызову из основного вычислительного устройства запускать определённый алгоритм из библиотеки и возвращать результат его работы.

Всё это позволяет нарисовать примерно следующую общую архитектуру, в рамках которой будут производиться квантовые вычисления:

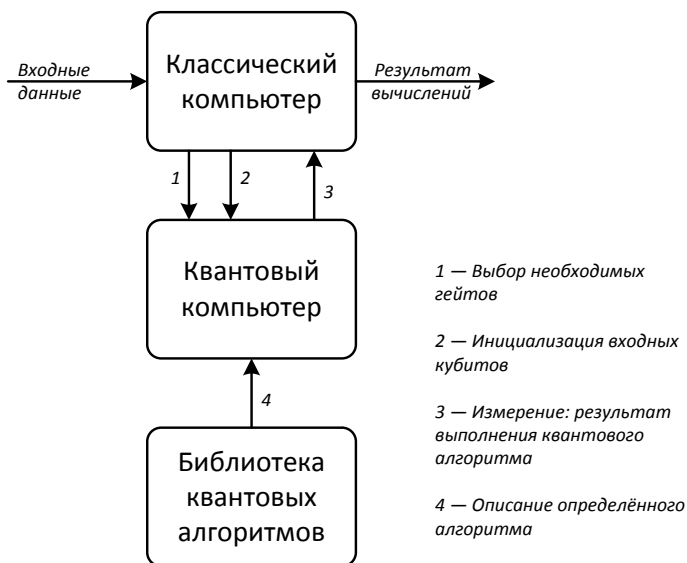


Рис. 15. Общая архитектура квантового компьютера

Исходя из этой диаграммы, можно дальше попытаться пофантазировать на тему того, как будет развиваться отрасль квантового программирования. Оставив в стороне «физиков», которые занимаются сейчас и будут заниматься в будущем поиском методов реализации квантового вычислительного устройства в «железе», можно разделить программистов на два больших класса:

- *Системные квантовые программисты* будут создавать новые изощрённые квантовые алгоритмы для размещения их в библиотеке квантовых алгоритмов и последующего использования в работе всей системы.
- *Прикладные квантовые программисты* будут использовать квантовые алгоритмы на практике, решая при помощи них прикладные задачи, но при этом они должны будут знать как номенклатуру алгоритмов в библиотеке, так и понимать смысл их работы, чтобы оптимальным образом производить их вызов и использование их результатов в прикладных программах для обычного компьютера.

Таким образом, программировать само квантовое вычислительное устройство никто не будет; его будет «программировать» классический компьютер — настраивать гейты, связывать их друг с другом, инициализировать кубиты и запускать квантовые вычисления. Как именно это будет реализовано, сейчас пока ещё никто не знает.

Краткие выводы

Модель квантовых вычислений — это некоторая математическая система, которая позволяет производить произвольные вычисления, причём уже показано, что для некоторых задач эта модель позволяет реализовать намного более эффективные алгоритмы, чем традиционная вычислительная модель, основанная на машине Тьюринга или лямбда-исчислении. Реализация этой новой вычислительной модели в «железе» не имеет фундаментальных преград, так что дело только за технологиями.

Сама модель сводится к выполнению ряда унитарных преобразований векторов в 2^n -мерном гильбертовом пространстве, где n — количество кубитов, а каждое унитарное преобразование представляет собой матрицу специального вида. Таким образом, очень упрощённо, можно сказать, что по своей сути

квантовые вычисления заключаются в перемножении матрицы размера $2^n \times 2^n$ (которая сама ещё должна быть получена как произведение целого набора матриц такого же или меньших размеров) на вектор размера 2^n . Эти вычисления, конечно же, можно выполнить и на обычных компьютерах, но сложность такого умножения будет расти экспоненциально в зависимости от количества моделируемых кубитов.

Реализация квантового компьютера позволит решить проблему экспоненциального роста сложности, при этом такой компьютер будет гибридным — скорее всего, аналоговое квантовое вычислительное устройство будет управляться обычным цифровым компьютером.

Глава 2.

Фреймворк для квантовых вычислений

Хоть мы и говорим на каком-то определённом языке и используем определённые концепции, отсюда вовсе необязательно следует, что в реальном мире имеется что-то, этим вещам соответствующее.

Роберт Оппенгеймер

После получения базовых знаний о том, что такое модель квантовых вычислений, можно перейти к реализации небольшого фреймворка на языке программирования Haskell для более глубокого вхождения в суть вопроса. В этой главе по шагам описывается реализация такого фреймворка, причём

для дидактических целей реализация осуществляется постепенно в том же порядке, как термины и определения вводились в предыдущей главе.

В конце главы будут предложены решения при помощи реализованного фреймворка нескольких более или менее интересных задач.

Квантовые состояния

Прежде всего необходимо реализовать модуль, в котором описывается базовое понятие модели квантовых вычислений — квантовое состояние. Пусть это будет модуль `QuantumState`, и в нём должен описываться тип данных для представления квантовых состояний, а также базовые функции работы с ними.

Поскольку мы договорились называть квантовым состоянием некоторое помеченное комплексное число, то отсюда с необходимостью следует соответствующее определение алгебраического типа данных на языке `Haskell` для его представления:

```
data QuantumState a = QS
    {
        amplitude :: Complex a,
        label      :: String
    }
```

Здесь определяется запись с именованными полями, и поле `amplitude` представляет комплексную амплитуду квантового состояния, а поле `label` представляет пометку при амплитуде. Поскольку здесь у нас используется тип `Complex`, необходимо импортировать модуль `Data.Complex` из стандартной поставки. Как видно, этот тип параметризован типовой переменной `a`, так что у нас будет прекрасная возможность использовать при решении практических задач различные числовые типы данных, при помощи которых задаются действительная

и мнимая части комплексного числа. Для разных задач это могут быть целые числа, рациональные числа или действительные числа.

Тонкий момент — почему для поля `label` использован тип `String`? Ведь можно было бы использовать для квантового состояния метку в виде символа, то есть взять здесь тип `Char`. Такое решение сделано специально, и в разделе, посвящённом кубитам, оно будет полностью описано и оправдано.

Теперь можно озаботиться реализацией для созданного типа экземпляра класса `Show` для представления квантового состояния в виде строки. Здесь мы будем использовать нотацию Дирака, поэтому надо написать своё экземпляра. Кроме того, стандартное представление комплексных чисел в модуле `Data.Complex` не совсем обычно, поэтому его мы тоже перепишем.

Вот так выглядит экземпляр класса `Show` для типа `QuantumState`:

```
instance (Eq a, Ord a, Num a, Show a) =>
  Show (QuantumState a) where
  show (QS a l) = brackets ++ "|" ++ l ++ ">"
  where
    brackets = if realPart a /= 0 && imagPart a /= 0
              then "(" ++ prettyShowComplex a ++ ")"
              else prettyShowComplex a
```

Здесь используется функция `prettyShowComplex`, которая как раз и необходима для нормального представления комплексного числа. Вот её определение:

```
prettyShowComplex :: (Eq a, Ord a, Num a, Show a) =>
  Complex a -> String
prettyShowComplex (r :+ 0) = prettyShowNumber r
prettyShowComplex (0 :+ i)
  | i == 1    = "i"
  | i == -1   = "-i"
  | otherwise = prettyShowNumber i ++ "i"
```

```
prettyShowComplex (r :+ i)
| i == 1    = prettyShowNumber r ++ " + i"
| i == -1   = prettyShowNumber r ++ " - i"
| i >= 0    = prettyShowNumber r ++ " + " ++
              prettyShowNumber i ++ "i"
| otherwise = prettyShowNumber r ++ " - " ++
              prettyShowNumber (abs i) ++ "i"
```

Здесь нет никаких эдаких особенностей. Если у заданного комплексного числа отсутствует мнимая часть, то в строку преобразуется только действительная часть (даже если это 0, поэтому данный клонз определения функции стоит первым). Если же у комплексного числа отсутствует действительная часть, но присутствует мнимая, то в строку преобразуется только мнимая, при этом добавляется знак мнимой единицы. В случае, если у заданного комплексного числа присутствует и действительная, и мнимая части, то выясняется знак мнимой части и в зависимости от него число преобразуется в строку либо как « $n + mi$ », либо как « $n - mi$ », где n и m — действительная и мнимая части соответственно.

Кроме того, в этой функции также производится проверка того факта, является ли мнимая часть положительной или отрицательной единицей. Если является, то число «1» не выводится в строку, так как «i» — уже единица, а также ставится соответствующий знак (плюс или минус).

Дополнительная функция `prettyShowNumber` используется для того, чтобы специальным образом преобразовать число в строку. Функции реализованы наиболее общим способом, поэтому для представления комплексных амплитуд в общем случае будут использоваться действительные числа. Однако часто в задачах и проблемах используются целые числа, которые в разрабатываемом фреймворке также будут представляться действительными числами. Но чтобы не выводить после десятичной точки ноль, как раз используется функция `prettyShowNumber`, чьё определение незамысловато:

```
prettyShowNumber :: (RealFrac a, Show a) => a -> String
prettyShowNumber f = if d == 0.0
                      then show n
                      else show f

where
  (n, d) = properFraction f
```

В самом определении метода `show` для экземпляра класса `Show` типа `QuantumState` есть дополнительная проверка того, имеются ли у амплитуды обе части — действительная и мнимая, и если это так, то само число берётся в круглые скобки. Это понадобится, когда в строку будут преобразовываться целые кубиты, а не одиночные квантовые состояния. В итоге квантовые состояния выводятся на экран в примерно таком виде: « $1|0\rangle$ », « $i|1\rangle$ », « $(2 - i)|+\rangle$ » и т. д.

Для быстрого создания квантовых состояний можно использовать стандартный тип, представляющий собой пару величин. Также можно написать двойственную для неё функцию. Вот их определения:

```
toPair :: QuantumState a -> (Complex a, String)
toPair (QS a l) = (a, l)

fromPair :: (Complex a, String) -> QuantumState a
fromPair (a, l) = QS a l
```

Собственно, осталось реализовать только лишь одну функцию служебного характера для получения комплексно-сопряжённого квантового состояния для заданного. В этом нет ничего сложного, поскольку в модуле `Data.Complex` есть функция для получения комплексно-сопряжённого числа для заданного. Но мы поступим мудрее, поскольку идиоматичность языка Haskell обязывает.

Сначала определим функцию высшего порядка для применения заданной функции к комплексной амплитуде заданного квантового состояния. Это просто:

```
applyQS :: (Complex a -> Complex a) ->
           QuantumState a -> QuantumState a
applyQS f (QS a l) = QS (f a) l
```

Будет полезным написать такую же функцию, только для предикатов — предикат «втягивается» в квантовое состояние и применяется к амплитуде его вероятности, но при этом при помощи функции высшего порядка сам предикат как бы применяется на уровне выше к квантовому состоянию. Эта функция пригодится, и её определение таково:

```
predicateQS :: (Complex a -> Bool) ->
              QuantumState a -> Bool
predicateQS p (QS a _) = p a
```

Ну а теперь очень легко реализовать функцию для возвращения комплексно-сопряжённого квантового состояния:

```
conjugateQS :: RealFloat a =>
              QuantumState a -> QuantumState a
conjugateQS = applyQS conjugate
```

На этом модуль для описания квантовых состояний можно считать законченным.



QuantumState.hs

*К книге прилагается файл
с исходным кодом на языке
Haskell с полным описанием
модуля для квантовых со-
стояний.*

Кубиты

Перейдём к кубитам. Как было определено в предыдущей главе, кубитом называется просто список квантовых состояний. Вот здесь-то и кроется причина, почему поле `label` в определении типа `QuantumState` имеет тип `String`. Дело всё в том, что с точки зрения реализации нет никакой разницы, есть ли у кубита два базисных состояния, или их произвольное число (но, как известно, равное степени двойки). Чем один кубит в двухмерном гильбертовом пространстве отличается от многокубитовой системы в 2^n -мерном пространстве? Да ничем!

Поэтому определение кубита в данном случае будет выглядеть как изоморфный тип, скрывающий в себе список квантовых состояний:

```
newtype Qubit a = Qubit [QuantumState a]
```

И теперь при помощи этого типа можно вполне себе представлять многокубитовые квантовые системы, у которых пометки у индивидуальных квантовых состояний уже не являются просто символами, но есть строки (например, `|01>`).

Перво-наперво для этого нового типа можно реализовать экземпляр класса `Show` для преобразования в строки и приятного отображения на экране. Имея уже реализованный экземпляр этого же класса для типа `QuantumState`, новый экземпляр реализовать очень просто:

```
instance (RealFloat a, Show a) => Show (Qubit a) where
  show (Qubit []) = ""
  show (Qubit [qs]) = show qs
  show (Qubit (qs:qss)) = show qs ++ " + " ++
    show (Qubit qss)
```


Всё это будет находиться в отдельном от предыдущего модуле, который будет называться так же, как и главная сущность, описываемая в нём, — `Qubit`.

Как и в случае с квантовым состоянием для кубита полезно реализовать функции для его создания из списка. Однако для кубита таких функций может быть несколько, поскольку для различных ситуаций могут использоваться списки амплитуд (обычно), а иногда нужны списки меток. В общем случае может потребоваться список пар вида (амплитуда, метка). Так что имеет смысл реализовать три пары функций (прямую и двойственную к ней):

```
toList :: Qubit a -> [Complex a]
toList = map amplitude . quantumStates

toLabelList :: Qubit a -> [String]
toLabelList = map label . quantumStates

fromLists :: [Complex a] -> [String] -> Qubit a
fromLists a l = Qubit $ zipWith QS a l

toPairList :: Qubit a -> [(Complex a, String)]
toPairList = map toPair . quantumStates

fromPairList :: [(Complex a, String)] -> Qubit a
fromPairList = Qubit . map fromPair
```

Для функций `toList` и `toLabelList` есть единственная двойственная функция `fromLists`, которая получает на вход два списка — список комплексных амплитуд и список меток.

Однако преобразование кубита в список — это слишком просто, равно как и создание кубита из списка. Все эти функции являются служебными, годящимися только для упрощения процесса построения описаний кубитов. В математике квантовой механики, как это было указано в первой главе, есть векторные представления кубитов. Имеет смысл реализовать функции для преобразования кубита во внутреннем представ-

лении (тип `Qubit`) в векторное и обратно. Здесь будет показано немного функциональной магии, но совсем немного, поскольку настоящие функции для таких преобразований должны отслеживать многочисленные нестандартные ситуации, а в приводимом здесь определении все они либо остаются за реализацией, либо смахиваются в вызов системной функции `error`, которая останавливает исполнение программы.

Итак, вот определение функции, которая переводит кубит в векторное представление:

```
toVector :: Num a => Qubit a -> [Complex a]
toVector q = if all `elem` "01" labels
              then map (fromMaybe (0 :+ 0) .
                        flip lookup qsPairs) basis
              else error "Некорректные метки кубита."

where
  n      = length $ label $ head $ quantumStates q
  labels = concatMap label $ quantumStates q
  qsPairs = map (swap . toPair) $
             sortBy (comparing label) $
             quantumStates q
  basis  = replicateM n "01"
```

Для начала необходимо рассмотреть все локальные определения. Замыкание `n` представляет вычисление количества кубитов в квантовой системе, поданной на вход. Здесь есть логическая ошибка, поскольку если в поданном на вход описании кубита есть метки разной длины (чего быть не может, но фреймворк этого не проверяет), то в качестве значения `n` возьмётся длина первой метки. Также вычисление этого замыкания небезопасно, поскольку программа может вывалиться, если у кубита вообще нет квантовых состояний (этого тоже быть не может, но фреймворк опять же не отслеживает).

Второе замыкание `labels` возвращает сцепленные метки всех квантовых состояний кубита. Это служебное замыкание, которое используется только в проверке того, что в метках

квантовых состояний кубита используются только символы из вычислительного базиса, то есть «0» и «1». Опять же, если так получилось, что у кубита нет квантовых состояний, то функция в целом отрабатывает некорректно (вернёт пустое векторное представление).

Замыкание `qsPairs` возвращает все квантовые состояния кубита, отсортированные по меткам и преобразованные в пары (то есть получился ассоциативный список), причём первым элементом пары идёт метка, а вторым — амплитуда. Это требуется для поиска недостающих квантовых состояний, если кубит представляет собой смешанные квантовые состояния.

Ну и, наконец, замыкание `basis` возвращает все возможные строки из символов «0» и «1» длиной n . Здесь использована монадическая функция `replicateM` для монады списка, которая и решает эту задачу. В итоге для $n = 2$, например, получается список `["00", "01", "10", "11"]`.

Сама функция `toVector` берёт этот список базисных квантовых состояний (`basis`) и каждое из них ищет в ассоциативном списке, полученном из кубита (`qsPairs`). Если метка найдена, то в результат записывается соответствующая амплитуда. Если метка не найдена, то в результат записывается число 0. Так что, например, кубит $|\Phi^+\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$ будет преобразован в список $\left[\frac{1}{\sqrt{2}}, 0, 0, \frac{1}{\sqrt{2}}\right]$.

Как ни странно, обратная функция `fromVector` по сравнению с только что определённой, выглядит совсем просто. Единственный нюанс — чтобы не вычислять логарифм по основанию 2, вторым параметром в неё необходимо передавать количество кубитов в квантовой системе — это число требуется для автоматической генерации всех необходимых меток в вычислительном базисе. Если сделать определение таким, то оно выглядит совсем просто:

```
fromVector :: Int -> [Complex a] -> Qubit a
fromVector n q = fromLists q $ replicateM n "01"
```

Здесь видно то же самое построение списка всех возможных комбинаций строк длины n из символов «0» и «1», на основании которого далее строится кубит при помощи уже определённой функции `fromLists`. Само собой разумеется, что длина векторного представления кубита должна быть в точности соответствующей. Фреймворков это опять же не проверяется.

Важной сервисной функцией, которая исполняет примерно те же действия, что и функция `applyQS` для квантовых состояний, является функция высшего порядка для «втягивания» заданной функции в кубит и применения её к множеству его квантовых состояний. Вот её определение:

```
liftQubit :: ([QuantumState a] -> [QuantumState b])
            -> Qubit a -> Qubit b
liftQubit f (Qubit qs) = Qubit $ f qs
```

Чуть ли не главной функцией для обработки кубитов, которая должна быть описана в этом модуле, является функция `entangle`, которая получает на вход два кубита, а возвращает один, представляющий собой сцепленную квантовую систему из двух входных кубитов. На удивление её определение очень простое:

```
entangle :: RealFloat a => Qubit a -> Qubit a -> Qubit a
entangle (Qubit qss1) (Qubit qss2)
  = Qubit $ filter (predicateQS (/= 0))
    [QS (((*) `on` amplitude) qss1 qss2)
      (((++) `on` label) qss1 qss2) | qss1 <- qss1,
                                     qss2 <- qss2]
```

По сути, это тензорное произведение двух кубитов в векторном представлении. С точки зрения реализации здесь берутся все квантовые состояния первого кубита и все квантовые состояния второго кубита, для каждой пары квантовых со-

стояний вычисляется новая амплитуда при помощи умножения амплитуд, а символ для этой амплитуды собирается как конкатенация символов первого и второго кубитов (поэтому операция некоммутативна). В результате выполнения этой операции получается многокубитовая система с количеством квантовых состояний, равным произведению количеств квантовых состояний входных кубитов, но из этого результирующего списка квантовых состояний удаляются все те, у которых получается нулевая амплитуда.

Теперь необходимо реализовать несколько функций для работы с различными кубитами. В первую очередь потребуется функция для получения из кет-вектора заданного кубита соответствующий ему бра-вектор. Понятное дело, что эта функция определяется проще простого:

```
conjugateQubit :: RealFloat a => Qubit a -> Qubit a
conjugateQubit = liftQubit (map conjugateQS)
```

Далее потребуется функция для получения скалярного произведения, то есть для выполнения операции $\langle \varphi | \psi \rangle$. Проще всего реализовать её при помощи преобразования кубитов в векторное представление. После этого все компоненты векторов можно попарно перемножить, результирующий список произведений сложить, после чего взять только действительную часть (мнимая, естественно, будет равняться 0):

```
scalarProduct :: RealFloat a => Qubit a -> Qubit a -> a
scalarProduct q1 q2 = realPart $
    sum $
        ((zipWith (*)) `on` toVector) q1 q2
```

Полезной функцией будет функция `norm`, которая вычисляет норму кубита, то есть длину вектора в гильбертовом пространстве. Поскольку функция для вычисления скалярного произведения у нас уже есть, равно как и функция для получения бра-вектора, определение функции `norm` становится чуть ли не тривиальным:

```
norm :: RealFloat a => Qubit a -> a
norm q = sqrt $ scalarProduct q (conjugateQubit q)
```

Поскольку для модели квантовых вычислений принято, что все кубиты должны быть нормированы, должна быть функция нормализации. В принципе, сложно представить себе ситуацию, когда есть нормированный кубит, который далее подвергается унитарным преобразованиям (которые все суть просто вращения векторов в гильбертовом пространстве, не изменяющие нормы), после чего кубит необходимо будет заново нормализовать. Однако такая функция видится полезной по причине того, что разработчик квантового алгоритма может конструировать описания кубитов некоторым специальным образом, имея промежуточные результаты, подвергаемые в конце вычислений нормализации. Поэтому функцию определим:

```
normalize :: RealFloat a => Qubit a -> Qubit a
normalize q =
  liftQubit (map (applyQS (/ (norm q :+ 0)))) q
```

Наконец, в фреймворке должна быть функция для осуществления операции измерения кубита. Эта функция недетерминированная, поскольку возвращает вероятностный результат в зависимости от амплитуд вероятностей всех входящих в кубит квантовых состояний. Поскольку распределение вероятностей неравномерное, а зависит именно от амплитуд вероятностей, необходимо реализовать служебную функцию для случайного выбора некоторого значения из списка, при этом вероятность выбора зависит от заданных значений вероятности.

Эта функция на первый взгляд не так проста:

```
getRandomElementWithProbabilities ::
  (Ord b, Num b, Random b) => [(a, b)] -> IO a
getRandomElementWithProbabilities l =
  (head . goodList) `fmap` randomRIO (0, sumProbs l)
where
  goodList p = map fst $
```

```
dropWhile (\(_, p') -> p' < p) $
map ((fst . last) &&& sumProbs) $
tail $
  inits l
sumProbs = sum . map snd
```

Здесь определена функция, которая получает на вход список пар. Первым элементом каждой пары стоит некоторое значение, а вторым — вероятность его проявления. Функция возвращает случайно выбранное значений среди всех первых элементов в зависимости от распределения вероятностей. И тут есть интересная особенность — сумма вероятностей в списке пар не обязательно должна равняться 1, поскольку в функции производится неявная нормализация.

В локальной функции `sumProbs` производится расчёт суммы всех вероятностей в заданном списке. Далее при помощи стандартной функции `randomRIO` выбирается случайное число от 0 до суммы всех вероятностей. Оно возвращается в монаде IO, поэтому для применения локальной функции `goodList` и последующего взятия головы результирующего списка используется метод `fmap`. А что же такое `goodList`?

Локальная функция `goodList` возвращает список всех значений, вероятность появления которых больше заданной. Этот список строится следующим образом:

- Сначала при помощи функции `inits` из модуля `Data.List` строится список всех начал исходного списка пар.
- Полученный на предыдущем шаге список начал первым элементов содержит пустой список, поэтому он выкидывается при помощи функции `tail`.
- Затем при помощи оператора `(&&&)` из модуля `Control.Arrow` строится список таких же пар, но уже для каждого элемента новое значение вероятности равно сумме вероятностей из исходного списка для всех элементов, лежащих раньше по списку, включая его самого. Выражение

`((fst . last) &&& sumProbs)` возвращает пару, первым элементом которой является первый элемент последней пары в списке, а вторым — сумма вероятностей всех элементов до него, включая его самого.

- Далее при помощи функции `dropWhile` от списка отсекаются все такие начальные элементы, вероятность которых меньше, чем заданная входным аргументом.
- В конце концов, из списка удаляются сами вероятности, и список пар преобразуется в список простых значений.

Вот именно из такого списка и берётся головной элемент, который и является искомым для отображения на случайное число.

Теперь можно реализовать функцию для измерения кубита. При наличии функции `getRandomElementWithProbabilities` её определение становится делом простым:

```
measure :: (RealFloat a, Random a)
        => Qubit a -> IO String
measure q =
  getRandomElementWithProbabilities $
    sortBy (compare `on` snd) $
    map (swap . \(a, l) ->
          (realPart (a * conjugate a), l)) $
    map toPair $
    quantumStates q
```

Для кубита берётся список его квантовых состояний, которые преобразуются в пары. Далее амплитуды во всех этих парах заменяются на вероятности (сразу же берётся только действительная часть комплексных чисел), а элементы пар меняются местами. На всякий случай новые пары сортируются по второму элементу (вероятности), после чего результирующий отсортированный список скормливается только что определённой функцией `getRandomElementWithProbabilities`. Она работает безупречно.

Для того чтобы облегчить себе и другим разработчикам, которые будут пользоваться созданным фреймворком, можно определить несколько константных функций для представления наиболее часто встречающихся кубитов. К таковым, без сомнений, можно отнести кубиты $|0\rangle$, $|1\rangle$, $|+\rangle$, $|-\rangle$, а также все кубиты, представляющие состояния Белла. Их определения будут довольно простыми:

```
qubitZero :: Qubit Double
qubitZero = Qubit [QS (1.0 :+ 0.0) "0",
                   QS (0.0 :+ 0.0) "1"]

qubitOne :: Qubit Double
qubitOne = Qubit [QS (0.0 :+ 0.0) "0",
                  QS (1.0 :+ 0.0) "1"]

qubitPlus :: Qubit Double
qubitPlus = Qubit [QS ((1/sqrt 2) :+ 1.0) "0",
                   QS ((1/sqrt 2) :+ 0.0) "1"]

qubitMinus :: Qubit Double
qubitMinus = Qubit [QS (1/sqrt 2 :+ 0.0) "0",
                    QS ((-1)/sqrt 2 :+ 0.0) "1"]

qubitPhiPlus :: Qubit Double
qubitPhiPlus = Qubit [QS (1/sqrt 2 :+ 0.0) "00",
                      QS (1/sqrt 2 :+ 0.0) "11"]

qubitPhiMinus :: Qubit Double
qubitPhiMinus = Qubit [QS (1/sqrt 2 :+ 0.0) "00",
                       QS ((-1)/sqrt 2 :+ 0.0) "11"]

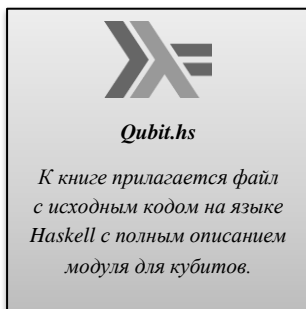
qubitPsiPlus :: Qubit Double
qubitPsiPlus = Qubit [QS (1/sqrt 2 :+ 0.0) "01",
                      QS (1/sqrt 2 :+ 0.0) "10"]

qubitPsiMinus :: Qubit Double
qubitPsiMinus = Qubit [QS (1/sqrt 2 :+ 0.0) "01",
                       QS ((-1)/sqrt 2 :+ 0.0) "10"]
```

Для тех же самых целей можно определить и эти же кубиты, но в векторном представлении. Это делается тривиально при помощи использования функции `toVector`, поэтому определения здесь приводиться не будут, только лишь общий вид таких функций:

```
{-  
qubit*' :: [Complex Double]  
qubit*' = toVector qubit*  
-}
```

Здесь символ (*) в имени функции обозначает название кубита и может принимать 8 значений: `Zero`, `One` и т. д.



Гейты

Как было показано ранее, гейт — это просто квадратная матрица с размерами $2^n \times 2^n$ для преобразования квантовой системы из n кубитов. Так что прежде всего надо определить типы для представления матриц и векторов. В принципе, для этих целей есть специальная библиотека (и даже, скорее всего, уже не одна), в которой вся матричная алгебра реализована достаточно эффективно, однако здесь мы сделаем небольшой кусочек подобной библиотеки, заботясь не об эффективности, а о простоте представления для более полного понимания модели квантовых вычислений.

Так что матрицы и векторы представим в виде списков. Это неэффективно и чревато логическими ошибками, поскольку разработчику придётся самостоятельно на уровне исходного кода следить за корректностью размеров векторов и матриц. Прикладные функции для работы с этими типами, которые будут написаны позже, не смогут эффективно проверять совпадение размерностей для валидного применения математических операций.

Для определения типов в целях представления векторов и матриц достаточно использовать синонимы:

```
type Vector a = [a]
```

```
type Matrix a = [Vector a]
```

В принципе, такое представление оправдано ещё и тем, что в стандартной библиотеке языка Haskell имеется огромное множество уже готовых функций для работы со списками. В частности, вот так просто и непринуждённо определяется функция для применения матрицы (то есть унитарного преобразования) к вектору (то есть к квантовой системе):

```
apply :: Num a => Matrix a -> Vector a -> Vector a  
apply m v = map (sum . zipWith (*) v) m
```

Как видно, это просто умножение матрицы на вектор. Но зато как элегантно!

Не менее элегантно выглядит функция для получения эрмитово-сопряжённой матрицы для заданной. Здесь мы вполне можем воспользоваться стандартными функциями и записать определение следующим образом:

```
adjoint :: RealFloat a  
        => Matrix (Complex a) -> Matrix (Complex a)  
adjoint = map (map conjugate) . transpose
```

Для векторной алгебры, применяемой в рамках модели квантовых вычислений, важны операции внутреннего

и внешнего произведений векторов (это то, что в нотации Дирака обозначается как $\langle \varphi | \psi \rangle$ и $|\varphi\rangle\langle\psi|$). Определим эти функции в виде операторов:

```
(<|>) :: Num a => Vector a -> Vector a -> a
v1 <|> v2 = sum $ zipWith (*) v1 v2

(|><|) :: RealFloat a => Vector (Complex a)
      -> Vector (Complex a) -> Matrix (Complex a)
v1 |><| v2 = [map (* c) v2 | c <- map conjugate v1]
```

Конечно, нам в обязательном порядке потребуются функции для осуществления многочисленных операций перемножения. Одна из них уже написана (`apply`), а теперь напомним определения операторов для умножения числа на матрицу, для обычного и тензорного перемножения матриц.

Умножить число на матрицу очень просто:

```
(<*>) :: Num a => a -> Matrix a -> Matrix a
c <*> m = map (map (c *)) m
```

Двойное использование функции `map` необходимо для «втягивания» числа с операцией умножения для использования со списками второго уровня вложенности. Оператор перемножения матриц выглядит не менее просто:

```
(<*>) :: Num a => Matrix a -> Matrix a -> Matrix a
m1 <*> m2 = [apply m1 col | col <- transpose m2]
```

Само собой разумеется, что разработчик всегда должен самостоятельно следить за корректностью размеров матриц и векторов, поскольку в противном случае реализованные функции не будут рапортовать ни о каких ошибках, а просто будут выдавать некорректные результаты.

Тензорное произведение матриц не так просто, и чтобы определить соответствующий оператор, необходимо будет подготовить несколько вспомогательных функций. Вот всё опреде-

ление, после внимательного изучения которого можно рассмотреть все вспомогательные функции:

```
m1 <+> m2 = concat $
    map collateRows $
    groups n [c <*> m2 | c <- concat m1]

where
    n = length $ head m1

    groups :: Int -> [a] -> [[a]]
    groups i s | null s = []
                | otherwise = let (h, t) = splitAt i s
                                in  h : groups i t

    collateRows :: [Matrix a] -> Matrix a
    collateRows = map concat . transpose
```

Образец `n` содержит лишь ширину первой матрицы. Здесь опять используется негласное соглашение о том, что разработчик должен сам следить за корректностью представления своих матриц.

Локальная функция `groups` используется для разбиения заданного списка на подсписки определённой длины. Она понадобится для разбиения списка матриц на матрицу матриц.

Очень важная функция `collateRows`. Она объединяет матрицы в списке специальным образом: берёт у каждой матрицы первые строки и конкатенирует их друг с другом, и это получается первая строка результирующей матрицы. То же самое со вторыми строками, с третьими и т. д. до конца. Эта функция получает на вход список матриц, то есть список списков списков, и потому достаточно сложна для восприятия. После полного описания функции будет приведён пример работы её самой и всех её частей, на котором будет чётко видно, что происходит.

Сам оператор тензорного произведения поступает с переданными ему на вход матрицами очень просто. Первая

матрица (список списков) полностью конкатенируется, в результате чего получается список комплексных чисел. Все эти комплексные числа умножаются на вторую матрицу, в результате чего получается список матриц.

Этот список матриц подаётся на вход функции `groups`, которая разбивает его на подсписки длиной, равной ширине первой матрицы `n`. Это значит, что в результате работы этой функции получается матрица матриц, то есть список списков списков списков. Для того чтобы из этой красоты снова получить матрицу, необходимо снять два уровня вложенности, но это надо сделать с умом.

Первый уровень вложенности снимается при помощи применения локальной функции `collateRows`, которая преобразует список матриц в матрицу. Ну и вторым шагом применяется просто стандартная функция `concat`, которая снимает самый внешний уровень вложенности.

Всё это очень непросто понять, поскольку когда есть списки четвёртого уровня вложенности, крутить-вертеть их в голове очень непросто. Поэтому небольшие поясняющие экземпляры исходного кода. На них используются «обычные» числа, а не комплексные, чтобы не загромождать иллюстрации.

Пусть мы хотим получить тензорное произведение двух гейтов Адамара. В командной строке интерпретатора определим такой гейт как:

```
> let h = (1/sqrt 2) <*> [[1, 1], [1, (-1)]]
```

Теперь последовательно разберём по шагам вычисление выражения `h <+> h`. На первом шаге первая матрица `h` конкатенируется, в результате чего получается (здесь опять же для незагромождения результат `1/sqrt 2` представлен как `0.7`):

```
> concat h
[0.7, 0.7, 0.7, -0.7]
```

Далее каждый элемент этого списка умножается на вторую матрицу h , в результате чего получается список матриц (здесь список третьего уровня показан на нескольких строках, чтобы было понятнее):

```
> [c <*> h | c <- concat h]
[[[ 0.5,  0.5], [ 0.5, -0.5]],
 [ 0.5,  0.5], [ 0.5, -0.5]],
 [ 0.5,  0.5], [ 0.5, -0.5]],
 [-0.5, -0.5], [-0.5,  0.5]]]
```

Теперь это добро необходимо опять группировать. Поскольку ширина матрицы h равна 2, то производится разбиение этого списка на группы по 2 элемента:

```
> groups 2 [c <*> h | c <- concat h]
[[[[ 0.5,  0.5], [ 0.5, -0.5]],
 [ 0.5,  0.5], [ 0.5, -0.5]]],
 [[ 0.5,  0.5], [ 0.5, -0.5]],
 [[-0.5, -0.5], [-0.5,  0.5]]]]
```

Итак, это матрица матриц. Тем читателям, кто этого не видит, рекомендуется медитировать над результатом выполнения этого выражения, пока его суть не перейдёт непосредственно в мозг. Если список списков представлять в виде квадратной матрицы, то этот список четвёртого уровня вложенности выглядит так:

$$\left(\begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} \end{pmatrix} \quad \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} \end{pmatrix} \right) \\ \left(\begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} \end{pmatrix} \quad \begin{pmatrix} -\frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{pmatrix} \right)$$

Понятно, что теперь надо «снять скобки» с внутренних матриц, то есть, в терминах языка Haskell, снять два уровня списков. Как было показано выше, первый уровень списков снима-

ется при помощи применения ко всем элементам верхнего списка (то есть спискам матриц) функции `collateRows`:

```
> collateRows $ head $ groups 2 [c <*> h | c <- concat h]
[[0.5, 0.5, 0.5, 0.5],
 [0.5, -0.5, 0.5, -0.5]]
```

Поскольку функция `collateRows` применяется через функцию `map`, то она применяется ко всем элементам полученного списка четвёртого уровня вложенности. В итоге получается как бы вектор матриц, каждая из которых в два раза шире, чем по высоте. В математической нотации это выглядит следующим образом:

$$\left(\begin{pmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \end{pmatrix} \right)$$

Наконец, при помощи применения функции `concat`, снимается ещё один уровень вложенности, и в итоге получается та матрица, которая требуется:

```
> h <+> h
[[0.5, 0.5, 0.5, 0.5],
 [0.5, -0.5, 0.5, -0.5],
 [0.5, 0.5, -0.5, -0.5],
 [0.5, -0.5, -0.5, 0.5]]
```

Так же как и для кубитов имеет смысл определить специальные константные функции, которые будут возвращать наиболее часто используемые гейты. Для простоты можно определить квантовые гейты I , X , Y , Z , H и $CNOT$. Вот они:

```
gateI :: Matrix (Complex Double)
gateI = [[1.0 :+ 0.0, 0.0 :+ 0.0],
         [0.0 :+ 0.0, 1.0 :+ 0.0]]
```



```
gateX :: Matrix (Complex Double)
gateX = [[0.0 :+ 0.0, 1.0 :+ 0.0],
         [1.0 :+ 0.0, 0.0 :+ 0.0]]

gateY :: Matrix (Complex Double)
gateY = [[0.0 :+ 0.0, 0.0 :+ (-1.0)],
         [0.0 :+ 1.0, 0.0 :+ 0.0]]

gateZ :: Matrix (Complex Double)
gateZ = [[1.0 :+ 0.0, 0.0 :+ 0.0],
         [0.0 :+ 0.0, (-1.0) :+ 0.0]]

gateH :: Matrix (Complex Double)
gateH = [[(1/sqrt 2) :+ 0.0, (1/sqrt 2) :+ 0.0],
         [(1/sqrt 2) :+ 0.0, ((-1)/sqrt 2) :+ 0.0]]

gateCNOT :: Matrix (Complex Double)
gateCNOT
  = [[1.0 :+ 0.0, 0.0 :+ 0.0, 0.0 :+ 0.0, 0.0 :+ 0.0],
     [0.0 :+ 0.0, 1.0 :+ 0.0, 0.0 :+ 0.0, 0.0 :+ 0.0],
     [0.0 :+ 0.0, 0.0 :+ 0.0, 0.0 :+ 0.0, 1.0 :+ 0.0],
     [0.0 :+ 0.0, 0.0 :+ 0.0, 1.0 :+ 0.0, 0.0 :+ 0.0]]
```



Gate.hs

*К книге прилагается файл
с исходным кодом на языке
Haskell с полным описанием
модуля для гейтов.*

Квантовые вычислительные схемы

Квантовая вычислительная схема — это последовательность применения унитарных преобразований к входным кубитам. В принципе, в ранее рассмотренных модулях уже определены все необходимые для этого функции; вернее даже — одна

функция, а именно функция `apply` в модуле `Gate`. Она применяет заданный первым аргументом гейт к заданному вторым аргументом кубиту.

На диаграммах, представляющие квантовые схемы, поток вычисления производится слева направо. Было бы здорово определить ещё одну функцию для следования традиции. Это просто:

```
ylppa :: Num a => Vector a -> Matrix a -> Vector a
ylppa = flip apply
```

По традиции же такая функция названа странным словом «`ylppa`» — это перевёрнутое задом наперёд слово «`apply`». Однако такое решение половинчато, поскольку очень сложно будет представлять себе квантовые схемы примерно следующим образом:

```
> toVector qubitZero `ylppa` gateX
```

Было бы намного удобнее определить какой-нибудь специальный оператор для этих целей. В этом нет ничего сложного:

```
(|>) :: Num a => Vector a -> Matrix a -> Vector a
(|>) = ylppa
```

И теперь квантовые схемы можно записывать очень даже приятно для взгляда:

```
> toVector qubitZero |> gateH |> gateZ |> gateH
```

И, наконец, для удобства можно определить ещё один оператор. Это тот же самый оператор `($)`, но только опять изменён порядок следования аргументов:

```
(>>>) :: a -> (a -> b) -> b
(>>>) = flip ($)
```

При помощи этого оператора можно производить такие финты:

```
> toVector qubitOne |> gateX >>> (measure . fromVector 1)
```

Для операторов ($|>$) и ($>>>$) можно было бы определить приоритетность и ассоциативность их выполнения, однако по умолчанию они будут иметь левостороннюю ассоциативность и наивысший приоритет (9), чего вполне достаточно для целей фреймворка.

На самом деле, конечно же, наиболее приемлемым способом организации квантовых вычислений является разработка специальной монады для этих целей. Поскольку монада абстрагирует собой некоторую вычислительную стратегию, а модель квантовых вычислений представляет собой именно что вычислительную стратегию, специальная монада стала бы самым естественным способом представления квантовых схем.

Данный вопрос, однако, хотя и интересен, но потребовал бы слишком много усилий для такой небольшой книги. К тому же, реализация собственной монады выходит далеко за рамки базового владения функциональной парадигмой программирования. Тем не менее, в следующей главе при описании языка программирования Quirreg данный вопрос будет отчасти раскрыт.

И на этом всё.



Circuit.hs

*К книге прилагается файл
с исходным кодом на языке
Haskell с полным описанием
модуля для квантовых схем.*

Как видно, прекрасный код. Он читается просто — взять два входных кубита $q1$ и $q2$, сцепить их друг с другом, после чего послать на вход гейту CNOT, потом гейту CNOT' (контролирующий кубит второй, а не первый), потом опять на вход гейту CNOT, после чего произвести измерение. Можно проверить эту прекрасную функцию, составив таблицу истинности для этой квантовой схемы:

```
truthTableSQ
= sequence_ [swapQubits q1 q2 >=> putStrLn |
             q1 <- basis,
             q2 <- basis]

where
  basis = [qubitZero, qubitOne]
```

Если запустить эту функцию на исполнение в командной строке интерпретатора, то на экран будет выведено:

```
> truthTableSQ
00
10
01
11
```

что абсолютно резонно, поскольку базисными состояниями для двухкубитной системы является множество $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$. Так что из результатов вывода видно, что во второй и третьей строке произошло изменение порядка кубитов.

Ещё можно попробовать решить такую простенькую задачу. Гейт X , например, можно реализовать через гейты Адамара H и гейт смены фазы Z . Как это сделать? Если посмотреть на выражения для кубитов $|0\rangle$ и $|1\rangle$ с одной стороны и $|+\rangle$ и $|-\rangle$ с другой стороны, то будет видно, что первая пара кубитов различается комплексными коэффициентами, а вторая — знаком при втором квантовом состоянии.

Гейт Адамара переводит кубит из базиса $\{|0\rangle, |1\rangle\}$ в базис $\{|+\rangle, |-\rangle\}$ и обратно. А гейт Z меняет знак при втором квантовом состоянии. Следующая таблица показывает преобразование HZH, применённое к кубитам $|0\rangle$ и $|1\rangle$.

Таблица 8. Различные двоичные функции одной переменной

Кубит	H	Z	H
$ 0\rangle$	$ +\rangle$	$ -\rangle$	$ 1\rangle$
$ 1\rangle$	$ -\rangle$	$ +\rangle$	$ 0\rangle$

Действительно, если осуществить перемножение соответствующих унитарных матриц, то получим:

$$\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Осталось реализовать код, который осуществляет проверку этого результата:

```
notHZN :: Qubit Double -> IO ()
notHZN q = (toVector q) |> gateH
           |> gateZ
           |> gateH
           >>> (measure . fromVector 1)
           >>= putStrLn
```

И проверка:

```
> mapM_ notHZN [qubitZero, qubitOne]
1
0
```

Третьей задачей, которую можно попробовать реализовать, является следующая. Пусть у нас есть двоичная функция

от одной переменной. Таких различных функций всего существует четыре:

Таблица 9. Различные двоичные функции одной переменной

№	$f\ 0$	$f\ 1$	Определение функции
1	0	0	$f\ x = 0$
2	1	1	$f\ x = 1$
3	0	1	$f\ x = x$
4	1	0	$f\ x = \text{not } x$

Функции 1 и 2 являются константными, а функции 3 и 4 — «сбалансированными». Второе название происходит от того, что количество нулей и единиц в множестве значений функций одинаково.

Как понять, является ли заданная функция константной или сбалансированной? В традиционной вычислительной модели необходимо осуществить с этой функцией 2 вычислительных шага — вычислить значения функции на 0 и на 1, после чего сравнить. Если они не отличаются друг от друга, то функция константна. Если отличаются, то, стало быть, сбалансирована. Вот так выглядит описание этого алгоритма в классическом виде (два вызова функции f выделены красным цветом):

```
deutsch' f = if f 0 == f 1
               then putStrLn "Функция f константна."
               else putStrLn "Функция f сбалансирована."
```

А вот при помощи квантовых вычислений необходимо сделать только один вызов функции f , чтобы понять, к какому типу она относится. Это получается из-за квантового параллелизма и одновременного вычисления значений заданной функции на всех возможных значениях аргумента. И затем при помощи

хитроумного способа можно понять, к какому же типу относится функция f .

Квантовая схема, реализующая этот алгоритм, показана на следующем рисунке:

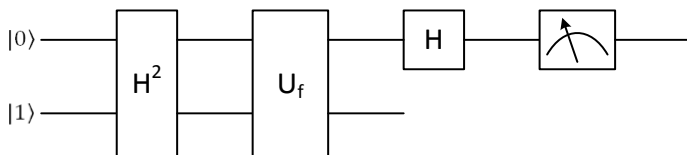


Рис. 16. Квантовая схема, реализующая алгоритм определения типа заданной функции

Само собой разумеется, что для квантового алгоритма функция f должна быть преобразована в унитарное преобразование U_f , и это получится оракул в рамках квантовой схемы. А вот так выглядит реализация данной схемы при помощи разработанного фреймворка:

```
deutsch :: Matrix (Complex Double) -> IO ()
deutsch f =
  do (result:_) <- circuit
    case result of
      '0' -> putStrLn "Функция f константна."
      '1' -> putStrLn "Функция f сбалансирована."
      _   -> return ()
  where
    gateH2 = gateH <+> gateH
    circuit = toVector (entangle qubitZero qubitOne)
              |> gateH2
              |> f
              |> gateH2
    >>> (measure . fromVector 2)
```

Теперь необходимо реализовать оракулы для четырёх функций, которые перечислены в таблице 7. Если вспомнить, то оракулом называется специального вида гейт, который осуществляет вычисление заданной функции. Поскольку у нас

здесь на схеме входных кубитов два, то эти гейты оракулов будут представлять собой матрицы размера 4×4 . Чтобы их создать, необходимо немного заняться квантовой схемотехникой.

Нам будет необходимо спроектировать четыре гейта, которые осуществляют следующее унитарное преобразование:

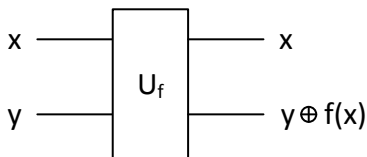


Рис. 17. Гейт для представления оракула функции f в рассматриваемом примере

Рассмотрим первую функцию, которая является константной и на любом своём аргументе возвращает 0. Для построения гейта необходимо воспользоваться таблицей следующего вида:

Таблица 10. Вспомогательная таблица для проектирования квантового оракула для первой функции

x	y	$f x$	$y \oplus f x$	Преобразование
0	0	0	0	$ 00\rangle \rightarrow 00\rangle$
0	1	0	1	$ 01\rangle \rightarrow 01\rangle$
1	0	0	0	$ 10\rangle \rightarrow 10\rangle$
1	1	0	1	$ 11\rangle \rightarrow 11\rangle$

Для того чтобы прочитать столбец «Преобразование», необходимо помнить, что в кубите слева от стрелки на первой позиции стоит значение x , а на второй — y . В кубите справа от стрелки на первой позиции опять стоит значение x , а на второй — $y \oplus f x$. Понятно, что в данном случае унитарное преобразование задаётся единичной матрицей размера 4×4 .

Абсолютно так же можно построить матрицу гейта для второй функции, которая является константной и всегда возвращает значение 1. Снова составим вспомогательную таблицу, которая поможет затем нарисовать унитарную матрицу, представляющую гейт. Вот она:

Таблица 11. Вспомогательная таблица для проектирования квантового оракула для второй функции

x	y	fx	$y \oplus fx$	Преобразование
0	0	1	1	$ 00\rangle \rightarrow 01\rangle$
0	1	1	0	$ 01\rangle \rightarrow 00\rangle$
1	0	1	1	$ 10\rangle \rightarrow 11\rangle$
1	1	1	0	$ 11\rangle \rightarrow 10\rangle$

Другими словами, данное унитарное преобразование должно всегда применять операцию отрицания ко второму кубиту, независимо от значения первого кубита. Матрица такого преобразования выглядит следующим образом:

$$U_{f_2} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Вдумчивый читатель уже наверняка понял, как выглядят матрицы для третьей и четвёртой функции. В любом случае, их построение будет хорошей тренировкой понимания всего пройденного к настоящему моменту материала. Ну а здесь мы просто покажем исходный код для представления всех оракулов:

```
unitaryF1 :: Matrix (Complex Double)
unitaryF1 =
    [[1.0 :+ 0.0, 0.0 :+ 0.0, 0.0 :+ 0.0, 0.0 :+ 0.0],
     [0.0 :+ 0.0, 1.0 :+ 0.0, 0.0 :+ 0.0, 0.0 :+ 0.0],
```

```

[0.0 :+ 0.0, 0.0 :+ 0.0, 1.0 :+ 0.0, 0.0 :+ 0.0],
[0.0 :+ 0.0, 0.0 :+ 0.0, 0.0 :+ 0.0, 1.0 :+ 0.0]]

unitaryF2 :: Matrix (Complex Double)
unitaryF2 =
  [[0.0 :+ 0.0, 1.0 :+ 0.0, 0.0 :+ 0.0, 0.0 :+ 0.0],
   [1.0 :+ 0.0, 0.0 :+ 0.0, 0.0 :+ 0.0, 0.0 :+ 0.0],
   [0.0 :+ 0.0, 0.0 :+ 0.0, 0.0 :+ 0.0, 1.0 :+ 0.0],
   [0.0 :+ 0.0, 0.0 :+ 0.0, 1.0 :+ 0.0, 0.0 :+ 0.0]]

unitaryF3 :: Matrix (Complex Double)
unitaryF3 = gateCNOT

unitaryF4 :: Matrix (Complex Double)
unitaryF4 =
  [[0.0 :+ 0.0, 1.0 :+ 0.0, 0.0 :+ 0.0, 0.0 :+ 0.0],
   [1.0 :+ 0.0, 0.0 :+ 0.0, 0.0 :+ 0.0, 0.0 :+ 0.0],
   [0.0 :+ 0.0, 0.0 :+ 0.0, 1.0 :+ 0.0, 0.0 :+ 0.0],
   [0.0 :+ 0.0, 0.0 :+ 0.0, 0.0 :+ 0.0, 1.0 :+ 0.0]]

```

(Здесь необходимо заметить в скобках, что ленивый разработчик давно бы уже написал сервисную функцию для преобразования матриц с целыми/действительными числами в матрицы с комплексными числами, у которых мнимая часть нулевая).

Теперь можно написать функцию для запуска разработанной ранее функции `deutsch` для всех четырёх разработанных оракулов. Вот её определение:

```

testDeutsch :: IO ()
testDeutsch = mapM_ deutsch [unitaryF1,
                             unitaryF2,
                             unitaryF3,
                             unitaryF4]

```

Если осуществить вызов этой функции из командной строки, то мы получим вполне ожидаемый результат:

```

> testDeutsch
Функция f константна.
Функция f константна.

```

Функция f сбалансирована.

Функция f сбалансирована.

Вот это красота! Мы действительно осуществили вызов функции f один раз во время выполнения квантового алгоритма (единственный вызов указан красным цветом в исходном коде функции `deutsch`). Конечно, нам пришлось преобразовать функцию в гейт для представления оракула, специального вида унитарного преобразования. Но, к счастью, эту операцию очень легко сделать автоматической — можно реализовать функцию, которая для заданной классической функции строит соответствующий квантовый оракул.

Задача, которую мы только что решили, представляет собой самую простую версию алгоритма Дойча, одного из простейших алгоритмов модели квантовых вычислений, который показывает двукратное увеличение эффективности для определения типа функции. Во всех подробностях этот алгоритм описывается в главе 4.



Краткие выводы

Для реализации модели квантовых вычислений вполне можно обойтись «обычным компьютером», однако, конечно же, программы на нём будут исполняться несоизмеримо долго, занимая огромное количество памяти. Всё, что необхо-

димо реализовать, — это векторная и матричная алгебра, причём элементами векторов и матриц являются комплексные числа.

Наиболее приемлемой для реализации модели квантовых вычислений видится парадигма функционального программирования, поскольку все ключевые понятия этой новой вычислительной модели естественным образом ложатся на идиомы функционального программирования.

Реализация фреймворка на языке Haskell в качестве примера позволяет понять, что сама модель квантовых вычислений не так сложна, как может показаться на первый взгляд. Однако, конечно же, решать реальные проблемы при помощи таких фреймворков не получится, поскольку в дело вступят именно ограничения по эффективности и количеству требуемой памяти — обычный компьютер будет требовать гигантского количества вычислительных ресурсов для манипуляции необходимым количеством кубитов и применения к ним унитарных преобразований. Таким образом, необходимо именно квантовое вычислительное устройство, которое на фундаментальном (аналоговом) уровне будет выполнять все те квантовые операции, которые ведут к повышению эффективности по сравнению с традиционной вычислительной моделью.

Созданный и описанный в этой главе фреймворк далёк, конечно же, от совершенства. В нём не реализовано много того, что должно быть сделано для полноценного описания модели квантовых вычислений на каком-либо языке программирования. Например, из самого банального, — здесь не реализована возможность применения гейта к части кубитов в квантовой схеме (это можно видеть на примере решения второй задачи, когда на квантовой схеме после применения оракула гейт Адамара применяется только к первому кубиту, равно как и измерение, а в реализации этой квантовой схемы и гейт Ада-

мара, и измерение применяются к обоим кубитам). Не реализована работа со вспомогательными битами, не созданы операции для инициализации кубитов и их уничтожения. Однако главная цель достигнута — показана принципиальная возможность создания языка программирования для квантовых вычислений.

Глава 3.

Язык программирования Quipper

«Понимать» — это, по-видимому, означает овладеть представлениями, концепциями, с помощью которых мы можем рассматривать огромное множество различных явлений в их целостной связи, иными словами, «охватить» их.

Вернер Гейзенберг

Развитие понимания модели квантовых вычислений и получение экспериментальных результатов по обработке информации при помощи кубитов не могло не сказаться на том, что данный вопрос начал прорабатываться и с другой стороны.

Несколько коллективов исследователей задались целью спроектировать и реализовать язык программирования, который был бы удобен для описания квантовых алгоритмов. Исторически первый таким языком стал язык QCL, или Quantum Computation Language (*язык квантовых вычислений*). Далее были попытки сделать ещё некоторые языки программирования для квантовых вычислений. А к сегодняшнему дню глубокое развитие при поддержке государственных институтов Канады и США получил язык программирования Quipper, который основан на языке Haskell, дополняя его всем необходимым для решения специфических задач модели квантовых вычислений.

Однако не стоит думать, что к настоящему моменту разработано только два языка квантового программирования. Как это ни странно, всё их множество так же можно разделить на языки с императивной основой и на языки функциональные. Так к императивным языкам квантового программирования относятся: квантовый псевдокод (это вообще был первый формализованный язык для квантового программирования, который, однако, остался только на бумаге), уже упомянутый QCL, затем язык Q, и, наконец, язык qGCL (*Quantum Guarded Command Language*).

С другой стороны, к функциональным языкам квантового программирования относятся следующие: QFC и QPL (оба разработаны П. Селинджером, различаются только синтаксисом — первый является языком квантовых схем), QML (тоже, как ни странно, основан на языке Haskell), квантовое лямбда-исчисление и, собственно, язык Quipper.

Следующая таблица представляет сводную информацию о существующих на сегодняшний день языках квантового программирования.

Таблица 12. Языки квантового программирования

Язык	Разработчик	Пара- дигма	Основан на языке	Примечание
Квантовый псевдокод	Knill E.	И	Псевдокод	Введена модель квантового вы- числительного устройства QRAM.
QCL	Ömer B.	И	C	Первый язык, имеющий реали- зацию.
Q	Mlnářik H.	И	C++	Второй язык, имеющий реали- зацию.
qGCL	Zuliani P.	И	GCL	
QFC	Selinger P.	Ф	—	Графический язык квантовых схем.
QPL	Selinger P.	Ф	—	Текстовый язык квантовых схем.
QML	Altenkirch T. Grattage J.	Ф	Haskell	
Квантовое лямбда- исчисление	Maymin P. Tonder A. Selinger P. Valiron B.	Ф	Лямбда- исчисле- ние	Есть попытка реа- лизации на языке Scheme.
Quipper	Green A. Lumsdaine P. Selinger P. Valiron B. Ross N.	Ф	Haskell	Язык имеет реа- лизацию компи- лятора и многих утилит для работы.

Конечно, на текущем этапе научно-технического прогресса необходимо понимать, что все перечисленные в таблице формализмы и языки программирования не имеют реализации для какого-либо квантового вычислительного устройства. Для языков, у которых есть реализация, просто созданы компилятор и специальный симулятор квантового компьютера. Компилятор компилирует исходный код в нативный

для симулятора, на котором программа может исполняться. Но симулятор, конечно же, сам исполняется на классическом компьютере, поэтому никаких выгод такой режим работы не даст. А вот когда квантовый компьютер будет разработан и создан, тогда уж и компиляторы будут переделаны для перевода исходных кодов в управляющие команды квантового вычислительного устройства. И тогда все написанные к настоящему моменту программы на языках квантового программирования смогут быть успешно выполнены на созданном устройстве.

В этой главе мы кратко рассмотрим описание языка QCL исключительно ради понимания того, что это такое (и как вообще можно описывать квантовые алгоритмы в рамках императивной парадигмы). После этого будет приведено описание языка Quipper, его особенностей по сравнению с базовым языком Haskell, а также будет приведено решение некоторых задач на нём.

Немного о языке QCL

Язык программирования QCL был разработан Бернардом Омером и описан в его диссертации в январе 2009 года. Этот язык программирования основан на структурном подходе и использует С-подобный синтаксис для выражения концепций модели квантовых вычислений. На текущий момент язык представляет собой достаточно развитую систему, автор предлагает для скачивания версию 0.6.3.

Автор языка отмечает, что одним из мотивов создания им языка QCL было то, что существует некоторая «пропасть» между физическим описанием модели квантовых вычислений и информатикой (computer science) — нотация Дирака, унитарные комплекснозначные матрицы, операторы и гейты и т. д. — всё это в совокупности делает модель квантовых вычислений

малопонятной для широкого использования специалистами по информатике. Проложить мост через эту пропасть должен был по замыслу автора язык QCL, поскольку он предоставил бы программистам высокоуровневые средства обработки информации, в том числе и при помощи квантового вычислительного устройства.

В языке QCL элементарным типом данных является *квантовый регистр*, то есть набор кубитов. Этот тип в языке называется `qureg`. Значение этого типа адресуется к физическим кубитам в так называемой квантовой куче, то есть внешней квантовой памяти, управляемой квантовым вычислительным устройством. На квантовых регистрах можно производить три элементарные операции — инициализацию кубитов, унитарные преобразования и измерения.

Вместе с тем, также в языке QCL имеется возможность определять то, что в классических языках называется «процедурами» или «функциями». Однако это не совсем то же самое, поскольку *комплексные квантовые операторы* языка QCL представляют собой связную последовательность базовых гейтов и других комплексных квантовых операторов. Каждый такой оператор принимает на вход один или несколько квантовых регистров, и возвращает, соответственно, такое же их количество (вспоминаем про обратимость вычислений). К комплексному квантовому оператору можно применять такую операцию высшего порядка, как *обращение* — все составляющие этого оператора преобразуются в эрмитово-сопряжённые, их последовательность меняется с последнего на первый.

Кроме типа данных для представления квантового регистра, автором языка было реализовано ещё некоторое количество более специализированных типов данных. К таковым относятся:

- Константный регистр `quconst`, который предполагает, что унитарный оператор не имеет права модифицировать значение кубита. Например, гейт CNOT в качестве первого (управляющего) кубита может принимать кубит с этим типом.
- Пустой регистр `quvoid`, который гарантированно не проинициализирован при передаче в унитарный оператор. Например, при реализации квантового оракула $f: |x\rangle|y\rangle \rightarrow |x\rangle|y \oplus f(x)\rangle$ второй регистр y обычно предполагается пустым.
- Вспомогательный регистр `quscratch`, который предполагает, что он пуст в начале, но при этом после работы унитарного оператора он также должен остаться пустым.

Несмотря на то, что все эти типы квантовых регистров на физическом уровне должны быть реализованы посредством одинакового механизма, наличие специализированных типов позволяет автоматически производить валидацию описанных ограничений.

Операторный подход, который используется в языке QCL, позволяет писать очень однообразный код — и классические, и квантовые процедуры выглядят одинаково.

С квантовыми регистрами можно производить несколько операций, которые позволяют создавать и индексировать регистры и отдельные кубиты в них. К этим операциям относятся следующие (предполагается, что переменная a — это квантовый регистр):

- a — ссылка на регистр. Результатом будет перечисление всех входящих в регистр кубитов: $\langle a_0, a_1, \dots, a_n \rangle$.
- $a[i]$ — ссылка на кубит. Результатом будет конкретный кубит, находящийся на i -ой позиции в регистре: $\langle a_i \rangle$.

- $a[i..j]$ — ссылка на подрегистр. Результатом будет часть квантового регистра: $\langle a_i, a_{i+1}, \dots, a_j \rangle$.
- $a[i::j]$ — ссылка на подрегистр иного рода. Результатом опять будет часть квантового регистра, но иная: $\langle a_i, a_{i+1}, \dots, a_{i+j-1} \rangle$.
- $a \ \& \ b$ — конкатенация двух квантовых регистров. Результатом будет новый регистр: $\langle a_0, a_1, \dots, a_n, b_0, b_1, \dots, b_m \rangle$.

Само собой разумеется, что в языке QCL есть как уже ранее реализованные самые разнообразные гейты, так и возможность по их созданию разработчиком программного обеспечения. Для этих целей можно создавать процедуры, операторы и функции двух типов — у всех этих программных сущностей есть свои свойства, которые позволяют гибко определять вычислительные блоки в рамках как классических, так и квантовых вычислений. Однако формат данной книги слишком узок для полного перечисления всех возможностей языка QCL.

Остаётся привести небольшой кусок исходного кода на рассматриваемом языке, чтобы сделать о нём свои выводы. Вот он:

```
/* Define Oracle */

const coin1=(random())>=0.5); // Define two random boolean
const coin2=(random())>=0.5); // constants

boolean g(boolean x) { // Oracle function g
  if coin1 {           // coin1=true -> g is constant
    return coin2;
  } else {             // coin1=false -> g is balanced
    return x xor coin2;
  }
}

qufunct G(quconst x, quvoid y) { // Construct oracle
                                // op. G from g
```

```

    if g(false) xor g(true) { CNot(y, x); }
    if g(false) { Not(y); }
}

/* Deutsch's Algorithm */

operator U(qureg x, qureg y) { // Bundle all
                                // unitary operations
    H(x);                       // of the algorithm
                                // into one
    G(x,y);                     // operator U
    H(x & y);
}

procedure deutsch() {          // Classical control structure
    qureg x[1];                 // allocate 2 qubits
    qureg y[1];
    int m;
    {                           // evaluation loop
        reset;                  // initialize machine state
        U(x, y);                // do unitary computation
        measure y,m;            // measure 2nd register
    } until m==1;               // value in 1st register valid?
    measure x,m;                // measure 1st register which
                                // contains g(0) xor g(1)
    print "g(0) xor g(1) =", m;
    reset;                      // clean up
}

```

Ну вот как-то так. Это реализация всё того же алгоритма Дойча, который был уже реализован при помощи разработанного в предыдущей главе фреймворка для квантовых вычислений. Как видно, здесь сплошь идёт императивный код, но и сам автор определяет свой язык как язык структурного квантового программирования. Насколько это удобно — решать каждому разработчику.

Математическая модель, синтаксис и денотационная семантика языка QCL полностью описаны в двух магистерских и докторской диссертациях его автора, которые можно найти

по следующему адресу в сети Интернет:
<http://www.itp.tuwien.ac.at/~oemer/qcl.html>.

Введение в язык Quipper

Язык программирования Quipper — это самая современная разработка, основанная на всех последних достижениях и идеях относительно модели квантовых вычислений. Этот язык является проблемно-ориентированным языком, базовым для которого является язык Haskell. Это значит, что синтаксисом языка Quipper является синтаксис языка Haskell, а семантика определяется заложенными в проблемно-ориентированные программные сущности правилами.

Разработкой языка занимается международный коллектив, в который входят ведущие специалисты в области квантовых вычислений и функционального программирования. Это, кстати, также подтверждает идею, высказанную во введении и первой главе этой книги идею о том, что именно функциональное программирование стоит ближе всего к модели квантовых вычислений.

Основная программная сущность языка Quipper — монада `Circ`, которая представляет квантовую схему. Именно в рамках этой монады производятся квантовые вычисления. Семантика языка Quipper предполагает три типа операций, которые могут быть применены к квантовому регистру (набору кубитов), это — инициализация, унитарное преобразование и измерение. Кубиты же могут быть основными (рабочими) и вспомогательными.

Несмотря на то, что язык Quipper реализован как встроенный язык программирования на базе языка Haskell, для него уже разработаны специальные утилиты и инструменты, которые облегчают программисту его труд (в частности, это, конечно же, симулятор квантового вычислительного устройства,

но также и всевозможные утилиты для построения квантового оракула из имеющейся классической функции, подсчёта количества гейтов в квантовой схеме, обращения квантовой схемы и т. д.). Кроме того в язык внедрено несколько новых служебных слов, необходимых для учёта концепций модели квантовых вычислений.

Использование базового языка программирования (в данном случае — языка Haskell), по словам авторов языка Quipper, имеет множество преимуществ (например, развитая инфраструктура), однако несёт и некоторое количество потенциальных проблем. В частности, при разработке квантовых программ есть риск «свалиться» в написание программ на языке Haskell, не используя специально введённые идиомы языка Quipper, что сделает реализованные алгоритмы переносимыми.

Кроме того, авторы языка Quipper отмечают, что в рамках языка Haskell им очень недостаёт двух важнейших идиом программирования, которые крайне требуются в модели квантовых вычислений — это *линейные типы* и *зависимые типы* (это странно, в языке Haskell есть зависимые типы, по крайней мере, в виде одного из расширений языка — прим. авт.). Это значит, что некоторые свойства квантовых программ придётся проверять на стадии исполнения, хотя они могли бы быть легко проверены на стадии компиляции.

Квантовые схемы, реализованные в языке Quipper, являются расширенными по сравнению со схемами, описываемыми в теоретической модели квантовых вычислений. Во-первых, в схемах языка Quipper разработчик может использовать как классические, так и квантовые типы данных, равно как и операции над ними. Классические данные могут управлять квантовыми операциями (но не наоборот, то есть квантовые данные не могут управлять классическими операциями). Как

отмечалось выше, кубиты и их регистры могут быть измерены, и в результате измерения квантовые данные преобразуются в классические. Также в схемах могут быть использованы так называемые *вспомогательные кубиты*, то есть такие кубиты, область работы с которыми явно определена, а потому их можно инициализировать и высвободить в определённом интервале времени работы с квантовой схемой.

Теперь имеет смысл перейти к описанию непосредственно особенностей и различных нюансов языка Quipper, в том числе и в их сравнении с языком Haskell.

Решение нескольких простых задач

Введение в язык Quipper в целом осуществлено. Конечно, в таком небольшом обзоре сложно уместить описание всех его возможностей, поэтому заинтересованного читателя можно лишь перенаправить на официальный сайт этого языка программирования. Здесь же мы рассмотрим решение некоторых простых (если не сказать, «простейших») задач в рамках модели квантовых вычислений с использованием языка Quipper.

Однако прежде чем рассматривать сами задачи, необходимо развернуть инструментарий для непосредственной работы. К этому инструментарию относятся следующие программные средства:

1. Haskell Platform, куда включены компилятор GHC и средство развёртывания приложений и библиотек Cabal.
2. Все дополнительные библиотеки, которые необходимы для корректной работы языка Quipper.
3. Для операционной среды Windows также потребуются специальные утилиты, которые позволят исполнять POSIX-команды.
4. И, наконец, поставочный пакет самого языка Quipper.

Всё вышеперечисленное можно свободно скачать из сети Интернет. Далее описывается процесс развёртывания среды разработки языка Quipper для операционной системы Windows, при этом я полагаю, что те, кто использует иную операционную систему, вполне способны самостоятельно развернуть всё необходимое для работы.

На момент написания книги последней стабильной версией Haskell Platform является 2013.2.0.0, и её можно получить по адресу <http://www.haskell.org/platform/>. После получения дистрибутива его необходимо установить, следуя стандартному мастеру установки и отвечая на все его вопросы простым нажатием на кнопку «Далее». После установки будет доступен компилятор языка Haskell GHC последней версии, а также огромное множество дополнительных инструментов для разработки.

Поскольку Haskell Platform установлен, необходимо добавить все необходимые для языка Quipper библиотеки. Однако перед этим следует актуализировать версию утилиты Cabal, что делается при помощи последовательного запуска в командной строке терминала следующих команд:

```
C:\> cabal update
```

(эта команда актуализирует список доступных для установки пакетов языка Haskell, а также их последние версии), и

```
C:\> cabal install cabal-install
```

(и сама утилита Cabal наверняка рекомендует это сделать после актуализации пакетов). Выполнение этой команды займёт приличное время, и для её работы требуется подключение к сети Интернет, поскольку утилита Cabal самостоятельно скачивает всё необходимое для работы.

Теперь при помощи последовательного вызова в командной строке команды:

```
C:\> cabal install *
```

где символ `*` обозначает следующие библиотеки (написание должно быть именно таким, с точностью до регистра символов):

- `random`
- `mtl`
- `primes`
- `Lattices`
- `zlib`
- `easyrender`
- `fixedprec`
- `newsynth`
- `containers`
- `set-monad`
- `QuickCheck`

При установке этих библиотек утилита Cabal сравнивает номера текущих версий, и если текущая установленная версия достаточна для работы (либо является последней доступной), то установка не осуществляется. В процессе установки перечисленных библиотек такое может возникать, ничего страшного в появлении подобных уведомлений нет.

После установки всех необходимых библиотек также надо поставить среду MSYS, инсталляционный пакет для которой можно получить по адресу <http://downloads.sourceforge.net/mingw/MSYS-1.0.11.exe>. После установки (рекомендуется использовать путь по умолчанию; по крайней мере, в пути к этому набору утилит не должно быть пробельных символов) необходимо ответить «Нет» на вопрос о наличии MinGW, а также прописать путь к подкаталогу `bin` в переменную окружения `PATH`. После этого утилиты, необходимые для обеспечения совместимости со стандартом POSIX, будут доступны, и можно будет перейти к развёртыванию самого пакета для языка Quipper.

Инсталляционный пакет языка Quipper представляет собой простой заархивированный каталог, который содержит в себе всю поставку всех исходных файлов на языке Haskell, в которых определяется проблемно-ориентированный язык, приводятся многочисленные примеры, а также предлагаются для изучения семь полноценно реализованных квантовых алгоритмов. Скачать этот пакет можно с официальной страницы языка Quipper по адресу <http://www.mscs.dal.ca/~selinger/quipper/>. На момент написания книги была доступна версия 0.6.

Проще всего разархивировать инсталляционный архив в корень диска, в каталог «C:\quipper-0.6». После разархивирования необходимо прописать в переменную окружения PATH путь «C:\quipper-0.6\quipper\scripts» (или соответствующий, если разархивирование пакета было произведено в иной каталог). После этих манипуляций язык Quipper будет доступен для работы. Он поставляется, как и компилятор GHC языка Haskell, с компилятором (quipper) и интерпретатором (quipperi), хотя, конечно же, это просто обвязки в виде BAT-файлов над компилятором GHC и интерпретатором GHCi соответственно.

Чтобы запустить режим интерпретации quipperi необходимо зайти в терминал MSYS, в котором перейти в каталог, где хранятся исходники для работы (например, имя файла Example.hs), после чего выполнить команду:

```
quipperi Example.hs
```

в результате которой будет запущен интерпретатор GHCi с подключением всех необходимых модулей языка Quipper. После этого можно начинать работать так, как можно работать непосредственно в GHCi.

Из-за того, что установочный пакет языка Quipper оформлен не через систему развёртывания Cabal, при работе

с компилятором и интерпретатором GHC возникают определённые сложности. Компилятор GHC «не видит» библиотек языка Quipper, поэтому если есть потребность запустить именно GHC или GHCi, это необходимо делать с указанием пути к библиотеке языка Quipper. Например:

```
ghci -ic:/quipper-0.6
```

если язык Quipper установлен в каталоге «C:\quipper-0.6».

Надо отметить, что создатели языка программирования Quipper уже задумались над поставкой библиотек языка в стиле Cabal, однако когда выйдет новый инсталляционный пакет — остаётся неизвестным.

Теперь перейдём к задачам, но перед этим хотелось бы отметить, что в этом разделе приводится несколько простейших задач, которые можно решить на языке Quipper, при этом приводимые определения функций в целом не объясняются (они слишком просты). Более детальное введение в библиотеку языка будет дано в следующем разделе.

Прежде всего, давайте посмотрим на некоторые базовые операции. Как можно реализовать классические вычисления в рамках модели квантовых вычислений? Понятно, что это можно сделать, и выкладки из Главы 1 ясно показывают на то, что квантовый компьютер может эмулировать классический не хуже, чем с полиномиальным замедлением. Теперь посмотрим, как это можно доказать при помощи языка Quipper. Операция НЕ в модели квантовых вычислений есть — это, просто-напросто, гейт X. А вот как определяется операция И:

```
andGate :: (Qubit, Qubit) -> Circ Qubit
andGate (a, b) = do c <- qinit False
                  qnot_at c `controlled` [a, b]
                  return c
```

Если посмотреть на диаграмму квантовой схемы, генерируемой по данному коду, то она выглядит вот так:

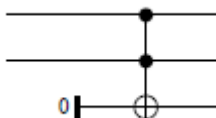


Рис. 18. Диаграмма квантовой схемы для функции `andGate`

Здесь очень хорошо видно, что это банальное использование элемента Тоффли в квантовом представлении. И, таким образом, имея в наличии операции И и НЕ, можно построить произвольную классическую функцию, которая может быть построена в рамках классической вычислительной парадигмы. Другое дело, что такое построение не всегда оптимально, однако же факт возможности построения доказан.

Язык Qirrer позволяет с лёгкостью оперировать теми же идиомами, которые доступны в языке Haskell. Например, такая естественная для последнего конструкция, как список, также может быть использована для работы с кубитами. Например, пусть надо применить операцию И ко всем кубитам в заданном списке. Если бы у нас были классические биты, то можно было бы воспользоваться свёрткой списка по функции (`&&`) с начальным значением `True`, и в результате получилось бы значение `True` тогда и только тогда, когда все биты в списке принимают это же значение. Но, поскольку у нас квантовые вычисления и необходимо иметь обратимую схему, так просто свёртку не устроить. Надо писать что-то типа такой функции:

```
andList :: [Qubit] -> Circ Qubit
andList []      = qinit True
andList [q]     = return q
andList (q:qs) = do c <- andList qs
                   r <- andGate (c, q)
```

```
return r
```

В результате генерации будет построена примерно следующая квантовая схема (на примере списка из 10 кубитов):

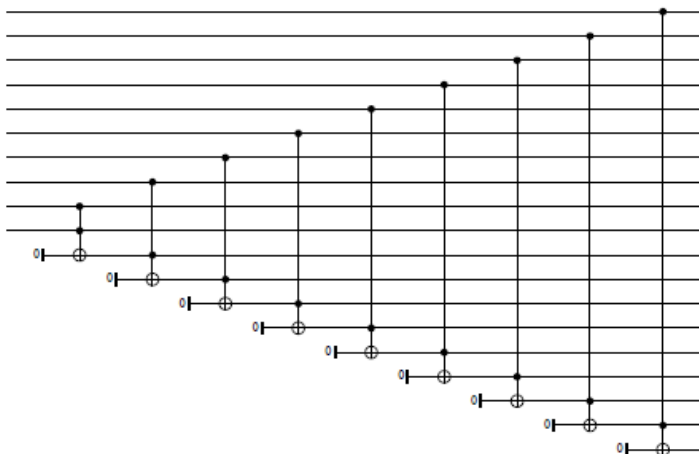


Рис. 19. Диаграмма квантовой схемы для операции конъюнкции списка из 10 кубитов

Результат операции возвращается в последнем (самом нижнем на диаграмме) кубите.

Несмотря на то, что авторы языка Quipper в своих обучающих материалах всегда используют «квази-императивный» стиль с использованием `do`-нотации, с монадой `Circ` вполне можно производить все монадические операции, скрывающие явную рекурсию и т. д. Например, альтернативным вариантом для функции `andList`, в котором используется стандартная функция для свёртки в рамках заданной монады `foldM`, будет следующая:

```
andList' :: [Qubit] -> Circ Qubit
andList' qs = do i <- qinit True
               foldM (curry andGate) i qs
```

Впрочем, эта функция генерирует иную квантовую схему, нежели показанная на предыдущем рисунке, поскольку свёртка осуществляется в ином порядке, чем в случае явной рекурсии, а также используется начальное значение, полученное из выражения (`qinit True`). Это значит, что в схеме будет явно присутствовать кубит $|1\rangle$. Диаграмма квантовой схемы для тех же 10 кубитов показана на следующем рисунке.

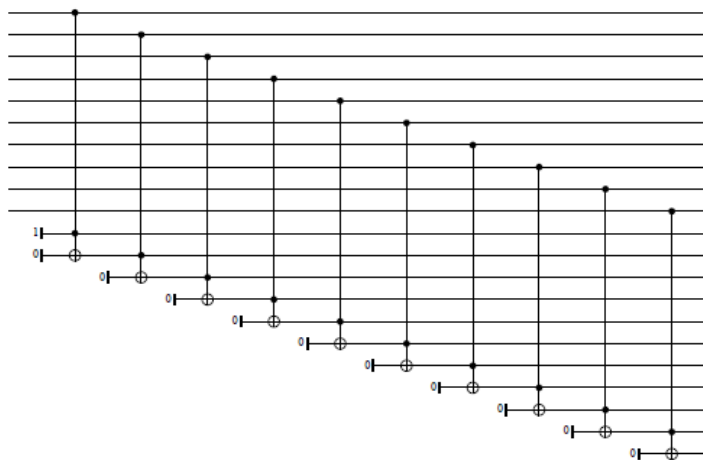


Рис. 19. Диаграмма альтернативной квантовой схемы для операции конъюнкции списка из 10 кубитов

Очевидно, что схемы, представленные на двух рисунках, выполняют одинаковую операцию и на произвольных списках дают одинаковый результат. Более того, для функции `andList` можно также сгенерировать квантовую схему, в которую будет фигурировать кубит $|1\rangle$ — для этого достаточно удалить второй клюз определения функции (она и без него будет прекрасно работать).

Хорошо, с булевой логикой на кубитах справились. А как насчёт арифметики? Хотя бы базовой? В поставке языка Quip-

рег есть целый модуль `Quipper.Arith`, который содержит описания функций и методов специальных классов для выполнения всех необходимых в разработке арифметических операций. Рассмотрим базовые.

Для определения арифметических операций в языке `Quipper` используется класс типов, аналогичный классу `Num` в стандартной поставке языка `Haskell`, только вот называется он, естественно, `QNum`. На текущий момент экземпляры этого класса реализованы только для типа `QDInt`, который представляет собой обёртку над кубитом `Qubit` для представления целых чисел ограниченной длины.

В следующей таблице рассматриваются три базовые арифметические операции (без деления) над двумя кубитами.

Таблица 13. Диаграммы квантовых схем для трёх базовых арифметических операций над двумя кубитами

Операция	Наименование метода	Квантовая схема
Сложение	<code>q_add</code>	
Вычитание	<code>q_sub</code>	
Умножение	<code>q_mult</code>	

Это интересно, но на квантовом уровне сложение ничем не отличается от вычитания, а умножение — это та же логическая операция И (кто бы сомневался?). Но, естественно, для большого числа кубитов всё будет не так. Ну и заинтересованному читателю, как обычно, рекомендуется уже самостоятельно исследовать такие операции, как взятие модуля числа (метод `q_abs`), изменение знака числа (`q_negate`) и вычисление знака числа (`q_signum`). Все эти методы также представлены в классе `QNum`.

В подкаталоге `tests` каталога развёрнутой поставки языка Quipper есть ещё большое количество примеров, которые можно самостоятельно изучить. К сожалению, все они поставляются без какой-либо документации и даже комментариев, но вдумчивому читателю, заинтересовавшемуся этим языком программирования, рекомендуется начать его изучение именно с этого каталога и файлов с примерами в нём.

Рассмотрев эти задачи, можно подумать, что язык Quipper несколько громоздок и многословен, хотя бы и по сравнению с тем фреймворком, самые начала которого были реализованы в прошлой главе. Однако надо отметить, что в языке Quipper используется своя крайне продуманная парадигма, которая позволяет управлять реальным квантовым вычислительным устройством, когда оно будет реализовано, что на текущем этапе совершенно нельзя сказать про тот фреймворк. В первую очередь, это позволяет осуществлять схемная модель языка, которая предполагает три этапа разработки, а не два. Функция преобразуется в квантовую схему, которая зависит от уже известных входных



QuipperSimple.hs

*К книге прилагается файл
с исходным кодом на языке
Quipper с перечисленными
в разделе примерами.*

параметров функции, и это главное и мощнейшее отличие. Этот этап генерации позволяет подготовить квантовое вычислительное устройство к непосредственно этапу вычислений: инициализировать все необходимые кубиты, настроить гейты, а потом запустить процесс вычислений. Тем самым достигается основная цель языка программирования — мы можем реализовывать на нём реальные задачи.

От простого к сложному

Итак, как уже было написано выше, все вычисления на языке Qirreg происходят в рамках специально разработанной монады `Circ`, которая представляет описание квантовой схемы. Данная монада имеет одну важнейшую особенность — те побочные эффекты, которые она скрывает в себе, представляют собой физическое взаимодействие с квантовым вычислительным устройством, а потому подчиняются законам квантовой механики. Это очень важно понимать монаду `Circ` как бы осуществляет ввод/вывод с квантовым вычислительным устройством (или его эмулятором).

Если какая-либо функция возвращает определённое значение в рамках этой монады, то это значит, что в программу возвращено *актуальное состояние* физического кубита, с которым работала эта функция. Для примера можно рассмотреть простейший код:

```
plusMinus :: Bool -> Circ Qubit
plusMinus b = do q <- qinit b
               r <- hadamard q
               return r
```

Внимательный читатель уже понял, что функция `plusMinus` возвращает кубит $|+\rangle$ или $|-\rangle$ в зависимости от входного параметра (кстати, здесь `b` — это именно параметр, а не входной аргумент схемы). Разберём действие функции по шагам:

1. Функция `qinit` осуществляет инициализацию одного кубита в общей памяти квантового вычислительного устройства в зависимости от входного значения. Если получено значение `False`, то кубит инициализируется в значение $|0\rangle$, а если входным значением является `True`, то кубит, соответственно, инициализируется в значение $|1\rangle$. Ссылка на этот кубит и его текущее состояние возвращается и сохраняется в образце `q`.
2. При помощи гейта `hadamard` (гейт Адамара) кубит, ссылка на который хранится в образце `q`, преобразуется в базис $\{|+\rangle, |-\rangle\}$. Преобразуется именно физический кубит в памяти квантового вычислительного устройства, и ссылка на него, а также его новое состояние возвращаются. Надо отметить, что в этом случае в образце `q` уже находится неактуальная информация, его нельзя использовать на чтение (на запись — пожалуйста). А новое состояние кубита сохраняется в образце `r`.
3. Далее состояние кубита возвращается в рамках монады `Circ`, которая устроена так же, как и стандартная монада `IO` — выйти из неё нельзя.

Таким образом, функция `plusMinus` получает на вход логическое значение `True` или `False` и в зависимости от этого возвращает ссылку на физический кубит, находящийся в квантовом состоянии $|+\rangle$ или $|-\rangle$ соответственно.

Интерес представляет реализация функции связывания двух кубитов в пару. Её определение на языке `Quipper` выглядит следующим образом:

```
entangle :: Qubit -> Circ (Qubit, Qubit)
entangle a = do b <- qinit False
               b <- qnot b `controlled` a
               return (a, b)
```

Это определение не соответствует тому, которое было сделано в рамках разработки фреймворка квантовых вычислений, описанного в предыдущей главе. Здесь функция `entangle` принимает на вход кубит и связывает его с вновь проинициализированным кубитом, который имеет начальное состояние $|0\rangle$ (результат выражения `qinit False`). Вторая строка блока `do` описывает гейт `CNOT` — вновь проинициализированный кубит `b` подвергается действию гейта `X` (оператор `qnot`) при контроле (инфиксный оператор `controlled`) входным кубитом `a`. Функция возвращает пару кубитов. Таким образом, на физическом уровне её действие заключается в получении некоторого кубита в суперпозиции квантовых состояний $\alpha|0\rangle + \beta|1\rangle$, и возвращении пары кубитов в квантовом состоянии $\alpha|00\rangle + \beta|11\rangle$.

Само собой разумеется, что создаваемые функции можно дальше использовать в построении квантовых схем. Например, две ранее определённые функции `plusMinus` и `entangle` можно использовать для получения состояния Белла $|\Phi^+\rangle$. Следующая функция реализует это:

```
bellPhiPlus :: Circ (Qubit, Qubit)
bellPhiPlus = do a <- plusMinus False
                (a, b) <- entangle a
                return (a, b)
```

или ещё проще:

```
bellPhiPlus :: Circ (Qubit, Qubit)
bellPhiPlus = plusMinus False >>= entangle
```

Эта функция возвращает пару кубитов в связанном состоянии $|\Phi^+\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$. Первое определение использует явную `do`-нотацию, в то время как второе использует монадический оператор (`>>=`), выполняющий те же самые действия, что и первое определение. Использование того или иного способа записи зависит от стиля написания исходного кода.

Было бы очень сложно писать программы на языке Quipper, если бы он позволял манипулировать только кубитами и их кортежами (как в ранее определённых функциях `entangle` и `bellPhiPlus`). Но, поскольку базовым языком всё-таки является язык Haskell, здесь не обошлось без создания класса типов для написания полиморфных функций с использованием типовых переменных в их типах.

Общая идея такова. Функции языка Quipper могут обрабатывать любую однородную структуру данных, которая содержит кубиты (тип `Qubit`), биты (тип `Bit`) или булевы значения (тип `Bool`). Для этих целей определён класс `QShape`, и любая функция, которая реализуется полиморфной, должна иметь в своей сигнатуре ограничение $(QShape\ qa\ ca\ ba) \Rightarrow$. Здесь `qa` — квантовый тип данных, `ca` — классический, `ba` — булевый. Под типом данных здесь понимается структура данных, которая содержит только какой-либо один из таких «листьевых» типов, либо иные структуры с этим же типом. Например, такая структура как $(Qubit, [Qubit])$, то есть пара из кубита и списка кубитов, будет являться квантовой структурой данных. Таким образом, можно строить кортежи, списки, деревья и т. д., содержащие кубиты, классические биты и булевы значения истинности. Но смешивать в одной структуре разные «листьевые» типы нельзя (конечно, синтаксис разрешит определить такую смешанную структуру, но её нельзя будет обработать полиморфными функциями класса `QShape`).

Вот, к примеру, как можно переписать предыдущие четыре функции в полиморфном варианте:

```
plusMinus' :: (QShape ba qa ca) => ba -> Circ qa
plusMinus' a = do
  qs <- qinit a
  qs <- mapUnary hadamard qs
  return qs

entangle' :: (QShape ba qa ca) => qa -> Circ (qa, qa)
```

```
entangle' qa = do
  qb <- qinit (qc_false qa)
  (qb, qa) <- mapBinary controlled_not qb qa
  return (qa, qb)

bellPhiPlus' :: (QShape ba qa ca) => ba -> Circ (qa, qa)
bellPhiPlus' s = do
  qa <- plusMinus' s
  (qa, qb) <- entangle' qa
  return (qa, qb)
```

Здесь необходимы многочисленные пояснения. Как видно, в сигнатурах функций конкретные типы заменены на типовые переменные. В частности, тип `Bit` заменён на переменную `ca`, а тип `Qubit` — на переменную `qa`.

Таким образом, функция `plusMinus'` получает на вход некоторую структуру данных, в которых листовым типом всегда является тип `Bit`. Ну а возвращает она соответствующую структуру данных, в которой значения типа `Bit` заменены на значения типа `Qubit`. Функция `qinit` уже является обобщённой, а вот применение гейта H при помощи функции `hadamard` необходимо оформить в виде отображения. Функция `mapUnary` принимает на вход функцию типа `Qubit -> Circ Qubit` и применяет её к каждому кубиту внутри заданной квантовой структуры данных. Это квантовый аналог метода `fmap` с действием в монаде `Circ`, если так можно выразиться.

Далее, в определении функции `entangle'` используется функция `qc_false`, которая получает на вход некоторую квантовую структуру данных и создаёт на её основе соответствующую ей булеву структуру данных, в которой все листовые значения установлены в `False`. Также функция `mapBinary` аналогична функции `mapUnary`, но она применяет функцию с типом `Qubit -> Qubit -> Circ (Qubit, Qubit)` к каждой соответствующей паре кубитов в двух одинаковых квантовых структу-

рах данных. Ну и вместо ручного контроля операции НЕ используется уже определённый оператор `controlled_not`.

Определение полиморфной функции `bellPhiPlus'` аналогично первому варианту с заменой вызовов функций `plusMinus` и `entangle` на их полиморфные варианты `plusMinus'` и `entangle'`.

Было бы странно, если бы мощный функциональный язык программирования, которым является язык Haskell, не позволял бы использовать рекурсивные определения в своём подязыке, которым является язык Quipper. Однако в этом вопросе не всё так просто. Дело в том, что квантовые вычисления должны быть обратимыми, а это обозначает в свой черёд, что любая квантовая схема должна быть направлена только от входа к выходу, в ней не может быть петель возврата управления. А циклы и рекурсия — это именно такие петли возврата. Так что здесь требуется несколько более тонкий подход.

Поскольку в языке Quipper принято, что есть три разных фазы написания и запуска программы (а не две, как в классическом случае), то в данном случае определение рекурсивных функций возможно на этапе до генерации схемы (и тем более до её исполнения). Можно написать рекурсивную функцию для генерации схемы, причём рекурсия может быть основана на любом из параметров, значение которого известно до старта генерации схемы. Этим самым будет обеспечено то, что сгенерированная квантовая схема будет развёрнута в линейную последовательность связанных друг с другом гейтов. И в ней не будет никаких петель с возвратами.

Теперь немного усложним наше рассмотрение. Давайте посмотрим на алгоритм квантового преобразования Фурье. В классическом случае под преобразованием Фурье понимается некоторая операция разложения заданной вещественной функции на элементарные составляющие, каждая из которых пред-

ставляет собой так называемые гармонические колебания с разными частотами, а амплитуды этих колебаний описываются другой функцией. То есть, по своей сути, преобразование Фурье — это выражение одной функции через другую некоторым специальным образом. Формула этого разложения следующая:

$$\hat{f}(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{-ix\omega} dx$$

Физический смысл этого преобразования заключается в том, что заданная функция может быть представлена в виде бесконечной суммы гармонических колебаний $e^{-ix\omega}$, при этом амплитуда этих колебаний задаётся через другую функцию следующим образом: $\frac{1}{\sqrt{2\pi}} |f(x)|$. Бесконечность суммы соответствует абсолютной точности выражения. Если же точность может быть ниже абсолютной, то и сумма гармонических колебаний может быть конечной. Другими словами, чем больше в сумме членов, тем точнее выражение исходной функции.

Большее прикладное значение имеет дискретное преобразование Фурье, в котором интеграл заменён на сумму, которая в любом случае конечна. Данное преобразование раскладывает заданную величину на конечную сумму гармоник, у каждой из которой задана определённая амплитуда. Формула дискретного преобразования Фурье имеет следующий вид:

$$X_k = \frac{1}{N} \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn}$$

Важной особенностью преобразования Фурье является то, что оно обратимо. Можно выполнять как прямое (выражать X через x), так и обратное (выражать x через X) преобразование. Обратимость преобразования сразу же наталкивает на мысль о том, что его можно реализовать в рамках модели квантовых

вычислений. И действительно, квантовое преобразование Фурье нашло своё важнейшее место в модели и используется во многих квантовых алгоритмах.

Квантовое преобразование Фурье является унитарным оператором, который действует на базисных векторах. Его формула следующая:

$$|a\rangle = \frac{1}{\sqrt{N}} \sum_{c=0}^{N-1} e^{-\frac{2\pi i a c}{N}} |c\rangle$$

Само собой разумеется, что это преобразование может быть представлено в виде унитарной матрицы следующего вида (естественно, что она квадратная):

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & & 1 \\ 1 & e^{-\frac{2\pi i}{N}} & e^{-\frac{4\pi i}{N}} & e^{-\frac{6\pi i}{N}} & & e^{-\frac{2\pi i}{N}(N-1)} \\ 1 & e^{-\frac{4\pi i}{N}} & e^{-\frac{8\pi i}{N}} & e^{-\frac{12\pi i}{N}} & \dots & e^{-\frac{2\pi i}{N}2(N-1)} \\ 1 & e^{-\frac{6\pi i}{N}} & e^{-\frac{12\pi i}{N}} & e^{-\frac{18\pi i}{N}} & & e^{-\frac{2\pi i}{N}3(N-1)} \\ & \vdots & & & \ddots & \vdots \\ 1 & e^{-\frac{2\pi i}{N}(N-1)} & e^{-\frac{2\pi i}{N}2(N-1)} & e^{-\frac{2\pi i}{N}3(N-1)} & \dots & e^{-\frac{2\pi i}{N}(N-1)^2} \end{pmatrix}$$

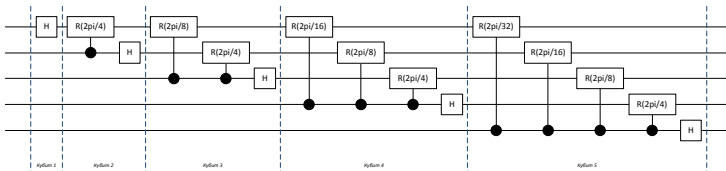
Для быстроты вычисления можно дать формулу расчёта любого её элемента в зависимости от его позиции: $A(m, n) = \exp(-2\pi i \frac{(m-1)(n-1)}{N})$, где нумерация позиций по столбцам и строкам начинается с 1. Это очень удивительно, но если переложить эту матрицу на квантовую схему, то окажется, что вся кажущаяся громоздкость сводится к последовательности простейших гейтов. Вся вышеперечисленная «математика» приведена здесь только лишь для сравнения с тем, как квантовое преобразование Фурье будет выглядеть в виде квантовой схемы и соответствующей ей функции на языке Quipper.

Итак, попробуем реализовать квантовое преобразование Фурье на языке Qirreg. Для этого рассмотрим его рекуррентное построение. Для одного кубита квантовое преобразование Фурье заключается в применении гейта H к этому кубиту. И, действительно, если посмотреть на формулу квантового преобразования Фурье и попытаться разложить по ней один кубит $|\varphi\rangle$ в базисных векторах, то получится следующая цепочка преобразований:

$$\begin{aligned} |\varphi\rangle &= \frac{1}{\sqrt{2}} \sum_{c=0}^1 e^{-\frac{2\pi i \varphi c}{2}} |c\rangle = \frac{1}{\sqrt{2}} e^{-\frac{2\pi i \varphi * 0}{2}} |0\rangle + \frac{1}{\sqrt{2}} e^{-\frac{2\pi i \varphi * 1}{2}} |1\rangle \\ &= \frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} e^{-\pi i \varphi} |1\rangle \end{aligned}$$

Эту цепочку преобразований можно рассмотреть для двух случаев. Если $|\varphi\rangle = |0\rangle$, то в результате применения этой цепочки получится $\frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |1\rangle$. Если же $|\varphi\rangle = |1\rangle$, то в результате получится $\frac{1}{\sqrt{2}} |0\rangle - \frac{1}{\sqrt{2}} |1\rangle$. А это и соответствует применению именно преобразования Адамара (гейт H).

Далее на сцену выступает рекуррентное соотношение. Квантовое преобразование Фурье для $(n + 1)$ кубитов равно квантовому преобразованию Фурье для n кубитов, после которого следует серия вращений вокруг оси Z на сфере Блоха. Эти вращения контролируются $(n + 1)$ -ым кубитом, который после этого опять подвергается преобразованию Адамара. Само вращение производится на угол $\frac{2\pi i}{2^m}$, где m — «расстояние» до вращаемого кубита от кубита $(n + 1)$, то есть для кубита n расстояние $m = 2$, для кубита $(n - 1)$ расстояние $m = 3$ и т. д. Проще всего это рекуррентное соотношение поясняет квантовая схема. Приведём пример для пяти кубитов:



**Рис. 20. Диаграмма квантового преобразования Фурье
для пяти кубитов**

Как раз при помощи представленной схемы видно, что она очень легко может быть записана при помощи рекурсивных функций. И действительно, попробуем описать эту схему в общем виде. Для начала определим функцию `rotations`, которая генерирует набор управляемых гейтов для вращения набора кубитов из заданного списка вокруг оси Z . Как полагается, эта функция должна получать на вход управляющий кубит, список кубитов, которые подвергаются вращению, а также будем передавать ей на вход целочисленный параметр, который отвечает за показатель степени в гейте вращения. Этот параметр будет задавать номер очередного кубита в списке, который подвергается вращению, на основании какового числа будет вычисляться расстояние m . Вот определение:

```
rotations :: Qubit -> [Qubit] -> Int -> Circ [Qubit]
rotations _ [] _ = return []
rotations c (q:qs) n = do qs' <- rotations c qs n
                           let m = ((n + 1) - length qs)
                           q' <- rGate m q `controlled` c
                           return (q':qs')
```

Определение довольно простое. Сначала мы получаем список кубитов после применения гейта вращения к хвосту списка кубитов, для чего используется та же самая функция, причём управляющий кубит, естественно, тот же, равно как и числовой параметр, поскольку нам каждый раз необходимо увеличивать показатель степени (см. диаграмму). Затем вычисляется расстояние m , формула для которого в точности отражает предыду-

щие рассуждения. Далее функция `rGate`, которая уже определена в библиотеке стандартных гейтов языка `Qirrer`, как раз представляет собой гейт для вращения вокруг оси Z на угол $\frac{2\pi i}{2^m}$. Ну и, наконец, окончательно возвращается список всех кубитов, подвергшихся вращению.

Теперь написать функцию для генерации квантовой схемы квантового преобразования Фурье не представляет никакой сложности. Вот её определение:

```
qft' :: [Qubit] -> Circ [Qubit]
qft' [] = return []
qft' [x] = do hadamard x
              return [x]
qft' (x:xs) = do xs'' <- qft' xs
                  xs' <- rotations x xs'' (length xs'')
                  x' <- hadamard x
                  return (x':xs')
```

У этой функции есть одна проблема (поэтому она названа с апострофом, причина чего станет понятна далее) — она возвращает обращённый список кубитов. В связи с этим будет менее конфузно написать специальную обёртку для неё:

```
qft :: [Qubit] -> Circ [Qubit]
qft qs = do qs <- qft' (reverse qs)
           return qs
```

Вот именно эта функция и генерирует квантовые схемы в зависимости от значения своего входного параметра, пример которой приведён на предыдущем рисунке.

Очень часто при реализации квантовых схем встречается ситуация, когда одна и та же подсхема встречается многократно. В этом случае язык `Qirrer` предоставляет для разработчика возможность оформить такую подсхему в виде специальной функции, которая на диаграмме будет обозначаться отдельным прямоугольником. Назовём такую подсхему «коробкой». Получается так, что это как бы представление чёрного ящика

для кубитов — кубит входит в такую коробку, преобразуется там (изменяется его квантовое состояние) и выходит из неё для продолжения участия в квантовой схеме. Для организации коробок используется функция высшего порядка `box`. Первым аргументом она принимает строку, которая представляет собой наименование коробки. Вторым — функцию для генерации квантовой схемы. Третьим — ту структуру данных, к которой применяется функция, второй параметр.

На этом всё.



QuipperHard.hs

*К книге прилагается файл
с исходным кодом на языке
Quipper с перечисленными
в разделе примерами.*

Дополнительные возможности

Кроме непосредственно возможностей для реализации квантовых алгоритмов и использования модели квантовых вычислений для решения произвольных вычислительных задач, язык Quipper содержит большое количество дополнительных возможностей, в том числе, облегчающих жизнь разработчику программного обеспечения.

Создатели языка Quipper не отказали себе в удовольствии реализовать в рамках этого языка множество полезных маленьких функций, которые облегчают разработчику процесс реализации и отладки программ. Эти функции не входят в состав модели квантовых вычислений, но обычно являются функциями высших порядков и предоставляют программисту различную информацию или возможности касательно квантовых схем.

Например, очень полезной в целях отладки квантовых схем является функция `print_simple` и её полиморфный вариант `print_general`. Эти функции принимают на вход функцию,

описывающую квантовую схему, и формат вывода схемы (например, PDF), а полиморфный вариант получает ещё и вид структуры данных, для которой генерируется квантовая схема. Результатом работы функций являются диаграммы квантовых схем в запрошенном формате. Например:

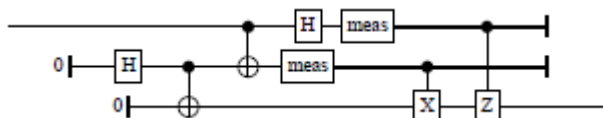


Рис. 21. Пример диаграммы квантовой схемы, генерируемой функцией `print_simple` языка Quipper

Данная диаграмма показывает схему так называемой квантовой телепортации, когда состояние одного кубита переносится на другой кубит. Честно говоря, этот процесс не является квантовым алгоритмом, как таковым, однако весь он полностью может быть описан в терминах модели квантовых вычислений, поэтому его обычно приводят в качестве одного из примеров в литературе по теме. Однако в данной книге этот процесс описываться не будет, поскольку его описание выходит за рамки настоящего издания.

Приведённая выше диаграмма квантовой схемы вполне наглядна. Однако если представить себе квантовые схемы, которые содержат десятки кубитов и сотни гейтов, то без усилий разобраться в диаграммах таких квантовых схем будет тяжело. Ради разработчика создатели языка Quipper внедрили в него специальный оператор, который позволяет вносить на диаграммы специальные пометки. Этот оператор не несёт никакой физической (квантово-механической) сути, у него нет побочных эффектов на квантовое вычислительное устройство. Единственное его предназначение — это упрощение чтения диаграммы квантовой схемы.

Оператор этот, `comment_with_label`, принимает на вход три параметра. Первым является метка (строка), которая ставится на диаграмме поперёк всех кубитов, которые затрагивает этот оператор. Вторым параметром как раз и является ссылка на квантовый регистр, который необходимо откомментировать. Третьим же параметром являются метки на проводах кубитов, при этом форма этого параметра должна соответствовать форме второго параметра, а полями являются строки, представляющие собой метки. Вот так, например, выглядит диаграмма квантовой схемы для телепортации четырёх кубитов:

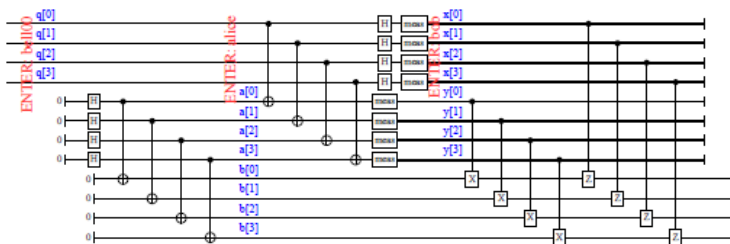


Рис. 22. Диаграмма квантовой схемы для телепортации списка из четырёх кубитов

В частности, для первой метки («ENTER: bell00») использован следующий вызов оператора `comment_with_label`:

```
comment_with_label "ENTER: bell00" q "q"
```

Кроме функции для печати квантовой схемы в виде диаграммы в библиотеке языка Quipper имеется огромное количество функций, которые действуют на всю квантовую схему в целом. Все эти функции могут быть использованы для работы со схемами до генерации самих схем, то есть, по своей сути, все эти функции являются в терминах языка Haskell, функциями высшего порядка. К числу таких функций относятся:

1. Обращение квантовой схемы.

2. Подсчёт количества кубитов (основных и вспомогательных) и гейтов.
3. Преобразование схемы в тождественную схему, в которой используются только двоичные гейты (или двойные гейты и элемент Тоффоли).
4. Симуляция работы квантовой схемы.

При этом надо понимать, что тут перечислены только наиболее интересные и важные функции, но в самой библиотеке их намного больше. Формат книги не позволяет описать всю библиотеку языка Quipper, поэтому заинтересованный читатель отсылается к технической документации по языку (доступна на официальном сайте языка).

Необходимо отметить, что на текущий момент для языка Quipper создано три симулятора, которые могут исполнять квантовые программы, сгенерированные на основе функций с описанием квантовых схем. Первый симулятор может эффективно исполнять классические алгоритмы, переведённые в квантовые схемы. Вторым симулятором можно также эффективно исполнять один специальный класс квантовых схем (схемы, в которых используются только гейты из так называемой *группы Клиффорда*). А третий симулятор является универсальным, и на нём можно исполнять произвольную квантовую схему, но при этом данный симулятор использует для своей работы экспоненциальное по кубитам количество памяти и времени.

Краткие выводы

Таким образом, сегодня уже создано несколько серьёзных языков программирования, которые полностью поддерживают модель квантовых вычислений в её современном понимании. Вряд ли эта модель претерпит какие-либо изменения в будущем, поскольку экспериментальная проверка положений квантовой механики позволяет утверждать, что в приближении,

достаточном для выполнения вычислений в рамках этой модели, законы квантовой механики работают именно так, как нужно для оных вычислений. Так что уже сегодня есть средства программирования будущих квантовых вычислительных устройств. Когда они будут созданы, для имеющихся языков программирования надо будет написать специальные компиляторы и «драйверы» для низкоуровневого доступа к этим устройствам в «железе».

Наиболее перспективным видится язык программирования Quipper, несмотря на то, что язык QCL был создан раньше и на первый взгляд выглядит более серьёзно. Во-первых, язык Quipper разрабатывается коллективом авторов, большинство из которых являются специалистами с мировым именем в области компьютерных наук и информатики. Во-вторых, язык Quipper основан на понятии монада и встроен в язык программирования Haskell, который предоставляет развитую инфраструктуру для разработки. Монада определяет стратегию вычислений и определяет побочные эффекты, и это очень гармонично ложится на модель квантовых вычислений. К тому же, схемная модель языка Quipper является очень продуманной и развитой по сравнению с остальными имеющимися языками квантового программирования независимо от наличия у них реализации.

Сейчас можно говорить о том, что сегодня изучение языков для квантового программирования может показаться преждевременным. Однако ситуация выглядит так, что в любой момент может быть объявлено о получении работающего прототипа квантового вычислительного устройства, и тогда начнётся «гонка вооружений» в среде разработчиков. Преимущество получают те, кто вовремя сориентировался и постиг тайны модели квантовых вычислений.

Так что я советую уже сейчас изучать как саму модель квантовых вычислений, так и какой-нибудь язык для программирования в её рамках. Само собой разумеется, моим личным предпочтением является язык Quipper. Во-первых, у этого языка уже сегодня есть полноценная реализация, которая кратко описана в этой главе. Во-вторых, язык Quipper разрабатывается большим числом сотрудников, большинство из которых являются очень известными личностями в области информатики и компьютерных наук. В-третьих, разработка языка поддерживается государственными институтами нескольких стран, так что у него точно есть будущее.

Ну и, наконец, для тех, кто вполне знает язык программирования Haskell, изучение языка Quipper займёт от силы пару часов, поскольку он является встроенным языком и его парадигма полностью гармонично влита в структуру языка Haskell.

Глава 4.

Детальное рассмотрение некоторых квантовых алгоритмов

*На свете есть столь серьёзные вещи,
что говорить о них можно
только шутя.*

Нильс Бор

Теперь мы перешли к наиболее интересной части нашего повествования, в которой мы рассмотрим отдельные квантовые алгоритмы. Некоторые из них лежат в основе всех разработанных к настоящему моменту квантовых алгоритмов. И если все вновь разработанные квантовые алгоритмы были и являются, скорее, плодом инженерной мысли, то есть были как бы спроектированы, если можно так выразиться, то рассматриваемые

в данной главе алгоритмы явились, судя по всему, в виде вдохновения, черпанного из источника квантовой музыки.

В этой главе мы рассмотрим пять алгоритмов, которые обычно предлагаются в той или иной комбинации во всех книгах и статьях по квантовой модели вычислений (см. также обзор литературы). В первую очередь, это алгоритм Гровера, в котором реализуется идиома квантового случайного блуждания с усилением амплитуды требуемых при поиске значений за счёт интерференции. Во-вторых, алгоритм Дойча и его расширение — алгоритм Дойча-Йожи, в которых используется идиома квантового параллелизма, тоже являющаяся отличительной особенностью модели квантовых вычислений. В-третьих, алгоритм Саймона, описывающий более практическую задачу, нежели алгоритм Дойча-Йожи.

Далее мы перейдём к рассмотрению самого интересного алгоритма, предложенного Питером Шором, — алгоритма факторизации целого числа, являющегося произведением двух нечётных простых чисел. Решение этой задачи при помощи модели квантовых вычислений показывает экспоненциальный выигрыш по сравнению с классической вычислительной моделью, а потому алгоритм является крайне интересными для изучения. Для рассмотрения этого алгоритма необходимо вспомнить о квантовом преобразовании Фурье, с которым мы уже познакомились и которое используется для поиска периода функции.

Рассмотрение всех алгоритмов, как это принято в настоящем издании, проводится с точки зрения разработчика программного обеспечения, поэтому описание даётся с минимумом физики, с умеренным количеством математики и с разъяснением алгоритмов при помощи методов кибернетики и схемотехники. Тем из читателей, кто хотел бы получить более фундаментальные представления об алгоритмах, реко-

мендуется изучить обзор литературы и перейти к источникам, более заточенным на физику и математику.

Алгоритм Гровера

Алгоритм Гровера был разработан американским математиком Ловом Гровером в 1996 году (уже намного позже того, как модель квантовых вычислений стала в моде). Этот алгоритм использует свойство *квантовой интерференции* для того, чтобы решать крайне насущную задачу по поиску значения некоторого параметра, на котором заданная функция выдаёт определённый результат.

Данный алгоритм не показывает экспоненциального выигрыша для задачи по сравнению с классической вычислительной моделью, однако для больших значений выигрыш (квадратичный) является существенным. Однако это общий алгоритм для решения довольно обобщённой задачи, и доказано, что лучшего результата в рамках модели квантовых вычислений добиться нельзя. В более частных алгоритмах (см. далее алгоритм Шора) это возможно.

Задача формулируется следующим образом. Пусть есть бинарная функция от n бинарных же аргументов, которая принимает значение 1 только на одном из них (а на остальных $2^n - 1$ значениях, стало быть, принимает 0). Необходимо найти это значение входных аргументов, имея только функцию (для квантовых вычислений данную в виде оракула, как это предполагается).

Естественно, что в классическом варианте в среднем необходимо просмотреть $\frac{2^n}{2}$ вариантов входных значений, поскольку в лучшем случае удастся найти «счастливый номер» с первой попытки, а в худшем потребуются перебрать все 2^n вариантов. А вот алгоритм Гровера позволяет сделать это

за $\frac{\pi}{4}\sqrt{2^n}$ обращений к оракулу. Понятно, что при увеличении числа входных кубитов n разница в производительности становится кардинальной. Однако, как уже написано выше, суперполиномиального увеличения производительности здесь нет и не предвидится.

По своей сути это задача неструктурированного поиска. Если есть неупорядоченный набор данных («стог сена») и требуется найти в нём какой-то один элемент, удовлетворяющий специфическому требованию (этот элемент один такой — «иголлка»), алгоритм Гровера поможет, поскольку альтернативой является классический перебор вариантов. Например, если рассмотреть аналогию с базой данных автомобилей с именами, упорядоченными по алфавиту (и пусть в ней будет взаимно однозначное соответствие между фамилией и номером автомобиля), то упорядоченным поиском является поиск номера автомобиля по фамилии (делается, например, методом дихотомии, если нет индекса). А неупорядоченным поиском является обратная задача — поиск фамилии по номеру автомобиля. В данной аналогии оракулом является функция (преобразованная соответствующим образом), которая по фамилии возвращает номер автомобиля. Тогда эта задача сводится к задаче Гровера при помощи кодирования фамилий в бинарное представление, а сама функция возвращает ответ на вопрос «Владеет ли данный автомобилист N автомобилем X ?», где N — входной параметр функции, а X — параметр алгоритма, как бы «зашиваемый» внутрь оракула при его построении. Соответственно, этот оракул возвращает значение 1 («да») только единственно для той фамилии, напротив которой в базе данных стоит искомый номер автомобиля.

Алгоритм Гровера состоит из следующих шагов:

1. *Инициализация начального состояния.* Необходимо подготовить равновероятностную суперпозицию состояний всех

- входных кубитов. Это делается при помощи применения соответствующего гейта Адамара, который равен тензорному произведению n унарных гейтов Адамара друг на друга.
2. *Применение итерации Гровера.* Данная итерация состоит из последовательного применения двух гейтов — оракула и так называемого гейта диффузии Гровера (будут подробно рассмотрены ниже). Эта итерация осуществляется $\sqrt{2^n}$ раз.
 3. *Измерение.* После применения итерации Гровера достаточное количество раз необходимо измерить входной регистр кубитов. С очень высокой вероятностью измеренное значение даст указание на искомый параметр. Если необходимо увеличить достоверность ответа, то алгоритм прогоняется несколько раз и вычисляется совокупная вероятность правильного ответа.

Квантовая схема этого алгоритма, само собой разумеется, зависит от размера входных данных, поскольку от этого напрямую зависит количество применений итерации Гровера. В общем виде такая схема может быть изображена при помощи диаграммы, показанной на следующем рисунке.

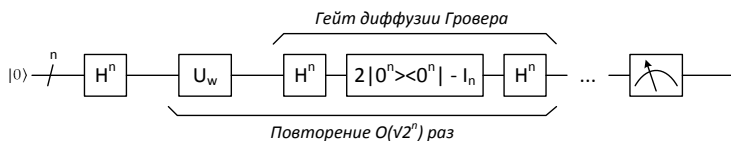


Рис. 23. Общая диаграмма квантовой схемы алгоритма Гровера

Теперь обратим своё внимание на оракул и гейт диффузии, поскольку они являются сердцем алгоритма. Оракул в данном случае должен быть построен не совсем стандартным способом. Если вспомнить процесс преобразования классической бинарной функции в оракул, то можно понять, что принимая

на вход два квантовых реестра (x, y) , он должен выдавать на выходе пару $(x, y \oplus f(x))$. Однако в случае алгоритма Гровера оракул должен инвертировать фазу y того квантового состояния, которое соответствует искомому значению x (напомним, что мы по заданному значению $f(x)$ ищем x , то есть решаем обратную задачу). Это значит, что оракул должен на выходе выдавать значение $(-1)^{f(x)}x$. Фазовый коэффициент $(-1)^{f(x)}$ установлен именно таким, поскольку $f(x) = 1$ тогда и только тогда, когда функция получает на вход искомое значение x , и именно в этом случае фазовый коэффициент становится равен -1 . Другими словами, оракул U_w действует следующим образом:

$$U_w|w\rangle = -|w\rangle$$

$$U_w|x\rangle = |x\rangle, x \neq w$$

Гейт диффузии, в свою очередь, представляет собой соединение трёх гейтов, два из которых стандартны — это многокубитовые гейты Адамара. А вот между ними находится специальный гейт, который, по сути, осуществляет переворот кубитов относительно среднего значения. Его представление в виде аналитической формулы $(2|0^n\rangle\langle 0^n| - I_n)$ несколько затуманивает суть, поскольку его матричное представление просто — в верхнем левом углу матрицы стоит значение 1, в остальных позициях главной диагонали стоит значение -1 , а в остальных местах стоят 0:

$$2|0^n\rangle\langle 0^n| - I_n = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & -1 & & 0 \\ & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & -1 \end{pmatrix}$$

Но того же самого можно достичь, если в вышеприведённой формуле заменить $|0^n\rangle$ на $|+^n\rangle$, и тогда не надо применять гейты Адамара до и после применения этого специального гейта. Так что оператор диффузии Гровера будет выглядеть как:

$$2|+^n\rangle\langle +^n| - I_n$$

Другими словами, с такой формой оператора диффузии квантовая схема алгоритма Гровера становится следующей:

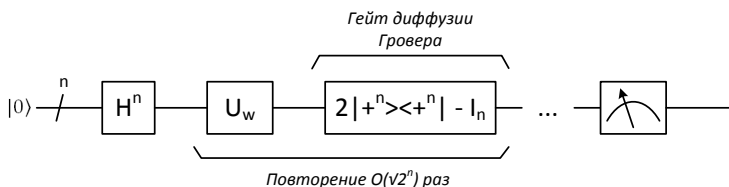


Рис. 24. Общая диаграмма квантовой схемы алгоритма Гровера с новым представлением гейта диффузии

Чтобы не быть голословным, можно рассмотреть этот алгоритм в его реализации на языке программирования. Для этого воспользуемся разработанным в главе 2 фреймворком, поскольку он позволит «заглянуть» в недра алгоритма (чего не позволил бы, например, язык Qirreg, поскольку предоставляет очень высокоуровневые средства генерации квантовых схем).

Но перед написанием кода нам опять надо немного заняться квантовой схемотехникой, поскольку надо спроектировать оракул в виде гейта. Для разнообразия (и укрепления навыков квантовой системотехники) можно рассмотреть функцию от трёх переменных, то есть оракул будет получать на вход 3 кубита (и, естественно, на выходе тоже будет 3 кубита). Рассмотрим такую задачу:

$$f(x_1, x_2, x_3) = x_1 \& x_2 \& x_3$$

Эта функция принимает значение 1 только на одном из восьми вариантов входных значений, что и требуется для алгоритма Гровера. Нет никакой разницы, какую именно

из восьми возможных функций такого вида на трёх переменных рассматривать, но эта функция более проста с технической точки зрения. Как обычно для построения оракула можно воспользоваться таблицей.

Таблица 13. Таблица для построения оракула функции

$$f(x_1, x_2, x_3) = x_1 \& x_2 \& x_3$$

X_1	X_2	X_3	$f(\bar{X})$	$(-1)^{f(\bar{X})}$
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	-1

Этой таблице соответствует следующая матрица 8×8 :

$$\begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & & \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & & 1 & 0 \\ & & & 0 & -1 \end{pmatrix}$$

Поскольку мы сейчас начинаем реализацию более серьезных вещей, чем рассматривались в главе, описывающий фреймворк, нам надо добавить некоторое количество полезных для конструирования оракулов, гейтов и прочего, служебных функций. В частности, в модуль `Gate` необходимо добавить три функции, которые будут использоваться очень часто.

Две из них позволяют создавать комплекснозначные матрицы на основе матриц из целых (перечислимых) чисел. Не секрет, что начальные состояния во многих алгоритмах задаются целыми числами, а оракулы часто состоят из значений 0 и 1, поэтому использование в коде комплексных чисел очень загромождает файл и препятствует пониманию. Поэтому две новых функции:

```
vectorToComplex :: Integral a
                => Vector a -> Vector (Complex Double)
vectorToComplex = map (\i -> fromIntegral i :+ 0.0)

matrixToComplex :: Integral a
                => Matrix a -> Matrix (Complex Double)
matrixToComplex = map vectorToComplex
```

Первая создаёт комплекснозначный вектор, а вторая — матрицу соответственно. А вот следующая функция используется для разбиения заданного списка на подсписки заданной длины. Раньше она была в составе функции для тензорного произведения матриц, а теперь её надо вынести в отдельное определение и засунуть в модуль `Qubit` из-за установленной иерархии модулей, поскольку функция хорошая, пригодится.

```
groups :: Int -> [a] -> [[a]]
groups i s | null s = []
           | otherwise = let (h, t) = splitAt i s
                           in  h : groups i t
```

Теперь перейдём к определению новых гейтов и операторов над ними. В соответствии с описанием алгоритма Гровера, нам потребуется оператор вычитания матриц друг из друга, а также функция для создания гейтов для нескольких кубитов на основе гейта для одного кубита. По крайней мере, эта функция потребуется для тензорного произведения гейтов I и H , чтобы иметь возможность применять их к квантовым регистрам, состоящим из нескольких кубитов.

Для начала определим простой оператор для получения разности матриц:

```
(<->) :: Num a => Matrix a -> Matrix a -> Matrix a
m1 <-> m2 = zipWith (zipWith (-)) m1 m2
```

Ну и для порядка определим такой же оператор для сложения матриц:

```
(<+>) :: Num a => Matrix a -> Matrix a -> Matrix a
m1 <+> m2 = zipWith (zipWith (+)) m1 m2
```

Здесь опять необходимо принять во внимание то, что разработчик сам должен следить за размерностью своих векторов и матриц.

Теперь перейдём к функциям для генерации гейтов. Пока в модуле `Gate` были реализованы функции для представления гейтов, обрабатывающих один или максимум два кубита. Однако при помощи оператора тензорного произведения и функции `entangle` можно создавать функции для представления гейтов, обрабатывающих произвольное количество кубитов. Например, вот так выглядит обобщённая функция, преобразующая заданный однокубитовый гейт в тот же гейт, обрабатывающий заданное количество кубитов:

```
gateN :: Matrix (Complex Double)
      -> Int -> Matrix (Complex Double)
gateN g n = foldl1 (<+>) $ replicate n g
```

Очень просто и изящно. При помощи стандартной функции `replicate` создаётся список из заданного количества однокубитовых гейтов (при этом разработчик сам должен следить за тем, что сюда передаются именно однокубитовые гейты, хотя функция вполне будет работать и с многокубитовыми). Далее этот список сворачивается посредством оператора тензорного произведения (<+>). А вот так эта функция используется:

```
gateIn :: Int -> Matrix (Complex Double)
gateIn = gateN gateI
```

```
gateHn :: Int -> Matrix (Complex Double)
gateHn = gateN gateH
```

Как видно, первая функция формирует тождественное преобразование для заданного количества кубитов, а вторая — преобразование Адамара опять же для заданного количества кубитов. Эти два многокубитовых гейта очень важны в модели квантовых вычислений и часто используются во всевозможных квантовых схемах.

Наконец-то можно обратиться к реализации самого алгоритма Гровера. Начнём с оракула:

```
oracle :: Matrix (Complex Double)
oracle = matrixToComplex [[1, 0, 0, 0, 0, 0, 0, 0],
                           [0, 1, 0, 0, 0, 0, 0, 0],
                           [0, 0, 1, 0, 0, 0, 0, 0],
                           [0, 0, 0, 1, 0, 0, 0, 0],
                           [0, 0, 0, 0, 1, 0, 0, 0],
                           [0, 0, 0, 0, 0, 1, 0, 0],
                           [0, 0, 0, 0, 0, 0, 1, 0],
                           [0, 0, 0, 0, 0, 0, 0, -1]]
```

Тут никакой сложности нет, необходимо просто закодировать матрицу. Конечно, вдумчивый читатель уже давно расширил представленный фреймворк и определил функцию высшего порядка для генерации подобных оракулов (и многих иных). Но мы пока будем действовать по старинке.

Теперь реализуем функцию для представления оператора диффузии Гровера. С учётом тех соображений относительно смены базиса (использовать кубит в квантовом состоянии $|+\rangle$ вместо $|0\rangle$), её определение будет крайне простым:

```
diffusion :: Matrix (Complex Double)
diffusion = 2 <*> (qubitPlus3 |><| qubitPlus3)
               <-> gateIn 3
where
    qubitPlus3 = toVector $
                foldl1 entangle $
```

```
replicate 3 qubitPlus
```

Берём три кубита в состоянии $|+\rangle$ (`qubitPlus`), делаем из них список кубитов и сворачиваем его при помощи функции `entangle`. Далее переводим полученный квантовый регистр в векторное представление. Так получается локальное определение `qubitPlus3`.

Затем этот квантовый регистр `qubitPlus3` векторно умножаем сам на себя, в результате чего получается матрица $|+\rangle\langle+|$, которая далее умножается на 2. Ну и из результата вычитается единичная матрица, подготовленная для трёх кубитов (то есть размера 8×8). При помощи реализованных ранее операторов написание таких функций становится делом очень приятным.

Что же, теперь осталось реализовать сам алгоритм Гровера. Вот определение функции для трёх кубитов:

```
grover :: Matrix (Complex Double) -> IO String
grover f = initial |> gateHn 3
           |> f |> diffusion
           |> f |> diffusion
           >>> (measure . fromVector 3)

where
    initial = toVector $
              foldr entangle qubitZero $
              replicate 2 qubitZero
```

В локальном образце `initial` опять готовится начальное состояние, которое, если присмотреться, равно $|000\rangle$. Далее оно направляется в гейт Адамара для трёх кубитов, в результате чего получается равновероятностная суперпозиция из восьми квантовых состояний, которые могут быть на трёх кубитах. Затем два раза прогоняется цикл Гровера, поскольку $2 \approx \sqrt{2^3}$. После этого осуществляется измерение.

Что будет, если добавить в эту квантовую схему третий цикл Гровера? Всё просто — результаты станут хуже. Читате-

лю предлагается самостоятельно подумать над ответом на вопрос «Почему?».

Поскольку алгоритм Гровера является вероятностным, он выдаёт правильный ответ только с какой-то очень высокой вероятностью. Это значит, что временами при запуске функции `grover` мы будем получать неправильный ответ, поэтому в данном случае предлагается оценить вероятность получения правильного ответа. Реализуем такую функцию:

```
main f n = do l <- replicateM n $ grover f
           return $
             map (length &&& head) $ group $ sort l
```

Она применяет алгоритм Гровера для заданного оракула заданное количество раз, а результатом её работы является гистограмма результатов алгоритма (то есть список пар вида (частота появления, результат)). Далее эта функция была запущена миллион раз, в результате чего был построен вот такой график:



Рис. 25. Гистограмма частот получения результатов в алгоритме Гровера для трёх кубитов

Правильный результат был получен примерно в 94.5 % случаев, а остальные результаты имеют частоту примерно

в 0.78 %. Этого вполне достаточно для того, чтобы иметь возможность запустить алгоритм Гровера три раза и выбрать из этих трёх запусков результат, повторившийся по крайней мере дважды.

Вдумчивый читатель к этому времени уже должен был задаться вопросом, а что если оракул будет возвращать фазу -1 на нескольких входных данных, а не на одном? В этом случае алгоритм Гровера тоже работает, как это ни странно (хотя ничего странного в этом как раз и нет, если рассмотреть последовательность умножений матриц). Однако для нахождения одного правильного ответа из множества требуется намного меньше итераций.

Пусть l — количество значений входных параметров, на которых функция принимает значение 1 (то есть оракул возвращает фазу -1). Тогда для нахождения одного правильного значения с большой вероятностью требуется $\sqrt{\frac{2^n}{l}}$ итераций Гровера. Это можно продемонстрировать следующим кодом:

```
oracle' :: Matrix (Complex Double)
oracle' = matrixToComplex [[1, 0, 0, 0, 0, 0, 0, 0],
                           [0, -1, 0, 0, 0, 0, 0, 0],
                           [0, 0, 1, 0, 0, 0, 0, 0],
                           [0, 0, 0, 1, 0, 0, 0, 0],
                           [0, 0, 0, 0, -1, 0, 0, 0],
                           [0, 0, 0, 0, 0, 1, 0, 0],
                           [0, 0, 0, 0, 0, 0, 1, 0],
                           [0, 0, 0, 0, 0, 0, 0, -1]]
```

Если запустить функцию `main` с этим оракулом и дать возможность выполнить построение гистограммы на миллионе запусках, то будет явлена примерно следующая диаграмма:



Рис. 26. Гистограмма частот получения результатов в алгоритме Гровера для трёх кубитов с тремя правильными ответами

Что это? Правильные ответы получили наименьшую частоту? А всё правильно, поскольку сейчас функция `grover`, которая вызывается из функции `main`, выполняет две итерации Гровера, а для оракула с тремя правильными ответами необходима одна итерация. Как только выполнено больше итераций, чем требуется по алгоритму, ситуация переворачивается с ног на голову (поскольку у нас происходит переворот относительно среднего). Но в данном конкретном случае одной итерации так же недостаточно, поскольку частотные вероятности правильных и неправильных ответов будут достаточно близки друг к другу (это резонно, поскольку была сделана только одна итерация).

В общем, как показывает этот пример, разработчик при использовании алгоритма Гровера всегда должен внимательно следить за количеством итераций и не допускать переворота ситуации в противоположную сторону.

На этом описание алгоритма Гровера можно завершить. Теперь читателю вполне должно быть понятно, как он работает, какова математическая подоснова, и как можно реализовать этот алгоритм на языке программирования.



Grover.hs

*К книге прилагается файл
с исходным кодом на языке
Haskell с описанной здесь
реализацией алгоритма.*

Алгоритмы Дойча и Дойча-Йожи

Мы уже рассмотрели алгоритм Дойча, когда решали простейшие задачи по квантовым вычислениям после описания разработанного фреймворка (вернее даже, уже дважды рассмотрели этот алгоритм). Так что здесь мы рассмотрим его расширение в виде алгоритма Дойча-Йожи. Но вначале краткая историческая справка и некоторые математические описания.

Дэвид Дойч разработал самый первый алгоритм в рамках модели квантовых вычислений в далёком 1985 году. Честно говоря, это был небольшой прорыв, поскольку этот алгоритм впервые показал превосходство модели квантовых вычислений над классической моделью. Далее он стал примером для исследователей, и в результате появились следующие прорывные алгоритмы (в первую очередь алгоритм факторизации Шора).

Сама по себе задача Дойча не очень показательна. Но он заметил, что при помощи модели квантовых вычислений можно одновременно вычислить значения некоторой двоичной функции на всём множестве входных значений. Поскольку входные кубиты можно перевести в равновероятностную суперпозицию всех базисных состояний, а потом применить заданную функцию, то как раз в этой случае проявится принцип *квантового параллелизма*. Однако что дальше? Достать все полученные значения функции невозможно, поскольку в результате измере-

ния с той или иной вероятностью получается одно значение из всего множества. Но здесь есть один момент. Поскольку до измерения в результирующем состоянии квантового регистра скрыто всё множество значений функции, при должных манипуляциях этим квантовым регистром можно получить информацию о некоторых общих свойствах функции. То есть, ещё раз, мы получаем не какие-то конкретные значения функции, но информацию о её общих свойствах.

И вот на свет появляется задача Дойча, в которой надо понять, является ли двоичная функция на одном двоичном же аргументе константной или сбалансированной. Свойство того, что функция является константной или сбалансированной — это как раз пример такого общего свойства функции. Так что принцип квантового параллелизма вполне можно применить для его изучения. Что и сделал Д. Дойч.

В 1992 году Ричард Йожа предложил расширить задачу Д. Дойча и рассмотреть функцию $f: \{0,1\}^n \rightarrow \{0,1\}$, то есть не ограничиваться 4-мя функциями, рассмотренными нами ранее. Проблема лишь в том, что количество различных таких функций равно 2^{2^n} , и среди этого количества большую часть составляют функции, которые не являются ни константными, ни сбалансированными. Поэтому задачу сформулировали несколько искусственным образом, а именно ограничили возможные виды функции, и она может быть либо константной, либо сбалансированной.

В этом случае алгоритм абсолютно тот же самый. Вот словесное описание последовательности его шагов:

1. Инициализировать начальное состояние из n кубитов, которое должно быть $|0^n\rangle$.
2. Применить к начальному состоянию гейт Адамара для n кубитов $H^{\otimes n}$, в результате чего получается равнове-

роятностная суперпозиция всех возможных значений n кубитов.

3. Применить оракул O_f , который строится несколько иначе, чем было рассмотрено ранее.
4. Снова применить гейт Адамара $H^{\otimes n}$.
5. Произвести измерение. Если в результате измерения будет получено значение $|0^n\rangle$, то функция константна. В противном случае она сбалансирована (при этом значение в результате измерения может быть использовано для получения общего понимания того, на каких значениях функция возвращает 1).

Вот диаграмма квантовой схемы описанного алгоритма:

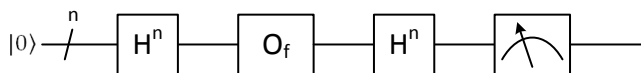


Рис. 27. Диаграмма квантовой схемы алгоритма Дойча-Йожи

Оракул O_f меняет фазу на -1 у тех квантовых состояний, для которых функция f возвращает значение 1. Здесь нет особого смысла использовать служебный кубит, поскольку матрица, у которой на главной диагонали стоят только 1 и -1 , а в остальных позициях стоят 0, унитарна.

Далее с математической точки зрения происходит следующее. Начальное квантовое состояние переходит в равновероятностную суперпозицию. Затем у каждого квантового состояния в этой суперпозиции меняется знак фазы, если функция принимает на этом квантовом состоянии значение 1). Потом, после второго применения гейта Адамара, все квантовые состояния «схлопываются». И если функция является константной, то происходит деструктивная интерференция фаз всех квантовых состояний, кроме $|0^n\rangle$, а у этого квантового со-

стояния наоборот происходит конструктивная интерференция, и его амплитуда становится равна 1.

Проще всего понять эти выкладки на примерах, поэтому реализуем этот алгоритм при помощи разработанного ранее фреймворка. Этот алгоритм очень интересен для проведения различных экспериментов и замеров частотных вероятностей, поэтому далее приводится некоторое расширение фреймворка и описывается один эксперимент, который будет небезынтересным заинтересованному читателю.

Для начала определим функцию для создания оракулов. То, что мы раньше вручную определяли оракулы в виде матриц, это хорошо, поскольку позволяет отточить навыки в квантовой схемотехнике. Но пришла пора сделать нормальную функцию для построения оракула для алгоритма Дойча-Йожи при использовании произвольной функции $f: \{0,1\}^n \rightarrow \{0,1\}$. Так что вот определение:

```
makeOracle :: ([Bool] -> Bool) -> Int
            -> Matrix (Complex Double)
makeOracle f n = matrixToComplex $
    map (uncurry makeLine) $
    zip domain [1..2^n]

where
    zeroes = replicate (2^n) 0
    domain = replicateM n [False, True]

    makeLine :: [Bool] -> Int -> [Int]
    makeLine x i = changeElement zeroes i
                  ((-1)^(if f x then 1 else 0))
```

Функция `makeOracle` принимает на вход функцию `f` с типом `[Bool] -> Bool`, а также количество кубитов `n`. Далее она строит всю область определения функции (`domain`), представляющую из себя список всех возможных комбинаций значений `False` и `True` длиной `n`. Область определения (длина списка со-

ставляет 2^n) сшивается в пары со списком индексов от 1 до 2^n . Это необходимо для построения матрицы.

Матрицу оракула строим при помощи локально определённой функции `makeLine`, которая строит одну строку матрицы. На вход она получает элемент области определения функции и номер строки матрицы. Далее она заменяет в списке из нулей длины 2^n элемент, стоящий на позиции, равной номеру строки матрицы, на 1 или -1 в зависимости от значения функции.

После этого матрица преобразуется в комплекснозначную при помощи уже известной функции `matrixToComplex` и возвращается в качестве результата.

Здесь есть вызов функции `changeElement`, которая меняет элемент в списке на заданной позиции. Её определение очень просто:

```
changeElement :: [a] -> Int -> a -> [a]
changeElement xs i x = take (i - 1) xs ++
                        [x] ++
                        drop i xs
```

Размерность списка и номер элемента не проверяются на адекватность, поэтому разработчику необходимо самостоятельно следить за этим делом. Ну это дело обычное для разрабатываемого фреймворка.

Теперь можно применить функцию `makeOracle` для построения оракула. Но в целях экспериментирования сделаем девять оракулов:

```
oracle :: Int -> [Bool] -> Bool
oracle 1 [_ , _ , _] = False
oracle 2 [x , y , z] = x && y && z
oracle 3 [x , y , _] = x && y
oracle 4 [x , y , z] = x && (z || y && not z)
oracle 5 [x , _ , _] = x
oracle 6 [x , y , z] = y && z || x && (not y || y && not z)
```

```

oracle 7 [x, y, z] = y || oracle 6 [x, y, z]
oracle 8 [x, y, z] = x || y || z
oracle 9 [_ , _ , _] = True

```

Первый аргумент функции `oracle` принимает на вход номер оракула. Частичное применение этой функции с номером даёт функцию, которую ожидает на вход функция `makeOracle`. Такой подход позволяет использовать списки номеров для массового вызова функции, что будет использовано далее.

Все девять оракулов определены так, как показывает следующая таблица:

Таблица 14. Таблица функций для проведения эксперимента с алгоритмом Дойча-Йожи

X_1	X_2	X_3	f								
			1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	1	1
0	1	0	0	0	0	0	0	0	1	1	1
0	1	1	0	0	0	0	0	1	1	1	1
1	0	0	0	0	0	0	1	1	1	1	1
1	0	1	0	0	0	1	1	1	1	1	1
1	1	0	0	0	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1	1

Ясно, что для номеров оракула 1 и 9 алгоритм должен вернуть $|000\rangle$, а вот что он вернёт для остальных функций? Давайте разберёмся...

Вот определение функции, реализующей сам алгоритм Дойча-Йожи. Оно очень несложное:


```

jozsza :: Matrix (Complex Double) -> Int -> IO String
jozsza f n = initial |> gateHn n
              |> f
              |> gateHn n
              >>> (measure . fromVector n)

where
    initial = toVector $
              foldl1 entangle $
              replicate n qubitZero

```

Всё по описанию: готовим начальное состояние (уже известным нам по алгоритму Гровера образом), применяем гейт Адамара, далее оракул, и снова гейт Адамара. После этого производим измерение. Как видно, определение этой функции общее, и она принимает на вход оракул для функции $f: \{0,1\}^n \rightarrow \{0,1\}$, надо просто указать значение n .

Для эксперимента нам опять потребуется строить гистограммы, причём уже для девяти разных вариантов, поэтому напомним уж для этого специальную функцию. Вот она:

```

histogram :: (Monad m, Ord a)
           => m a -> Int -> m [(Int, a)]
histogram qs n = do l <- replicateM n qs
                    return $
                      map (length &&& head) $
                      group $
                      sort l

```

Её код практически полностью дублирует ранее описанный код в алгоритме Гровера, поэтому сразу перейдём к определению функции `main`:

```

main :: Int -> IO [(Int, String)]
main n = mapM (\i -> histogram
              (jozsza
               (makeOracle
                (oracle i) 3) 3) n)
          [1..9]

```

Вот тут и используется тот метод, использованный в определении функции `oracle`. Мы просто берём список

[1...9] и при помощи функции `map` применяем к каждому его элементу процесс построения оракула, его прогонки через алгоритм Дойча-Йожи и построения гистограммы. В результате запуска этой функции (скажем, для каждого оракула прогоним алгоритм один миллион раз), получается занимательная таблица:

Таблица 15. Результаты эксперимента с алгоритмом Дойча-Йожи

X_1	X_2	X_3	f								
			1	2	3	4	5	6	7	8	9
0	0	0	1 000 000	561 592	249 916	62 405		62 504	250 329	562 946	1 000 000
0	0	1		69 964		62 373		62 706		62 096	
0	1	0		62 666	249 794	62 694		62 515	249 748	62 240	
0	1	1		62 108		62 098		62 525		62 560	
1	0	0		63 144	250 264	562 886	1 000 000	562 153	250 030	62 595	
1	0	1		62 304		62 517		62 357		62 763	
1	1	0		62 799	250 026	62 505		62 386	249 893	62 628	
1	1	1		62 423		62 522		62 854		62 172	

Результаты очень показательны. Они абсолютно подтверждают то, что для константных функций результат должен быть $|000\rangle$. Но почему для сбалансированной функции № 5 результатом стало $|100\rangle$, а не, например, $|001\rangle$? Единица в первом кубите показывает, что функция сбалансирована именно по первому кубиту (вспомним её определение). Если бы мы использовали определение:

`oracle 5 [_ , _ , z] = z`

то в результате 100 % измерений дали бы именно тат $|001\rangle$. А если бы функция не была сбалансирована по какой-либо переменной, то её результат был бы вероятностным.

Далее смотрим на функции № 2 и 8. Они *почти* константны. Именно поэтому с более 55 % вероятностью на выходе получается результат $|000\rangle$. То же самое можно сказать о функциях № 4 и 6 — они *почти* сбалансированы, поэтому опять с более чем 55 % вероятности в результате выходит уже понятное нам значение $|100\rangle$.

Что касается функций № 3 и 5, то они одинаково далеко находятся от константных и сбалансированных, поэтому результаты примерно равновероятностно распределены между четырьмя возможными значениями. Эти значения опять намекают на то, на каких именно входных аргументах функция принимает значение 1.

Заинтересованному читателю рекомендуется дальше поэкспериментировать с разработанным алгоритмом, чтобы закрепить у себя понимание того, как он работает. Ну а после этого можно перейти к рассмотрению следующего алгоритма.



Jozsa.hs

*К книге прилагается файл
с исходным кодом на языке
Haskell с описанной здесь
реализацией алгоритма.*

Алгоритм Саймона

В 1994 году Даниэль Саймон опубликовал статью «О мощи квантовых вычислений», в которой описал алгоритм, позже названный его именем, имеющий экспоненциальное превосходство над имеющимися алгоритмами в рамках классической вычислительной модели. Задача, которую решает этот алго-

ритм, является не такой оторванной от реальности, как задача Дойча, хотя и ещё несколько абстрактной. Тем не менее, эта задача и представленный алгоритм вызвали огромный интерес у исследователей, и в дальнейшем алгоритм Саймона стал основой ряда квантовых алгоритмов, в том числе и для алгоритмов Шора факторизации целых чисел и вычисления дискретного логарифма.

Задача Саймона звучит следующим образом. Пусть дана двоичная функция $f: \{0,1\}^n \rightarrow \{0,1\}^m$, при этом $m \geq n$. Кроме того известно, что данная функция является либо взаимно однозначной, либо имеется некоторое число $s \in \{0,1\}^n$, называемое *периодом*, что для любой пары различных входных значений x и x' функция f возвращает одинаковое значение тогда и только тогда, когда $x' = x \oplus s$. Для заданной функции, удовлетворяющей этим условиям, необходимо найти период s .

Другими словами, алгоритм Саймона возвращает следующее значение:

- s , если существует нетривиальная (не равная 0^n) строка s такая, что $\forall x' \neq x : f(x') = f(x) \Leftrightarrow x' = x \oplus s$.
- 0 , если функция f взаимно однозначная.
- Неопределено, в противном случае.

Алгоритм Саймона вычисляет период функции s за линейное время (и линейное количество вызовов оракула), в то время как любому классическому алгоритму требуется экспоненциальное время в зависимости от длины входного аргумента функции.

Для получения периода s с большой вероятностью (алгоритм Саймона, как полагается, является вероятностным алгоритмом) необходимо повторить полиномиальное от длины входного слова число раз следующую процедуру:

1. Применить к входному регистру $|0^n\rangle$ преобразование Адамара для n кубитов.
2. Применить к полному набору кубитов оракул C_f .
3. Вновь применить только к входному регистру преобразование Адамара. После этого входной регистр необходимо измерить для получения значения $(x' \oplus s)$.

Применив эту последовательность шагов не более $(n - 1)$ раз, можно доподлинно восстановить значение периода s .

Диаграмма квантовой схемы описанного алгоритма выглядит следующим образом:

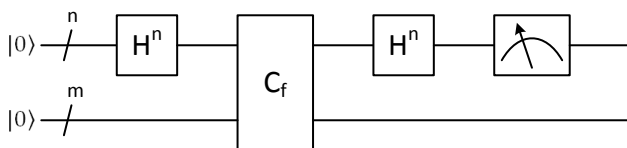


Рис. 28. Диаграмма квантовой схемы алгоритма Саймона

Давайте, как это принято, попробуем реализовать пример такой квантовой схемы при помощи ранее разработанного фреймворка для квантовых вычислений, дорабатывая его по мере необходимости. Для этого, как полагается, выберем тестовую функцию и запроектируем оракул для неё.

В качестве функции предлагается рассмотреть такую: $f: \{0,1\}^2 \rightarrow \{0,1\}^2$, поскольку в этом случае входов и выходов будет по 4, а это значит, что размеры матриц будут 16×16 . Похоже, что это предел для выполнения упражнений на бумаге, поэтому остановимся на таком варианте. Однако читателя никто не ограничивает, и в качестве упражнений для тренировки можно рассмотреть и функции более высоких размерностей.

Пусть дана некоторая функция, которая определена следующим образом:

```
targetF :: (Bool, Bool) -> (Bool, Bool)
targetF (False, False) = (False, True)
targetF (False, True)  = (False, True)
targetF (True, False)  = (True, False)
targetF (True, True)   = (True, False)
```

Здесь использован некаррированный вариант передачи входных аргументов, и это сделано исключительно для единообразия кода. Как видно, эта функция представляет собой описание следующей таблицы истинности:

Таблица 16. Таблица истинности функции `targetF`

X_1	X_2	$f(X_1, X_2)$	
0	0	0	1
0	1	0	1
1	0	1	0
1	1	1	0

Если внимательно присмотреться, то станет понятно, что периодом s в данном случае является строка $(0,1)$, поскольку значение функции одинаково для любой пары (x_1, x_2) и $(x_1 \oplus 0, x_2 \oplus 1) = (x_1, \bar{x}_2)$. Такая функция подходит под задачу Саймона, а потому может быть использована для исследования реализации алгоритма.

Для построения оракула, как это обычно бывает, воспользуемся табличной техникой. Вот таблица, которая показывает то, как оракул должен преобразовывать входные кубиты.

Таблица 17. Таблица для проектирования оракула
для алгоритма Саймона

X_1	X_2	Y_1	Y_2	$f(X_1, X_2)$		$f(X_1, X_2) \oplus (Y_1, Y_2)$		Преобразование
0	0	0	0	0	1	0	1	$ 0000\rangle \rightarrow 0001\rangle$
0	0	0	1	0	1	0	0	$ 0001\rangle \rightarrow 0000\rangle$
0	0	1	0	0	1	1	1	$ 0010\rangle \rightarrow 0011\rangle$
0	0	1	1	0	1	1	0	$ 0011\rangle \rightarrow 0010\rangle$
0	1	0	0	0	1	0	1	$ 0100\rangle \rightarrow 0101\rangle$
0	1	0	1	0	1	0	0	$ 0101\rangle \rightarrow 0100\rangle$
0	1	1	0	0	1	1	1	$ 0110\rangle \rightarrow 0111\rangle$
0	1	1	1	0	1	1	0	$ 0111\rangle \rightarrow 0110\rangle$
1	0	0	0	1	0	1	0	$ 1000\rangle \rightarrow 1010\rangle$
1	0	0	1	1	0	1	1	$ 1001\rangle \rightarrow 1011\rangle$
1	0	1	0	1	0	0	0	$ 1010\rangle \rightarrow 1000\rangle$
1	0	1	1	1	0	0	1	$ 1011\rangle \rightarrow 1001\rangle$
1	1	0	0	1	0	1	0	$ 1100\rangle \rightarrow 1110\rangle$
1	1	0	1	1	0	1	1	$ 1101\rangle \rightarrow 1111\rangle$
1	1	1	0	1	0	0	0	$ 1110\rangle \rightarrow 1100\rangle$
1	1	1	1	1	0	0	1	$ 1111\rangle \rightarrow 1101\rangle$

Этой таблице соответствует следующая матрица (сразу пишем её на языке Haskell, чтобы не тратить время и место):

```
oracle :: Matrix (Complex Double)
oracle = matrixToComplex
  [[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
   [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
   [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
   [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
```

```

[0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0]

```

Теперь реализуем непосредственно функцию, которая кодирует представленную ранее квантовую схему алгоритма Саймона для случая $f: \{0,1\}^2 \rightarrow \{0,1\}^2$. Вот очень несложное определение:

```

simon :: Matrix (Complex Double) -> IO String
simon o = initial |> gateH2I2
          |> o
          |> gateH2I2
          >>> (measure . fromVector 4)

where
  initial = toVector $
            foldr entangle qubitZero $
            replicate 3 qubitZero
  gateH2I2 = gateHn 2 <+> gateIn 2

```

Как видно, здесь опять представлена техника, которая позволяет писать код, очень похожий на сами квантовые схемы. При помощи операции ($|>$) собираются последовательности гейтов, через которые пропускается квантовый регистр кубитов.

В данном случае в локальном определении `initial` создаётся квантовое состояние $|0000\rangle$, которое затем пропускается через гейт `gateH2I2`, тоже определённый локально. Этот гейт представляет собой преобразование Адамара, применённое к первым двум кубитам четырёхкубитного регистра, а вторые

два кубита остаются без изменений (к ним применяется гейт I , не изменяющий квантового состояния).

Далее всё делается в соответствии с алгоритмом и его квантовой схемой.

Поскольку алгоритм является вероятностным, имеет смысл реализовать функцию `main` для многократного запуска функции `simon` и сбора результатов её исполнения. Определим её следующим образом:

```
main :: Int -> IO [(Int, String)]
main n = do l <- replicateM n $ simon oracle
          return $
            map (length &&& head) $
              group $
                sort l
```

Если запустить эту функцию с достаточно большим параметром (я запускал со значением 1 000 000), то на выходе будут достаточно достоверные значения частотных вероятностей измерения того или иного результата. В данном конкретном примере в качестве результатов будут такие значения квантовых состояний: $|0001\rangle$, $|0010\rangle$, $|1001\rangle$ и $|1010\rangle$, и все они будут иметь вероятность получения примерно 25 % (а в теории так ровно 0.25). Однако, честно говоря, в соответствии с буквой алгоритма надо измерять только входной регистр, то есть в данном случае первые два кубита. Это значит, что вероятности квантовых состояний $|0001\rangle$ и $|0010\rangle$ надо просто сложить, чтобы получить вероятность получения квантового состояния $|00\rangle$ для первых двух кубитов. Другими словами, в этом примере в результате измерения входного регистра будут равновероятно получаться значения $|00\rangle$ и $|10\rangle$.

Результат $|00\rangle$ является «тривиальным» и нас не интересует (поскольку какое бы значение не складывать по модулю 2 со значением 0, будет получаться исходное значение) — нулевое значение всегда является «тривиальным периодом»

для любой двоичной функции. Так что остаётся только другое нетривиальное значение, и оно равно $|10\rangle$, как и было загадано.

Для закрепления пройденного к настоящему моменту материала вдумчивому читателю рекомендуется несколько несложных упражнений по этой теме:

1. Реализация функции `makeOracle` для автоматического создания оракулов произвольной размерности.
2. Исследование алгоритма для функции, у которой есть только нулевой период (которая является перестановкой).
3. Исследование алгоритма для функций, у которых есть больше одного периода (а можно ли построить такую функцию?).

После выполнения этих задач можно перейти к следующему, наиболее интересным в данном контексте разделам этой главы.



Simon.hs

*К книге прилагается файл
с исходным кодом на языке
Haskell с описанной здесь
реализацией алгоритма.*

Алгоритм Шора

Теперь мы можем перейти к рассмотрению алгоритма, который наделал больше всего шума, и, собственно, из-за которого, по мнению многих, новая вычислительная модель, основанная на законах квантовой механики, получила такое развитие. Это алгоритм Шора для факторизации целых чисел, являющихся произведением двух простых нечётных чисел.

Всё дело в том, что именно на гипотезе алгоритмической сложности задачи факторизации числа основаны многочисленные современные алгоритмы и системы криптографии. Найденный Питером Шором в 1994 году алгоритм позволяет решить эту задачу за полиномиальное время (стало быть, поли-

номиально количество гейтов) и на полиномиальном количестве кубитов, в то время как лучшие классические алгоритмы решают её за суперполиномиальное (субэкспоненциальное) время. А это значит, что как только квантовый компьютер с достаточным количеством кубитов будет создан, вся современная криптография окажется под угрозой компрометации. Собственно, она и будет скомпрометирована тут же, поскольку любая информация, сокрытая с использованием этого подхода, может быть получена любым лицом, у кого имеется доступ к такому квантовому компьютеру.

Поскольку алгоритм Шора разительно отличается от всех ранее рассмотренных алгоритмов в части наличия серьёзной прикладной значимости, сам алгоритм более сложен с точки зрения математики и архитектуры. Здесь уже задействованы две вычислительные парадигмы — классическая часть готовит входные данные для алгоритма Шора, а также управляет циклами и возвратами в целях нахождения правильного результата; ну а квантовая часть, как это обычно бывает, просто исполняет линейную последовательность унитарных преобразований над специально подготовленными состояниями входных кубитов.

Итак, алгоритм факторизации Шора... Его суть заключается в сведении задачи факторизации к задаче поиска периода функции. Если известен период функции, то факторизация осуществляется при помощи алгоритма Евклида за полиномиальное время на классическом компьютере. И вот квантовая часть алгоритма факторизации как раз занимается поиском периода функции. А классическая часть алгоритма сначала специальным образом готовит оную функцию, а потом проверяет найденный квантовой частью период на достаточность для решения задачи. Если период найден правильно (алгоритм вероятностный, так что может найти не то, что хочется), то задача решена. Если же нет, то квантовая часть

алгоритма прогоняется ещё раз. А, поскольку, проверка правильности решения для задачи факторизации очень проста (умножили два числа да сравнили с третьим), то эту часть алгоритма вообще можно не принимать во внимание с точки зрения подсчёта сложности.

Чтобы не быть голословными и слишком теоретизированными, давайте попробуем рассмотреть алгоритм факторизации Шора на простом примере. Для этого просто разложим на простые множители некоторое число, являющееся произведением двух и в точности двух простых чисел, ни одно из которых не является числом 2. Поскольку компания IBM уже факторизовала на своём прототипе квантового компьютера число 15, здесь предлагается факторизовать число 21, хотя это вообще не принципиально, поскольку главное в рассмотрении то, чтобы хватило вычислительных мощностей.

Сначала рассмотрим алгоритм вручную, а потом, как это полагается, напомним программу на языке Haskell с использованием всё того же ранее разработанного фреймворка.

Для того чтобы факторизовать число при помощи алгоритма Шора, необходимо найти период периодической функции $f(x) = a^x \pmod{M}$, где a — некоторый параметр, выбираемый произвольно до старта алгоритма, а M — число, которое необходимо факторизовать. Самое главное ограничение заключается в том, чтобы у чисел a и M не было общих делителей, больших 1. Поскольку мы хотим факторизовать число 21, в качестве параметра a возьмём число 2. Обычно это значение и берут (чаще всего оно подходит), но иногда необходимо брать другое значение, и критерии выбора будут описаны позже.

Теперь составим такую таблицу...

Таблица 18. Расчёт периода для факторизации числа 21

x		0	1	2	3	4	5	6	7
a^x	2^x	1	2	4	8	16	32	64	128
$a^x \bmod M$	$2^x \bmod 21$	1	2	4	8	16	11	1	2

Из этой таблицы видно, что в данном конкретном примере периодом функции является значение $r = 6$. Другими словами, чтобы найти период при помощи такой таблицы, необходимо найти минимальное значение x , большее 1, для которого $a^x \equiv 1 \pmod{M}$. Соответственно, алгоритм Шора выигрывает именно на этом этапе, поскольку при помощи модели квантовых вычислений найти период функции можно с полиномиальной сложностью, чего нельзя сделать в рамках классической вычислительной модели.

Далее остаётся дело техники. Множители числа M определяются как наибольшие общие делители (НОД) между $a^{\frac{r}{2}} \pm 1$ и M . Отсюда следует ещё одно ограничение на значение параметра a , выбираемого перед запуском алгоритма, — найденный период должен быть чётным, то есть нацело делиться на 2. Поскольку в рассматриваемом примере $r = 6$, данное условие выполнено. Так что в качестве $a^{\frac{r}{2}} \pm 1$ получаются два числа — $2^3 - 1 = 7$ и $2^3 + 1 = 9$. Вычислить наибольшие общие делители можно с помощью уже упоминавшегося алгоритма Евклида, и для чисел 7 и 9 с числом 21 это будет пара (7, 3). Соответственно, эта пара и есть разложение числа 21 на простые множители.

Теперь рассмотрим структурированное и более или менее формализованное описание алгоритма факторизации. Сам алгоритм состоит из следующих шагов:

1. Выбрать случайное число a , меньшее M : $a < M$.

2. Вычислить $\text{НОД}(a, M)$. Это может быть сделано при помощи алгоритма Евклида.
3. Если $\text{НОД}(a, M)$ не равен 1, то существует нетривиальный делитель числа M , так что алгоритм завершается (вырожденный случай).
4. В противном случае необходимо использовать *квантовую подпрограмму* поиска периода функции $f(x) = a^x \bmod M$.
5. Если найденный период r является нечётным, то вернуться на шаг 1 и выбрать другое число a .
6. Если $a^{\frac{r}{2}} \equiv M - 1 \pmod{M}$, то вернуться на шаг 1 и выбрать другое число a .
7. Наконец, определить два значения $\text{НОД}(a^{\frac{r}{2}} \pm 1, M)$, которые и являются нетривиальными делителями числа M .

Собственно, здесь все шаги, кроме 4-го, представляют собой тривиальные действия, которые мы рассматривать не будем. Имеет смысл рассмотреть только четвёртый шаг, на котором выполняется квантовый алгоритм в виде подпрограммы полного алгоритма факторизации Шора.

Здесь важным моментом является выбор количества кубитов, которые будут использованы в квантовой схеме. В соответствии с рекомендациями автора алгоритма необходимо выбрать такое количество кубитов q , при котором выполняется неравенство $M^2 \leq 2^q < 2M^2$. Выполнение этого неравенства обозначает, что данного количества кубитов будет достаточно для того, чтобы выполнить функцию, для которой ищется период, достаточное количество раз, чтобы сработала *конструктивная интерференция*. В рассматриваемом примере, когда $M = 21$, достаточным числом q будет, очевидно, число 9:

$$21^2 = 441 \leq 2^9 = 512 < 882$$

Итого, требуется 9 кубитов во входном квантовом регистре и столько же кубитов во вспомогательном квантовом регистре,

который используется для получения значения квантового оракула, в который необходимо преобразовать функцию. Стало быть, всего необходимо 18 кубитов, а это значит, что для факторизации выбранного числа требуется матрица размера $2^{18} \times 2^{18}$, то есть 262144×262144 . Такой размер гейта не очень способствует его построению в ручном режиме, поэтому здесь мы воспользуемся некоторым упрощением, которое скажется только на уровне точности получаемых результатов (но не на принципе). Мы будем использовать всего 9 кубитов, причём 4 из них будут рабочими, а 5 — вспомогательными. Выбранное число вспомогательных кубитов вполне достаточно для представления всех значений рассматриваемой периодической функции, поскольку максимальным её значением является 16, для представления которого требуется как раз 5 битов.

Для построения оракула на этом количестве кубитов можно воспользоваться электронными таблицами. Для этого надо просто составить такую же таблицу, которая составлялась при построении оракулов во всех предыдущих примерах, только теперь в ней будет уже 512 строк. Эти строки можно заполнить в автоматизированном режиме при помощи инструментов, предлагаемым оболочкой для обработки электронных таблиц. Но всё равно это будет очень долго и муторно.

Тем не менее, посмотрим, как эта таблица будет выглядеть для выбранного количества кубитов — у нас используется 4 входных и 5 вспомогательных кубитов. То, как будет выглядеть заголовок такой таблицы, а также начальные и конечные строки, показано в следующей таблице.

**Таблица 19. Таблица для проектирования оракула
для алгоритма факторизации Шора**

X_1	X_2	X_3	X_4	Y_1	Y_2	Y_3	Y_4	Y_5	$f(\bar{X})$					$f(\bar{X}) \oplus \bar{Y}$					
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	1
...																			
1	1	1	1	1	1	1	1	0	1	0	0	0	0	0	0	1	1	1	0
1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	1	1	1	1

Как видно, здесь нет ничего необычного. Оракул строится стандартным образом, то есть абсолютно так же, как и для всех прочих алгоритмов (если только для какого-то алгоритма не указана иная техника построения оракула).

Теперь эту таблицу необходимо преобразовать в матрицу. Для этого можно воспользоваться упрощённым методом, применив для него небольшого рода автоматизацию. Поскольку каждая строка в этой таблице соответствует строке в матрице оракула (точь в точь вместе с номерами строк), то необходимо лишь узнать, каким столбцам матрицы оракула соответствуют строки. Для этого надо просто взять и для каждой строки посчитать десятичное число, получаемое из двоичного, которое, в свою очередь, составляется из первых четырёх и последних пяти столбцов указанной таблицы. Сделать это можно всё теми же средствами электронной таблицы. В итоге получится список из 512 целых чисел, в котором позиция числа определяет номер строки в матрице оракула, а само число определяет номер столбца в этой же матрице. Остаётся написать функцию для преобразования такого списка в саму матрицу. Это не слишком сложно:

```
oracle :: Matrix (Complex Double)
```



```
oracle = matrixToComplex $
    map (qubitFromInt 9) oracleList

qubitFromInt :: Integral a => a -> Int -> [a]
qubitFromInt b n = changeElement (replicate (2^b) 0)
    (n + 1) 1
```

Здесь список `oracleList` представляет собой тот самый список из 512 элементов. Мы не будем приводить его здесь, разве что только его начало:

```
oracleList = [1, 0, 3, 2, 5, 4, 7, 6, 9, 8, 11, 10, 13,
```

Функция `qubitFromInt` строит векторное представление кубита, заданного десятичным числом. Она использует ранее определённую функцию `changeElement`, которая меняет элемент в заданном списке на заданной позиции. Поскольку последняя функция нумерует элементы в списке, начиная с 1, а здесь имеет место нумерация с 0, то приходится к номеру числа в списке прибавлять единицу. И в итоге эта функция выдаёт список из 511 нулей и 1 единицы, которая стоит на заданном месте.

Но иногда нам надо будет строить оракулы и для большего количества кубитов, так что таким подходом не обойтись. Придётся автоматизировать дальше и написать функцию для вычисления списка, который используется при построении оракула. Это сделать очень несложно:

```
makeOracleList :: Int -> Int -> (Int -> Int) -> [Int]
makeOracleList m n f =
    [x * 2^n + (y `xor` f x) | x <- [0..2^m - 1],
                                y <- [0..2^n - 1]]
```

Если эту функцию запустить со следующими параметрами:

```
> makeOracleList 4 5 (\x -> 2^x `mod` 21)
```

то в результате получится абсолютно такой же список, как и `oracleList`. И эта функция будет использоваться далее

для построения полноценного оракула на 9 кубитах. Это делается при помощи переопределения функции `oracle`:

```
oracle :: Matrix (Complex Double)
oracle = matrixToComplex $
  map (qubitFromInt 18) $
    makeOracleList 9 9 (\x -> 2^x `mod` 21)
```

Нужно отметить, что функция `makeOracleList` позволяет строить оракулы для произвольных функций типа `(Int -> Int)`, так что её резонно переместить в модуль `Gate`.

Теперь можно реализовать «сердце» этого алгоритма, а именно квантовое преобразование Фурье. Как ни странно, его проще всего реализовать на основе аналитической формулы, при помощи которой вычисляются элементы матрицы, представляющей гейт. Для напомнимания, вот эта формула:

$$A(m, n) = \exp(-2\pi i \frac{(m-1)(n-1)}{N})$$

Здесь m и n — номера позиции элемента по строкам и столбцам, при этом счёт начинается с 1. При помощи формулы Эйлера ($e^{ix} = \cos x + i \sin x$) гейт для квантового преобразования Фурье кодируется следующим образом:

```
qft :: Int -> Matrix (Complex Double)
qft q = [[euler m n | m <- [1..2^q]] | n <- [1..2^q]]
  where
    euler m n = let power = (-2) * pi * (m - 1) * (n - 1)
                  / 2^q
                in cos power :+ sin power
```

Городить что-то на основе квантовых схем в данном случае было бы просто неразумно. А так получается простейшая функция, которая ещё и строит гейт для заданного количества кубитов (параметр `q`).

Теперь перед реализацией квантовой процедуры поиска периода надо определить несколько вспомогательных функций

и констант (и далее по тексту такие константы будут определены ещё в большем количестве):

```
numberToFactor :: Int
numberToFactor = 21

simpleNumber :: Int
simpleNumber = 2

nofAncillas :: Int
nofAncillas = 5

nofWorkingQubits :: Int
nofWorkingQubits = 4

nofQubits :: Int
nofQubits = nofWorkingQubits + nofAncillas

periodicFunction :: Int -> Int
periodicFunction x = simpleNumber^x `mod` numberToFactor
```

Константа `numberToFactor` задаёт число, которое мы будем разлагать на простые множители. Константа `simpleNumber` представляет собой взаимно простое число с предыдущей константой. Далее константы `nofAncillas`, `nofWorkingQubits` и `nofQubits` представляют количество вспомогательных и рабочих кубитов, а также общее количество кубитов в квантовой схеме. Такой подход к параметризации алгоритма выбран для того, чтобы не определять функции с большим количеством аргументов, но при этом оставалась бы возможность быстро менять параметры алгоритма в одном месте.

Функция `periodicFunction` являет собой как раз ту периодическую функцию, задачу поиска периода которой и выполняет квантовая подпрограмма алгоритма Шора. Как видно, здесь во всю используются определённые ранее константы.

Теперь можно реализовать и квантовую схему в виде функции. Она будет выглядеть вот так:

```
quantumFactoring :: Matrix (Complex Double) -> IO String
quantumFactoring o =
    initial |> gateHn nofWorkingQubits <+>
        gateIn nofAncillas
    |> o
    |> qft nofWorkingQubits <+>
        gateIn nofAncillas
    >>> (measure . fromVector nofQubits)
```

Это определение соответствует следующей диаграмме квантовой схемы:

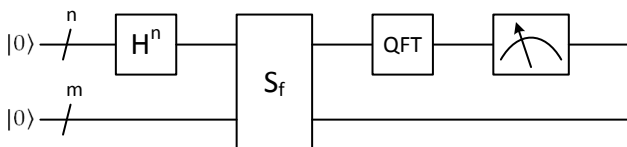


Рис. 29. Диаграмма квантовой схемы алгоритма Шора для факторизации целых чисел

Надеюсь, что к этому времени читатель уже настолько поднаторел в понимании модели квантовых вычислений, что может сначала прочитать код функции, а затем восстановить по нему диаграмму квантовой схемы. Это очень полезный навык, так что всем читающим эти строки рекомендуется в нём потренироваться.

Проблемой (или, скорее, особенностью) алгоритма Шора является то, что результат измерения, полученный после выполнения функции `quantumFactoring` представляет собой некоторое грубое приближение к отношению некоторого числа к периоду функции. Получить из этого значение самого периода можно только посредством запуска приведённой квантовой схемы некоторого количества раз, сбора результатов с дальнейшей манипуляцией ими специальным образом.

К настоящему времени разработано несколько различных методов вычисления периода функции на основании списка собранных результатов, причём разные методы обладают разной сложностью и требуют различное количество кубитов для своей работы. Самым простым (но чуть более трудоёмким) способом получения периода является *метод цепных дробей*.

Цепной дробью называется представление вещественного числа в следующем виде:

$$a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{a_3 + \dots}}}$$

где a_0 — целое, а остальные коэффициенты натуральные. Рациональные числа могут быть представлены конечной цепной дробью, а иррациональные — бесконечной.

При помощи цепных дробей можно найти хорошие приближения к искомому периоду периодической функции, при этом размер этого приближения ограничен некоторым количеством битов (равным количеству рабочих кубитов). Для выполнения этих громоздких вычислений имеет смысл написать специальные функции и другие определения. Начнём с типа для представления цепных дробей:

```
newtype Chain = Chain
    {
        unChain :: [Integer]
    }
deriving (Eq, Show)
```

Цепная дробь — это просто список её коэффициентов. Мы заворачиваем список в новый конструктор изоморфного типа **newtype** в целях инкапсуляции поведения.

Теперь определим функции преобразования рационального числа в цепную дробь и обратно. Это довольно просто

в полном соответствии с математическими правилами (разработанными ещё Эйлером):

```
toChain :: Rational -> Chain
toChain = Chain . unfoldr step
  where
    step r@(a :% b) | a == 0      = Nothing
                   | b < a      = Nothing
                   | otherwise = Just (m, b % a - m % 1)
                      where m = b `div` a

fromChain :: Chain -> Rational
fromChain = foldr ((\a b -> rev $ b + a) .
                  (% 1)) 0 .
                  unChain
  where rev (a :% b) = b :% a
```

В дополнение к этим функциям необходимо определить ещё и функцию для преобразования числа из цепной дроби в обычную так, что числитель и знаменатель ограничены определённым количеством бит. Эта функция как раз и потребуется для приближённого вычисления периода. Вот её определение:

```
fromChainLimited :: Int -> Chain -> Rational
fromChainLimited n = last .
    takeWhile ((< 2^n) . denominator) .
    map (fromChain . Chain) .
    inits .
    unChain
```

Имея такие функции, несложно написать определение для получения периода на основании измеренного значения входного реестра кубитов. Вот оно:

```
getPeriod :: Integer -> Integer
getPeriod s = denominator $
    fromChainLimited nofWorkingQubits $
    toChain $
    s % 2^nofWorkingQubits
```

Поскольку эта функция довольно важна с точки зрения алгоритма факторизации, её необходимо рассмотреть подробнее.

На вход ей подаётся число s , полученное после измерения входного реестра. Далее строится дробь, в числителе которой стоит это число, а в знаменателе — 2 в степени равной количеству входных кубитов. Эта дробь далее преобразуется в цепную, а затем обратно в обычную, но с ограничением по количеству бит для её представления. После этого берётся знаменатель, который и возвращается в качестве результата. В этом и состоит суть метода цепных дробей в применении к алгоритму факторизации Шора.

В общем-то, всё готово для реализации полного цикла алгоритма. Но остаётся не совсем ясным вопрос о том, сколько раз запускать квантовую подпрограмму, а сколько раз сам алгоритм факторизации, чтобы вероятность получения правильного ответа была большая, но при этом тратилось бы не так уж много временных ресурсов. Для получения ответов на эти вопросы можно провести небольшой, но интересный эксперимент.

Для эксперимента напомним несколько функций. Первая функция возвращает частотную вероятность получения правильного ответа в зависимости от количества запусков квантовой подпрограммы и самого квантового алгоритма. Вот её определение:

```
investigate :: Int -> Int -> IO Int
investigate n m = liftM sum $ replicateM 100 try
  where
    try = do rs <- collectPeriods n m
            let fs = filter (\(f1, f2) -> f1 /= 1 &&
                                         f2 /= 1) $
                        map getFactors rs
            case fs of
              ((f1, f2):_) -> return 1
              []           -> return 0
```

Эта функция 100 раз запускает полный цикл квантового поиска простых делителей, в котором сам алгоритм запускается n раз, а квантовая подпрограмма в рамках каждого запуска

алгоритма запускается m раз. За это количество запусков считается количество успешного разложения числа — берётся просто сумма элементов списка, в котором значение 1 обозначает, что простые делители найдены, и 0 — то, что не найдены. Так что результатом является примерное процентное соотношение корректной отработки алгоритма.

Здесь есть вызов функции `collectPeriods`, которая собирает большинство возможных периодов периодической функции для числа, которое факторизацией при помощи алгоритма Шора. Вот её определение:

```
collectPeriods :: Int -> Int -> IO [Integer]
collectPeriods n m =
  do l <- replicateM n $ findRandomPeriod m
  return $
    filter (\r -> simpleNumber^(r `div` 2) `mod`
            numberToFactor /= numberToFactor - 1) $
    filter even $
    map head $
    group $
    sort l
```

Опять же эта функция n раз запускает следующий процесс (получается, что сам алгоритм факторизации): производится вызов квантовой подпрограммы m раз, результаты её работы собираются в список, который сортируется и группируется, дубликаты удаляются, а над полученным списком производится фильтрация в соответствии с критериями получения хорошего периода, описанными в математическом определении алгоритма Шора. Сначала удаляются нечётные периоды, а затем такие, которые дают тривиальные делители (для них $a^{\frac{r}{2}} \equiv M - 1 \pmod{M}$). Результирующий список подходящих периодов и возвращается.

Как видно, здесь использована функция `findRandomPeriod`, которая запускает квантовую подпрограмму заданное количество раз. Рассмотрим её определение:


```

findRandomPeriod :: Int -> IO Integer
findRandomPeriod n =
  do l <- replicateM n $ quantumFactoring oracle
  return $
    foldr lcm 1 $
      map getPeriod $
        filter (/= 0) $
          map snd $
            filter (\(i, _) -> i >= n `div` 10) $
              map (length &&& (binToInt . head)) $
                group $
                  sort $
                    map (take nofWorkingQubits) l
  where
    binToInt s = sum $
      zipWith (*) (map (2^) [0..])
        (map (\c -> if c == '0'
                      then 0
                      else 1) $
          reverse s)

```

Данное определение довольно громоздко, поэтому требуются пояснения. Функция запускает m раз квантовую подпрограмму, которую мы уже определили (`quantumFactoring`). Далее собранные результаты работы этой подпрограммы собираются в список, который подвергается следующим манипуляциям. Сначала вычлняются результаты измерения только входных кубитов, поскольку вспомогательные нас не интересуют. Далее этот список сортируется, группируется, из него удаляются дубликаты. Заодно при удалении дубликатов считается количество удалённых дубликатов (то есть, получается, что это опять частотная вероятность проявления такого результата в рамках всех измерений), а само значение переводится из строки в число. Далее список фильтруется так, что из него удаляются все результаты, частотная вероятность проявления которых меньше, чем десятикратно уменьшенное число запусков. Таким отфильтрованные результаты являются помехами и в алгоритме не рассматриваются. Далее отфильтровываются

нулевые результаты, поскольку нулевой результат тривиален и ведёт к получению тривиальных делителей. После этого при помощи вызова функции `getPeriod` получаются все возможные периоды по результатам выполнения квантовой подпрограммы. И затем (и это очень важно) вычисляется *наименьшее общее кратное* всех таких периодов, которое и возвращается в качестве результата.

Вуаля! Этот набор функций позволяет провести исследование и узнать частотную вероятность получения корректного результата. Но для массового запуска функции `investigate` желательно немного подготовиться. Для этого напомним ещё одно определение:

```
runInvestigation :: [Int] -> [Int] -> IO ()
runInvestigation ns ms =
  mapM_ showIt [(n, m), investigate n m] |
    n <- ns,
    m <- ms]
where
  showIt ((x, y), n) = do
    n' <- n
    putStrLn ("(" ++ show x ++ ", " ++ show y ++
      "): " ++ show n')
```

Эта функция просто запускает двойной цикл запуска функции `investigate` в заданных интервалах сбора информации, а результаты выводит на экран в удобочитаемом виде.

Данная функция была запущена на интервалах `[1..10]` для количества вызова алгоритма и `[1..50]` для количества вызова квантовой подпрограммы. В результате построен вот такой примечательный график:

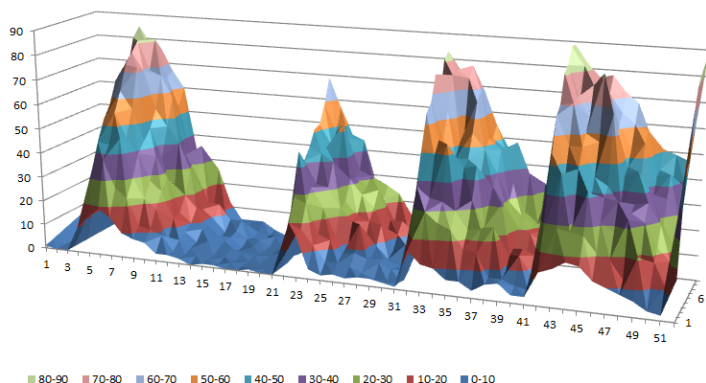


Рис. 30. График распределения частотной вероятности получения корректного результата при помощи алгоритма Шора в зависимости от количества вызовов алгоритма (короткая ось) и количества выполнения квантовой подпрограммы (длинная ось)

Этот график представляет довольно красивый с эстетической точки зрения результат. И при помощи него же можно вполне определить, что для корректного получения результата с вероятностью, приближающейся к 90 %, достаточно запустить алгоритм 10 раз, в каждом запуске использовать квантовую подпрограмму 3 раза. Поэтому так и определим:

```
nofShorAttempts :: Int
nofShorAttempts = 10
```

```
nofQFCalling :: Int
nofQFCalling = 3
```

Теперь можно написать определение функции `main`, которая будет факторизовать число `numberToFactor`. Сделаем её дружелюбной к пользователю:

```
main :: IO ()
```

```

main =
  do putStrLn ("Факторизуем число: " ++
               show numberToFactor)
     putStrLn ("Используем для этого число: " ++
               show simpleNumber)
     putStrLn ("Запускаем квантовый алгоритм " ++
               "факторизации " ++
               show nofShorAttempts ++ " раз")
     rs <- collectPeriods nofShorAttempts nofQFCalling
     let fs = filter (\(f1, f2) -> f1 /= 1 && f2 /= 1) $
               map getFactors rs
     case fs of
       ((f1, f2):_) ->
         putStrLn ("Ура, разложение найдено: " ++
                   show numberToFactor ++ " = " ++
                   show f1 ++ " * " ++ show f2)
       [] ->
         putStrLn ("Увы, квантовый алгоритм Шора " ++
                   "потерпел фиаско. Попробуйте " ++
                   "запустить его ещё раз.")

```

В результате выполнения этой функции на экран консоли будет выведено что-то типа:

```

> main
Факторизуем число: 21
Используем для этого число: 2
Запускаем квантовый алгоритм факторизации 10 раз
Ура, разложение найдено: 21 = 7 * 3

```

Собственно, это всё. Но для закрепления материала можно реализовать тот же самый алгоритм, но без квантовой части. Поиск периода будет осуществляться простым перебором списка. Вот определение функции:

```

classicFactoring :: Int -> [(Int, Int)]
classicFactoring m =
  map ((gcd m *** gcd m) . makePrerequisites) $
    filter testProperPeriods $
      map findProperPeriods simpleNumbers
  where
    simpleNumbers :: [Int]
    simpleNumbers = [a | a <- [2..m - 1], gcd m a == 1]

```

```

findProperPeriods :: Int -> (Int, Int)
findProperPeriods a = (a, (+ 1) $
    length $
    takeWhile (/= 1) $
    map (\x -> a^x `mod` m) [1..])

testProperPeriods :: (Int, Int) -> Bool
testProperPeriods (a, r) = even r &&
    a^(r `div` 2) `mod` m /= m - 1

makePrerequisites :: (Int, Int) -> (Int, Int)
makePrerequisites (a, r) = (subtract 1 *** (+ 1)) $
    double $ a^(r `div` 2)

double :: a -> (a, a)
double x = (x, x)

```

В числе материалов, прилагаемых к данной книге, имеется файл с электронной таблицей, в которой также рассматривается алгоритм факторизации, а также приводятся результаты экспериментирования, описанные выше. Ну а заинтересованному читателю рекомендуется изменить значение константы `numberToFactor` на 15 (это ещё одно число, которое *можно* разложить на простые множители при помощи алгоритма Шора на имеющихся количествах кубитов) и посмотреть, что получится.



Shor.hs

*К книге прилагается файл
с исходным кодом на языке
Haskell с описанной здесь
реализацией алгоритма.*

Краткие выводы

Таким образом, квантовые алгоритмы не так страшны, как могут показаться на первый взгляд при чтении литературы по теме квантовых вычислений, где оные алгоритмы расписаны исключительно в математических терминах. Если переложить

их на какой-либо язык программирования, то алгоритмы становятся вполне понятными и обозримыми.

При таком подходе к изучению квантовых алгоритмов необходимо помнить несколько моментов:

1. Размерности используемых в вычислениях векторов (квантовых регистров) и матриц (гейтов) растут экспоненциально в зависимости от количества задействованных в алгоритме кубитов, поэтому изучать «на бумаге» алгоритмы с количеством кубитов больше 4 очень затруднительно, а на классическом компьютере при помощи эмуляции квантового вычислительного устройства возможно рассматривать работу квантовых алгоритмов над регистрами размером не более 10 — 15 кубитов.
2. Эмуляция алгоритмов возможна на классических компьютерах, но при эмуляции не будет обнаружено никакой выгоды по сравнению с классической вычислительной моделью. Более того, для некоторых алгоритмов (например, для алгоритма факторизации Шора) будет налицо явный проигрыш, поскольку для факторизации очень маленьких чисел потребуется перемножать матрицы гигантских размеров, в то время как простейшие переборные алгоритмы решат эту же задачу для маленьких чисел в мгновение ока. Выгода наступает только на квантовом вычислительном устройстве (которого ещё нет) и на больших выходных данных.
3. Когда наконец-то появится полноценный универсальный квантовый компьютер, то для работы на нём, вероятно, придётся изучать какой-нибудь язык программирования, типа языка Qirreg. Это значит, что синтаксис придётся изучать если не с нуля, то в любом случае это будет несколько иное, нежели представленный в этой книге фреймворк. Однако семантика модели квантовых вычислений, показанная здесь, останется с большой долей вероятности именно такой.

Всё вышеперечисленное можно свести к простому тезису: если некий разработчик программного обеспечения уже сегодня хочет подготовиться к появлению в реальности универсального квантового компьютера, то ему следует начинать изучать как модель квантовых вычислений (её принципы и особенности), так и разработанные к настоящему моменту квантовые алгоритмы.

Изучение квантовых алгоритмов лучше всего (по опыту автора) проводить в три этапа. Первым этапом в голову загружается математическое описание алгоритма, в котором необходимо разобраться самым доскональным образом. На втором этапе необходимо реализовать данный алгоритм на своём любимом языке программирования. Например, можно сделать фреймворк для осуществления квантовых вычислений, как это было сделано в этой книге. Затем при помощи фреймворка реализуются все интересующие исследователя алгоритмы (само собой разумеется, запустить эти алгоритмы получится только на входных данных небольшого размера). Будет лучше всего, если исследователь поставит и проведёт какие-нибудь эксперименты с реализованными алгоритмами, чтобы наилучшим образом проникнуть в суть оных алгоритмов.

Наконец, на третьем этапе разработчику программного обеспечения рекомендуется погрузиться в какой-либо из уже существующих языков квантового программирования (Qirppre или QCL), и уже при помощи этого языка реализовать интересующие алгоритмы. Заодно можно повторить все поставленные ранее эксперименты.

Вдумчивому читателю, например, после ознакомления с этой главой, уже должно быть ясно, как можно использовать алгоритм Гровера для квадратичного ускорения произвольных задач. Например, при помощи алгоритма Гровера можно реализовать ту же самую факторизацию, и этот алгоритм будет рабо-

тять быстрее, чем классический переборный алгоритм (но медленнее специализированного алгоритма Шора). Так что можно, к примеру, попытаться решить такую задачу: факторизовать заданное число при помощи алгоритма Гровера.

Я буду крайне признателен читателям, которые пришлют мне на адрес электронной почты roman.dushkin@gmail.com результаты своих исследований в этой области.

Глава 5.

Квантовый «зоопарк»

Если ученый не может объяснить, чем он занимается, уборщице, моющей пол в его лаборатории, значит, он сам не понимает, чем он занимается.

Эрнест Резерфорд

Мы детально рассмотрели несколько интересных алгоритмов, с которых, собственно, модель квантовых вычислений получила свою популярность. Так выходит, что именно эти алгоритмы всегда описываются в источниках информации по квантовым вычислениям, предназначенным для массового читателя (хотя, конечно же, такой читатель должен быть подготовлен). Лишь изредка в отдельных книгах и учебниках встречаются описания других алгоритмов.

Здесь мне хотелось бы отступить от указанной традиции и привести описания чуть более пятидесяти квантовых алгоритмов, которые разработаны к настоящему моменту. Во-первых, это покажет, что с момента рождения модели квантовых вычислений человеческая мысль не стоит на месте, и теоретические разработки в этом направлении ведутся (к печали некоторых критиков, которые ошибочно пишут, что с 90-ых годов прошлого века не разработано ни одного нового квантового алгоритма, который показывал бы преимущество по сравнению с классическими вычислениями). Во-вторых, перечень алгоритмов покажет заинтересованному читателю, что есть множество мест приложения модели квантовых вычислений, а также множество направлений, куда можно двигаться дальше после прочтения этой книги.

Данная глава основана на списке «Квантовый зоопарк», который ведёт Стивен Джордан из Национального Института Стандартов и Технологий США. Конечно, список этот постоянно обновляется, и эта глава в конечном счёте станет несколько неактуальной. Однако на момент написания книги здесь собраны все приведённые в списке алгоритмы, для каждого из которых дано краткое описание и список первоисточников. Думаю, что русскому читателю будет интересно ознакомиться с этим списком на родном языке, а не переводить для себя каждый раз одно и то же.

В этой главе будут приведены описания алгоритмов, разделённые на три больших раздела: алгебраические и теоретико-числовые алгоритмы, алгоритмы со специальными оракулами и алгоритмы аппроксимации и эмуляции. В каждом разделе описывается несколько алгоритмов так, как они описаны их авторами, то есть зачастую в терминах отдельных направлений математики (обычно, теории групп и т. д.). Для хорошего понимания описания с точки зрения теории сложности алгоритмов читателю необходимо владеть O -нотацией (включая все

дополнительные обозначения: O , Ω , Θ и др.), а для понимания конкретных алгоритмов надо иметь знания в той области, в рамках которой алгоритм разработан.

Надеюсь, что изучение информации, представленной в этой главе, позволит читателю ещё глубже погрузиться в модель квантовых вычислений и подойти уже к тому уровню, когда в голове начинают зарождаться зачатки мыслей о собственных алгоритмах. Если у кого-либо из читающих эти строки произойдёт внезапное озарение, и на свет явится новый квантовый алгоритм, я буду считать задачу этой книги перевыполненной стократно. Ну а ежели по результатам выпуска издания появится клуб любителей квантовых вычислений, которые с задором и технологично будут штамповать квантовые схемы для решения разнообразных задач, то и в данном случае задача книги будет перевыполнена многократно.

В этой главе в том же самом формате, что и все остальные алгоритмы, будут описаны и те из них, что мы уже изучили в предыдущей главе. Это сделано ради единообразия представления информации.

В конце главы все описанные алгоритмы будут выстроены в более или менее стройную систему, из представления которой станет вполне понятно текущее состояние изучаемого направления исследования.

Алгебраические и теоретико-числовые алгоритмы

Факторизация

Ускорение: сверхполиномиальное.

Описание: Для заданного n -битного целого числа необходимо найти его разложение на простые множители. Квантовый

алгоритм Питера Шора решает эту задачу за время $\text{poly}(n)$. Наиболее быстрому из известных классических алгоритмов требуется сверхполиномиальное время от n для решения этой задачи. Алгоритм Шора компрометирует систему криптографии RSA. Ядром алгоритма является задача поиска периода, которая может быть сведена к задаче поиска скрытой абелевой подгруппы.

Данный алгоритм детально описан в главе 4.

- Shor P. W. *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*. SIAM Journal on Computing, 26(5):1484-1509, 1997. [arXiv:quant-ph/9508027](https://arxiv.org/abs/quant-ph/9508027).
- Shor P. W. *Algorithms for Quantum Computation: Discrete Logarithms and Factoring*. In Proceedings of FOCS 1994, pg. 124-134.

Дискретный логарифм

Ускорение: сверхполиномиальное.

Описание: Даны три n -битных числа a , b и N , для которых известно, что $b = a^s \bmod N$ для некоторого числа s . Задачей является найти число s . Как показано П. Шором, эта задача может быть решена при помощи квантового компьютера за время $\text{poly}(n)$. Наиболее быстрому из известных классических алгоритмов требуется сверхполиномиальное время от n для решения этой задачи. Посредством похожей техники, которая описана в основополагающей работе П. Шора. Квантовый компьютер также может решить задачу поиска дискретного логарифма на эллиптических кривых, таким образом компрометируя криптографию, основанную на эллиптических кривых. Сверхполиномиальное квантовое ускорение для этой задачи также подтверждено для полугрупп. См. также задачу поиска скрытой абелевой подгруппы.

- Shor P. W. *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*. SIAM Journal on Computing, 26(5):1484-1509, 1997. [arXiv:quant-ph/9508027](https://arxiv.org/abs/quant-ph/9508027).
- Proos J., Zalka C. *Shor's discrete logarithm quantum algorithm for elliptic curves*. Quantum Information and Computation, Vol. 3, No. 4, pg. 317-344, 2003. [arXiv:quant-ph/0301141](https://arxiv.org/abs/quant-ph/0301141).
- Childs A., Ivanyos G. *Quantum computation of discrete logarithms in semigroups*. 2013. [arXiv:1310.6238](https://arxiv.org/abs/1310.6238).
- Banin M., Tsaban B. *A reduction of semigroup DLP to classic DLP*. 2013. [arXiv:1310.7903](https://arxiv.org/abs/1310.7903).

Уравнение Пелля

Ускорение: сверхполиномиальное.

Описание: Для заданного положительного целого числа d , не являющегося квадратом, уравнением Пелля является $x^2 - dy^2 = 1$. Для любого такого d имеется бесконечное множество пар (x, y) , решающих это уравнение. Пусть (x_1, y_1) — пара, которая минимизирует выражение $x + y\sqrt{d}$. Если число d требует для представления n бит (то есть $0 \leq d < 2^n$), то запись пары (x_1, y_1) может потребовать экспоненциально большое количество битов. Поэтому в общем случае невозможно найти пару (x_1, y_1) за полиномиальное время.

Пусть $R = \log(x_1 + y_1\sqrt{d})$. В этом случае $[R]$ уникальным образом идентифицирует пару (x_1, y_1) . Как показал Ш. Холгрэн, для заданного n -битового числа d квантовый компьютер может найти $[R]$ за время $\text{poly}(n)$. Неизвестно классических алгоритмов, решающих эту задачу за полиномиальное время. Факторизация сводится к этой задаче. Этот алгоритм компрометирует систему криптографии Бухмана-Вильямса. См. также задачу поиска скрытой абелевой подгруппы.

- Hallgren S. *Polynomial-time quantum algorithms for Pell's equation and the principal ideal problem*. In Proceedings of the 34th ACM Symposium on Theory of Computing, 2002.

Главный идеал

Ускорение: сверхполиномиальное.

Описание: Даны n -битное целое число d и обратимый идеал I кольца $\mathbb{Z}[\sqrt{d}]$. Идеал I является главным, если существует $\alpha \in \mathbb{Q}(\sqrt{d})$, такое, что $I = \alpha\mathbb{Z}[\sqrt{d}]$. Значение α может экспоненциально зависеть от числа d . Однако значение $[\log \alpha]$ уникальным образом идентифицирует α . Задачей является определение, является ли I главным идеалом, и если является, то необходимо вычислить $[\log \alpha]$. Как показано Ш. Холгреном, эта задача может быть решена за полиномиальное время на квантовом компьютере. Также разработан квантовый алгоритм, решающий эту же задачу на меньшем числе кубитов (см. вторую ссылку). Задача факторизации сводится к решению уравнения Пелля, которое, в свою очередь, сводится к задаче поиска главного идеала. Поэтому задача поиска идеала является, по крайней мере, такой же сложной, как и факторизация, а потому, по видимому, не лежит в классе сложности P. См. также задачу поиска скрытой абелевой подгруппы.

- Hallgren S. *Polynomial-time quantum algorithms for Pell's equation and the principal ideal problem*. In Proceedings of the 34th ACM Symposium on Theory of Computing, 2002.
- Schmidt A. *Quantum Algorithms for many-to-one Functions to Solve the Regulator and the Principal Ideal Problem*. 2009. [arXiv:0912.4807](https://arxiv.org/abs/0912.4807).

Единичная группа

Ускорение: сверхполиномиальное.

Описание: Числовое поле $\mathbb{Q}(\theta)$ имеет степень d , если наименьшая степень полинома, корнем которого является θ , есть d . Множество \mathcal{O} элементов поля $\mathbb{Q}(\theta)$, которые являются корнями монарных полиномов $\mathbb{Z}[x]$, образует кольцо, называемое кольцом целых в $\mathbb{Q}(\theta)$. Множество единиц (обратимых элементов) кольца \mathcal{O} образуют группу, обозначаемую \mathcal{O}^* . Как показано Ш. Холгреном и независимо от него А. Шмидтом и У. Воллмером для произвольного поля $\mathbb{Q}(\theta)$ с фиксированной степенью за полиномиальное время на квантовом компьютере можно найти множество генераторов для \mathcal{O}^* на основании заданного описания θ . Для этой задачи неизвестны классические алгоритмы, работающие за полиномиальное время. Ш. Холгрэн со товарищи последовательно обнаружили, как достичь полиномиального масштабирования в зависимости от степени. Алгоритм основан на решении задачи о поиске скрытой абелевой подгруппы над аддитивной группой действительных чисел.

- Hallgren S. *Fast quantum algorithms for computing the unit group and class group of a number field*. In Proceedings of the 37th ACM Symposium on Theory of Computing. 2005.
- Schmidt A., Vollmer U. *Polynomial time quantum algorithm for the computation of the unit group of a number field*. In Proceedings of the 37th Symposium on the Theory of Computing, pg. 475-480. 2005.
- Eisenträger K., Hallgren S., Kitaev A., Song F. *A quantum algorithm for computing the unit group of an arbitrary degree number field*. STOC. 2014.

Группа классов

Ускорение: сверхполиномиальное.

Описание: Числовое поле $\mathbb{Q}(\theta)$ имеет степень d , если наименьшая степень полинома, корнем которого является θ , есть d . Множество \mathcal{O} элементов поля $\mathbb{Q}(\theta)$, которые являются

корнями монарных полиномов $\mathbb{Z}[x]$, образует кольцо, называемое кольцом целых в $\mathbb{Q}(\theta)$. Для кольца идеалы по модулю простых идеалов формируют группу, называемую группой классов. Как показано Холгреном квантовый компьютер по заданному θ может за полиномиальное время найти множество генераторов группы классов целочисленного кольца для поля с произвольной константной степенью d . Для решения этой задачи неизвестно классического алгоритма, работающего за полиномиальное время. См. также задачу поиска скрытой абелевой подгруппы.

- Hallgren S. *Fast quantum algorithms for computing the unit group and class group of a number field*. In Proceedings of the 37th ACM Symposium on Theory of Computing. 2005.

Суммы Гаусса

Ускорение: сверхполиномиальное.

Описание: Пусть \mathbb{F}_q есть конечное поле. Элементы поля \mathbb{F}_q , отличные от нуля, формируют группу \mathbb{F}_q^\times с умножением, а сами элементы поля \mathbb{F}_q формируют (абелеву, но не обязательно циклическую) группу \mathbb{F}_q^+ со сложением. Мы можем выбрать некоторую характеристику χ^\times из \mathbb{F}_q^\times и характеристику χ^+ из \mathbb{F}_q^+ . Соответствующая сумма Гаусса является внутренним произведением этих характеристик: $\sum_{x \neq 0 \in \mathbb{F}_q} \chi^+(x) \chi^\times(x)$. Как показано В. ван Дамом и Г. Серуси, на квантовом компьютере сумма Гаусса может быть оценена с полиномиальной точностью за полиномиальное время. Несмотря на то, что конечное кольцо не формирует группу с умножением, множество его единиц формирует такую группу. Выбирая представление для аддитивной группы кольца и выбирая представление для мультипликативной группы его единиц, можно получить сумму Гаусса над единицами конечного кольца. Этот результат также может быть получен на квантовом компьютере

с полиномиальной точностью за полиномиальное время. При этом неизвестен классический алгоритм с такими характеристиками. Алгоритм поиска дискретного логарифма сводится к алгоритму оценки суммы Гаусса. Некоторые статистические суммы модели Поттса могут быть вычислены за полиномиальное время на квантовом компьютере при помощи данного алгоритма.

- Dam van W., Seroussi G. *Efficient quantum algorithms for estimating Gauss sums*. 2002. [arXiv:quant-ph/0207131](https://arxiv.org/abs/quant-ph/0207131).
- Geraci J., Lidar D. A. *On the exact evaluation of certain instances of the Potts partition function by quantum computers*. Comm. Math. Phys. Vol. 279, pg. 735. 2008. [arXiv:quant-ph/0703023](https://arxiv.org/abs/quant-ph/0703023).

Экспоненциальное сравнение

Ускорение: полиномиальное.

Описание: Пусть заданы числа $a, b, c, f, g \in \mathbb{F}_q$. Необходимо найти такие $x, y \in \mathbb{F}_q$, что $af^x + bg^y = c$. Как показано в работе В. ван Дама и И. Шпарлинского на квантовом компьютере эту задачу можно решить за время $\tilde{O}(q^{3/8})$, в то время как лучший алгоритм на классическом компьютере решает её за время $\tilde{O}(q^{9/8})$. Этот квантовый алгоритм основан на вычислении дискретного алгоритма и квантовом поиске.

- Dam van W., Shparlinski I. *Classical and quantum algorithms for exponential congruences*. Proceedings of TQC 2008, pg. 1-10. [arXiv:0804.1109](https://arxiv.org/abs/0804.1109).

Элементы матриц для представлений групп

Ускорение: сверхполиномиальное.

Описание: Все представления конечных групп и компактных линейных групп могут быть выражены в виде

унитарных матриц при наличии выбранного подходящим образом базиса. Сопряжение регулярного представления группы при помощи схемы квантового преобразования Фурье над этой группой даёт прямую сумму неприводимых представлений группы. Поэтому эффективное квантовое преобразование Фурье над симметричной группой совместно с тестом Адамара предоставляет быстрый квантовый алгоритм для аддитивной аппроксимации отдельных элементов матрицы произвольных неприводимых представлений S_n . Таким же образом при использовании квантового преобразования Шура можно эффективно аппроксимировать элементы матрицы неприводимых представлений $SU(n)$, которые имеют полиномиальный вес.

В литературе также описаны прямые реализации при помощи эффективных квантовых схем отдельных неприводимых представлений для групп $U(n)$, $SU(n)$, $SO(n)$ и A_n . Экземпляры, которые кажутся экспоненциально сложными для классических алгоритмов, описаны там же.

- Beals R. *Quantum computation of Fourier transforms over symmetric groups*. In Proceedings of STOC/ 1997, pg. 48-53.
- Bacon D., Chuang I. L., Harrow A. W. *The quantum Schur transform: I. efficient qudit circuits*. In Proceedings of SODA. 2007, pg. 1235-1244. [arXiv:quant-ph/0601001](https://arxiv.org/abs/quant-ph/0601001).
- Jordan S. P. *Fast quantum algorithms for approximating the irreducible representations of groups*. 2008. [arXiv:0811.0562](https://arxiv.org/abs/0811.0562).

Проверка произведения матриц

Ускорение: полиномиальное.

Описание: Для заданных матриц A , B и C размера $n \times n$ проверка произведения отвечает на вопрос, является ли верным равенство $AB = C$. Лучший классический алгоритм из известных

отвечает на этот вопрос за время $O(n^2)$, в то время как лучший классический алгоритм для произведения матриц использует время $O(n^{2.373})$. А. Амбайнис с коллегами разработали квантовый алгоритм, который решает эту задачу за время $O(n^{7/4})$. После этого Г. Бурман и К. Шпалек улучшили алгоритм до показателя $(n^{5/3})$. Последний алгоритм основан на результатах использования квантового блуждания.

- Ambainis A., Buhrman H., Høyer P., Karpimizki M., Kurur P. *Quantum matrix verification*. Unpublished Manuscript, 2002.
- Buhrman H., Špalek R. *Quantum verification of matrix products*. In Proceedings of the 17th ACM-SIAM Symposium on Discrete Algorithms, pages 880-889, 2006. [arXiv:quant-ph/0409035](https://arxiv.org/abs/quant-ph/0409035).
- Szegedy M. *Quantum speed-up of Markov chain based algorithms*. In Proceedings of the 45th IEEE Symposium on Foundations of Computer Science, pg. 32, 2004.

Сумма подмножеств

Ускорение: полиномиальное.

Описание: Для заданного списка целых чисел x_1, \dots, x_n и целевого целого числа s необходимо определить, существует ли в списке некоторый подсписок, сумма элементов которого в точности равна s . Эта задача является NP-полной, поэтому вряд ли существует алгоритм для классического или квантового компьютера, который решал бы её за полиномиальное время. В сложных случаях заданные целые числа имеют порядок 2^n . Есть описанный квантовый алгоритм, который решает данную задачу за время $O(2^{0.241n})$. Этот квантовый алгоритм применяет вариант квантового блуждания А. Амбайниса для различения элементов, что позволяет ускорить изощрённый классический алгоритм Хаугрейва-Грэма (Howgrave-Graham) и Жю (Joux). Наиболее эффективный классический алгоритм для этой задачи использует время $O(2^{0.291n})$.

- Bernstein D. J., Jeffrey S., Lange T., Meurer A. *Quantum algorithms for the subset-sum problem*. From cr.yp.to.
- Ambainis A. *Quantum walk algorithm for element distinctness*. SIAM Journal on Computing, 37:210-239, 2007. [arXiv:quant-ph/0311001](https://arxiv.org/abs/quant-ph/0311001).

Алгоритмы со специальными оракулами

Квантовый поиск (алгоритм Гровера)

Ускорение: полиномиальное.

Описание: Пусть задан оракул с N входами. Для одного входного значения w соответствующее выходное значение равно 1, а на всех остальных входных значениях выходом является 0. Задачей является найти w . На классическом компьютере эта задача решается за $\Omega(N)$ запросов к оракулу. Квантовый алгоритм Л. Гровера достигает этой цели за $O(\sqrt{N})$ вызовов оракула.

Этот алгоритм был последовательно обобщён для решения следующих задач:

1. Общий поиск при наличии нескольких целевых значений.
2. Оценка суммы значений произвольной функции.
3. Поиск глобального минимума произвольной функции.
4. Использование преимуществ альтернативных начальных состояний.
5. Использование преимуществ неравномерных априорных вероятностей.
6. Работа с оракулами, чьё время исполнения варьирует в зависимости от входного значения.
7. Оценка значений определённых интегралов.
8. Сходимость функции к неподвижной точке.

Обобщение алгоритма Гровера, известное как «оценка амплитуды», сегодня является важным примитивом среди квантовых алгоритмов. Оценка амплитуды является основой для большинства известных квантовых алгоритмов, связанных с поиском коллизий и свойствами графов. Одним из естественных применений алгоритма Гровера является ускорение решения NP-полных задач, таких как 3-SAT (задача о выполнимости булевых формул для 3-конъюнктивных нормальных форм). Такое применение не является тривиальным, поскольку наилучший из известных классических алгоритмов для задачи 3-SAT не просто осуществляет полный перебор. Тем не менее, методы усиления амплитуды позволяют достичь квадратичного ускорения для самых лучших классических алгоритмов для решения задачи 3-SAT. Также показана возможность квадратичного ускорения для иных задач выполнимости формул.

Данный алгоритм детально описан в главе 4.

- Grover L. K. *Quantum mechanics helps in searching for a needle in a haystack*. Physical Review Letters, 79(2):325-328, 1997. [arXiv:quant-ph/9605043](https://arxiv.org/abs/quant-ph/9605043).
- Boyer M., Brassard G., Høyer P., Tapp A. *Tight bounds on quantum searching*. Fortschritte der Physik, 46:493-505, 1998.
- Brassard G., Høyer P., Tapp A. *Quantum counting*. 1998. [arXiv:quant-ph/9805082](https://arxiv.org/abs/quant-ph/9805082).
- Mosca M. *Quantum searching, counting, and amplitude amplification by eigenvector analysis*. In R. Freivalds, editor, Proceedings of International Workshop on Randomized Algorithms, pages 90-100, 1998.
- Dürr C., Høyer P. *A quantum algorithm for finding the minimum*. 1996. [arXiv:quant-ph/9607014](https://arxiv.org/abs/quant-ph/9607014).

- Nayak A., Wu F. *The quantum query complexity of approximating the median and related statistics*. In Proceedings of 31st ACM Symposium on the Theory of Computing, 1999. [arXiv:quant-ph/9804066](https://arxiv.org/abs/quant-ph/9804066).
- Biham E., Biham O., Biron D., Grassl M., Lidar D. *Grover's quantum search algorithm for an arbitrary initial amplitude distribution*. Physical Review A, 60(4):2742, 1999. [arXiv:quant-ph/9807027](https://arxiv.org/abs/quant-ph/9807027), [arXiv:quant-ph/0010077](https://arxiv.org/abs/quant-ph/0010077).
- Montanaro A. *Quantum search with advice*. In Proceedings of the 5th conference on Theory of quantum computation, communication, and cryptography (TQC 2010). [arXiv:0908.3066](https://arxiv.org/abs/0908.3066).
- Ambainis A. *Variable time amplitude amplification and a faster quantum algorithm for solving systems of linear equations*. 2010. [arXiv:1010.4458](https://arxiv.org/abs/1010.4458).
- Novak E. *Quantum complexity of integration*. Journal of Complexity, 17:2-16, 2001. [arXiv:quant-ph/0008124](https://arxiv.org/abs/quant-ph/0008124).
- Grover L. *Fixed-point quantum search*. Phys. Rev. Lett. 95(15):150501, 2005. [arXiv:quant-ph/0503205](https://arxiv.org/abs/quant-ph/0503205).
- Tulsi T., Grover L. K., Patel A. *A new algorithm for fixed point quantum search*. Quantum Information and Computation 6(6):483-494, 2005. [arXiv:quant-ph/0505007](https://arxiv.org/abs/quant-ph/0505007).
- Brassard G., Høyer P., Mosca M., Tapp A. *Quantum amplitude amplification and estimation*. In Samuel J. Lomonaco Jr. and Howard E. Brandt, editors, Quantum Computation and Quantum Information: A Millennium Volume, volume 305 of AMS Contemporary Mathematics Series. American Mathematical Society, 2002. [arXiv:quant-ph/0005055](https://arxiv.org/abs/quant-ph/0005055).
- Ambainis A. *Quantum Search Algorithms*. SIGACT News, 35 (2):22-35, 2004. [arXiv:quant-ph/0504012](https://arxiv.org/abs/quant-ph/0504012).
- Cerf N. J., Grover L. K., Williams C. P. *Nested quantum search and NP-hard problems*. Applicable Algebra in Engineering, Communication and Computing, 10 (4-5):311-338, 2000.

Скрытая абелева подгруппа

Ускорение: сверхполиномиальное.

Описание: Пусть G является конечно порождённой абелевой группой и пусть H есть некоторая подгруппа G такая, что разность G/H является конечной. Пусть f является функцией на G такой, что для любых $g_1, g_2 \in G, f(g_1) = f(g_2)$ тогда и только тогда, когда g_1 и g_2 находятся в одном смежном классе H . Задачей является найти H (то есть множество генераторов для H) при помощи запросов к функции f . Эта задача решается на квантовом компьютере при помощи $O(\log|G|)$ запросов, в то время как на классическом компьютере требуется $\Omega(|G|)$ запросов. Впервые этот алгоритм был полностью сформулирован Д. Бони и Р. Липтоном. Однако точное определение авторства алгоритма затруднительно, поскольку как написано в главе 5 книги Нильсена и Чанга (см. Обзор литературы), этот алгоритм в качестве специальных случаев обобщает множество исторически важных алгоритмов, включая алгоритм Саймона, который вдохновил П. Шора на разработку алгоритма нахождения периода функции, который является ядром его алгоритмов факторизации и нахождения дискретного логарифма. Алгоритм поиска скрытой абелевой подгруппы лежит в основе таких алгоритмов, как уравнение Пелля, главный идеал, единичная группа и группа классов. Кстати, в некоторых случаях задача поиска скрытой абелевой подгруппы может быть вообще решена при помощи одного запроса к оракулу.

- Boneh D., Lipton R. J. *Quantum cryptanalysis of hidden linear functions*. In Don Coppersmith, editor, CRYPTO'95, Lecture Notes in Computer Science, pages 424-437. Springer-Verlag, 1995.
- Nielsen M. A., Chuang I. L. *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge, UK, 2000.

- Simon D. *On the Power of Quantum Computation*. In Proceedings of the 35th Symposium on Foundations of Computer Science, pg. 116-123, 1994.
- Beaudrap de J. N., Cleve R., Watrous J. *Sharp quantum versus classical query complexity separations*. Algorithmica, 34(4):449-461, 2002. [arXiv:quant-ph/0011065v2](https://arxiv.org/abs/quant-ph/0011065v2).

Скрытая (неабелева) подгруппа

Ускорение: сверхполиномиальное.

Описание: Пусть G является конечно порождённой группой и пусть H есть некоторая подгруппа G , у которой конечное число левых смежных классов. Пусть f является функцией на G такой, что для любых $g_1, g_2 \in G, f(g_1) = f(g_2)$ тогда и только тогда, когда g_1 и g_2 находятся в одном левом смежном классе H . Задачей является найти H (то есть множество генераторов для H) при помощи запросов к функции f . Эта задача решается на квантовом компьютере при помощи $O(\log|G|)$ запросов, в то время как на классическом компьютере требуется $\Omega(|G|)$ запросов. Однако этот результат не может считаться эффективным квантовым алгоритмом, поскольку в общем случае для изучения квантовых состояний, получаемых при помощи запросов, может потребоваться экспоненциальное количество времени. Эффективные квантовые алгоритмы для решения задачи скрытой подгруппы известны для некоторых конкретных типов неабелевых групп. В литературе есть обзоры этого списка. Особенный интерес представляют собой симметричные и двугранные группы. Решение для симметричных групп позволит решить задачу изоморфизма графов. Решение для двугранных групп поможет решить проблемы оптимизации на решётках. Но, несмотря на значительные усилия, до сих пор нет эффективного решения для этого типа групп. Однако Г. Куперберг показал, что есть алгоритм с временем работы $2^{O(\sqrt{\log N})}$ для двугранных

групп D_N . О. Регев далее улучшил этот алгоритм таким образом, что он не только требует субэкспоненциального времени, но и полиномиальной памяти.

- Ettinger M., Høyer P., Knill E. *The quantum query complexity of the hidden subgroup problem is polynomial*. Information Processing Letters, 91(1):43-48, 2004. [arXiv:quant-ph/0401083](https://arxiv.org/abs/quant-ph/0401083).
- Hallgren S., Russell A., Ta-Shma A. *Normal subgroup reconstruction and quantum computation using group representations*. SIAM Journal on Computing, 32(4):916-934, 2003.
- Roetteler M., Beth T. *Polynomial-time solution to the hidden subgroup problem for a class of non-abelian groups*. 1998. [arXiv:quant-ph/9812070](https://arxiv.org/abs/quant-ph/9812070).
- Ivanyos G., Magniez F., Santha M. *Efficient quantum algorithms for some instances of the non-abelian hidden subgroup problem*. In Proceedings of the 13th ACM Symposium on Parallel Algorithms and Architectures, pages 263-270, 2001. [arXiv:quant-ph/0102014](https://arxiv.org/abs/quant-ph/0102014).
- Moore C., Rockmore D., Russell A., Schulman L. *The power of basis selection in Fourier sampling: the hidden subgroup problem in affine groups*. In Proceedings of the 15th ACM-SIAM Symposium on Discrete Algorithms, pages 1113-1122, 2004. [arXiv:quant-ph/0211124](https://arxiv.org/abs/quant-ph/0211124).
- Inui Y., Le Gall F. *Efficient quantum algorithms for the hidden subgroup problem over a class of semi-direct product groups*. Quantum Information and Computation, 7(5/6):559-570, 2007. [arXiv:quant-ph/0412033](https://arxiv.org/abs/quant-ph/0412033).
- Bacon D., Childs A. M., Dam van W. *From optimal measurement to efficient quantum algorithms for the hidden subgroup problem over semidirect product groups*. In Proceedings of the 46th IEEE Symposium on Foundations of Computer Science, pages 469-478, 2005. [arXiv:quant-ph/0504083](https://arxiv.org/abs/quant-ph/0504083).

- Chi D. P., Kim J. S., Lee S. *Notes on the hidden subgroup problem on some semi-direct product groups*. Phys. Lett. A 359(2):114-116, 2006. [arXiv:quant-ph/0604172](https://arxiv.org/abs/quant-ph/0604172).
- Ivanyos G., Sanselme L., Santha M. *An efficient quantum algorithm for the hidden subgroup problem in extraspecial groups*. In Proceedings of the 24th Symposium on Theoretical Aspects of Computer Science, 2007. [arXiv:quant-ph/0701235](https://arxiv.org/abs/quant-ph/0701235).
- Magno C., Cosme M., Portugal R. *Quantum algorithm for the hidden subgroup problem on a class of semidirect product groups*. 2007. [arXiv:quant-ph/0703223](https://arxiv.org/abs/quant-ph/0703223).
- Ivanyos G., Sanselme L., Santha M. *An efficient quantum algorithm for the hidden subgroup problem in nil-2 groups*. In LATIN 2008: Theoretical Informatics, pg. 759-771, Springer (LNCS 4957). [arXiv:0707.1260](https://arxiv.org/abs/0707.1260).
- Friedl K., Ivanyos G., Magniez F., Santha M., Sen P. *Hidden translation and orbit coset in quantum computing*. In Proceedings of the 35th ACM Symposium on Theory of Computing, pages 1-9, 2003. [arXiv:quant-ph/0211091](https://arxiv.org/abs/quant-ph/0211091).
- Gavinsky D. *solution to the hidden subgroup problem for poly-near-Hamiltonian-groups*. Quantum Information and Computation, 4:229-235, 2004.
- Childs A. M., Dam van W. *Quantum algorithm for a generalized hidden shift problem*. In Proceedings of the 18th ACM-SIAM Symposium on Discrete Algorithms, pages 1225-1232, 2007. [arXiv:quant-ph/0507190](https://arxiv.org/abs/quant-ph/0507190).
- Denney A., Moore C., Russell A. *Finding conjugate stabilizer subgroups in $PSL(2;q)$ and related groups*. Quantum Information and Computation 10(3):282-291, 2010. [arXiv:0809.2445](https://arxiv.org/abs/0809.2445).
- Wallach N. *A quantum polylog algorithm for non-normal maximal cyclic hidden subgroups in the affine group of a finite field*. 2013. [arXiv:1308.1415](https://arxiv.org/abs/1308.1415).
- Lomont C. *The hidden subgroup problem — review and open problems*. 2004. [arXiv:quant-ph/0411037](https://arxiv.org/abs/quant-ph/0411037).

- Regev O. *Quantum computation and lattice problems*. In Proceedings of the 43rd Symposium on Foundations of Computer Science, 2002. [arXiv:cs/0304005](https://arxiv.org/abs/cs/0304005).
- Kuperberg G. *A subexponential-time quantum algorithm for the dihedral hidden subgroup problem*. SIAM Journal on Computing, 35(1):170-188, 2005. [arXiv:quant-ph/0302112](https://arxiv.org/abs/quant-ph/0302112).
- Regev O. *A subexponential time algorithm for the dihedral hidden subgroup problem with polynomial space*. 2004. [arXiv:quant-ph/0406151](https://arxiv.org/abs/quant-ph/0406151).

Алгоритм Бернштейна-Вазирани

Ускорение: полиномиальное.

Описание: Пусть дан оракул, который принимает на вход n битов, а на выходе выдаёт 1 бит. Для заданного входа $x \in \{0,1\}^n$ выходом является значение $x \odot h$, где h — это «скрытая» строка из n битов, а символ \odot обозначает побитовое внутреннее произведение по модулю 2. Задачей является найти значение h . На классическом компьютере для решения этой задачи требуется выполнить n запросов. Как показано Э. Бернштейном и У. Вазирани на квантовом компьютере это можно осуществить при помощи одного запроса. Более того, можно создать рекурсивную версию этой задачи, называемой рекурсивным разложением Фурье, для решения которой на квантовом компьютере требуется экспоненциально меньше запросов, чем на классическом.

- Bernstein E., Vazirani U. *Quantum complexity theory*. In Proceedings of the 25th ACM Symposium on the Theory of Computing, pages 11-20, 1993.

Алгоритм Дойча-Йожи

Ускорение: экспоненциальное в классе сложности P, нет ускорения в классе сложности BPP.

Описание: Дан оракул, который получает на вход n бит, а на выходе отдаёт 1 бит. Точно известно, что на всех возможных 2^n вариантах входного значения, либо все значения оракула, либо ни одного значения, либо ровно половина значений оракула равны 1. Задачей является классификация функции-оракула на сбалансированный (половина вариантов входного параметра даёт 1 на выходе) и константный (либо все, либо ни один входной вариант не даёт значения 1 на выходе). Д. Дойчем было показано, что при $n = 1$ требуется один запрос к оракулу, в то время как любой детерминированный классический алгоритм требует два запроса. Исторически это был первый хорошо определённый квантовый алгоритм, который работает эффективнее классического. Далее Д. Дойчем и Р. Йожей был разработан алгоритм, который требует одного запроса к оракулу для произвольного n . Несмотря на то, что эту задачу можно решить вероятностным алгоритмом за $O(1)$ запросов, задача Дойча-Йожи остаётся примером проблемы, требующей экспоненциально много запросов в классическом случае.

Данный алгоритм детально описан в главе 4.

- Deutsch D. *Quantum theory, the Church-Turing principle, and the universal quantum computer*. Proceedings of the Royal Society of London Series A, 400:97-117, 1985.
- Deutsch D., Jozsa R. *Rapid solution of problems by quantum computation*. Proceedings of the Royal Society of London Series A, 493:553-558, 1992.

Вычисление формулы

Ускорение: полиномиальное.

Описание: Булево выражение называется формулой, если каждая переменная в нём используется только один раз. Формула соответствует схеме, в которой нет операции FANOUT (дублирования) и которая имеет топологию дерева. Формализм

разделения программ Б. В. Рейхарда позволяет сказать, что сложность квантовых запросов для формулы с N переменными и $O(1)$ выходами составляет $\Theta(\sqrt{N})$. Этот результат основывается на длинном ряду исследований, который начался с открытия Э. Фархи со товарищи, которые показали, что деревья операция И-НЕ на 2^n переменных могут быть вычислены на квантовых компьютерах за время $O(2^{0.5n})$ при помощи непрерывного по времени квантового блуждания, в то время как на классическом компьютере требуется $\Omega(2^{0.753n})$ запросов. В большом количестве случаев квантовые алгоритмы вычисления формул являются более эффективными не только с точки зрения количества запросов, но и с точки зрения используемого времени. Формализм разделения программ также даёт оценку нижней границы квантовой сложности. Хотя изначально алгоритм Гровера был открыт с другой точки зрения, он может рассматриваться как особый случай задачи вычисления булевой формулы, в которой все операции являются операциями ИЛИ. Квантовая сложность вычисления небулевых формул также изучалась, однако результаты всё ещё сложно проинтерпретировать. А. Чайлдс со своими коллегами обобщил результаты этих исследований для случая, когда входные переменные могут повторяться, то есть в случае если первый слой операций содержит операции дублирования FANOUT. Они получили квантовый алгоритм, который использует $O(\min\{N, \sqrt{S}, N^{1/2}G^{1/4}\})$ запросов, где N — количество входных переменных, которые не подвергаются дублированию; S — количество переменных, которые дублируются; G — количество операций в формуле. Также в литературе есть описания специальных случаев, когда количество операций И-НЕ ограничено. В таких случаях при помощи квантового компьютера иногда удаётся достигнуть сверхполиномиального ускорения по сравнению с классическим.

- Reichardt B. W. *Reflections for quantum query algorithms*. In Proceedings of the 22nd ACM-SIAM Symposium on Discrete Algorithms (SODA), pg. 560-569, 2011. [arXiv:1005.1601](https://arxiv.org/abs/1005.1601).
- Childs A. M., Cleve R., Jordan S. P., Yeung D. *Discrete-query quantum algorithm for NAND trees*. Theory of Computing, 5:119-123, 2009. [arXiv:quant-ph/0702160](https://arxiv.org/abs/quant-ph/0702160).
- Ambainis A., Childs A. M., Reichardt B. W., Špalek R., Zheng S. *Every AND-OR formula of size N can be evaluated in time $n^{1/2 + o(1)}$ on a quantum computer*. In Proceedings of the 48th IEEE Symposium on the Foundations of Computer Science, pages 363-372, 2007. [arXiv:quant-ph/0703015](https://arxiv.org/abs/quant-ph/0703015). [arXiv:0704.3628](https://arxiv.org/abs/0704.3628).
- Reichardt B. W., Špalek R. *Span-program-based quantum algorithm for evaluating formulas*. Proceedings of STOC 2008. [arXiv:0710.2630](https://arxiv.org/abs/0710.2630).
- Reichardt B. W. *Span-program-based quantum algorithm for evaluating unbalanced formulas*. 2009. [arXiv:0907.1622](https://arxiv.org/abs/0907.1622).
- Reichardt B. W. *Faster quantum algorithm for evaluating game trees*. In Proceedings of the 22nd ACM-SIAM Symposium on Discrete Algorithms (SODA), pg. 546-559, 2011. [arXiv:0907.1623](https://arxiv.org/abs/0907.1623).
- Farhi E., Goldstone J., Gutmann S. *A quantum algorithm for the Hamiltonian NAND tree*. Theory of Computing 4:169-190, 2008. [arXiv:quant-ph/0702144](https://arxiv.org/abs/quant-ph/0702144).
- Reichardt B. W. *Span programs and quantum query complexity: The general adversary bound is nearly tight for every Boolean function*. In Proceedings of the 50th IEEE Symposium on Foundations of Computer Science (FOCS '09), pg. 544-551, 2009. [arXiv:0904.2759](https://arxiv.org/abs/0904.2759).
- Cleve R., Gavinsky D., Yeung D. L. *Quantum algorithms for evaluating MIN-MAX trees*. In Theory of Quantum Computation, Communication, and Cryptography, pages 11-15, Springer, 2008. (LNCS Vol. 5106). [arXiv:0710.5794](https://arxiv.org/abs/0710.5794).

- Childs A., Kimmel S., Kothari R. *The quantum query complexity of read-many formulas*. In Proceedings of ESA 2012, pg. 337-348, Springer. (LNCS 7501). 2011. [arXiv:1112.0548, 2011](#).
- Zhan B., Kimmel S., Hassidim A. *Super-polynomial quantum speed-ups for Boolean evaluation trees with hidden structure*. ITCS 2012: Proceedings of the 3rd Innovations in Theoretical Computer Science, ACM, pg. 249-265. [arXiv:1101.0796](#).
- Kimmel S. *Quantum adversary (upper) bound*. 39th International Colloquium on Automata, Languages and Programming — ICALP 2012 Volume 7391, p. 557-568. [arXiv:1101.0797](#).

Градиенты, структурированный поиск и обучающие полиномы

Ускорение: полиномиальное.

Описание: Пусть дан оракул для некоторой гладкой функции $f: \mathbb{R}^d \rightarrow \mathbb{R}$. Входы и выходы функции f задаются оракулу с конечной точностью. Задачей является получение оценки градиента ∇f для некоторой конкретной точки $x_0 \in \mathbb{R}^d$. Для решения этой задачи квантовому компьютеру требуется один запрос, в то время как на классическом необходимо выполнить, по крайней мере, $d + 1$ запрос. Д. Булджер описал потенциальное применение этого алгоритма в задачах оптимизации. Далее было показано, что на квантовом компьютере можно найти минимум квадратичной формы в d -мерном пространстве при помощи $O(d)$ запросов, в то время как на классическом компьютере требуется $\Omega(d^2)$ запросов. Также описаны квантовые алгоритмы с одним запросом для отыскания минимумов резервуаров, основанные на расстоянии Хэмминга. Ещё описан квантовый алгоритм, который может получить все d^2 элементов квадратной матрицы посредством $O(d)$ запросов, и более общо: можно найти все d^n производных n -го порядка при помощи $O(d^{n-1})$ запросов. По-

казано, что квадратичные формы и мультилинейные полиномы d переменных над конечными полями могут быть получены на квантовом компьютере в d раз эффективнее, чем на классическом компьютере.

- Jordan S. P. *Fast quantum algorithm for numerical gradient estimation*. Physical Review Letters, 95:050501, 2005. [arXiv:quant-ph/0405146](https://arxiv.org/abs/quant-ph/0405146).
- Bulger D. *Quantum basin hopping with gradient-based local optimisation*. 2005. [arXiv:quant-ph/0507193](https://arxiv.org/abs/quant-ph/0507193).
- Jordan S. P. *Quantum Computation Beyond the Circuit Model*. PhD thesis, Massachusetts Institute of Technology, 2008. [arXiv:0809.2307](https://arxiv.org/abs/0809.2307).
- Yao A. *On computing the minima of quadratic forms*. In Proceedings of the 7th ACM Symposium on Theory of Computing, pages 23-26, 1975.
- Hogg T. *Highly structured searches with quantum computers*. Physical Review Letters 80: 2473, 1998.
- Hunziker M., Meyer D. A. *Quantum algorithms for highly structured search problems*. Quantum Information Processing, Vol. 1, No. 3, pg. 321-341, 2002.
- M. Rötteler *Quantum algorithms to solve the hidden shift problem for quadratics and for functions of large Gowers norm*. In Proceedings of MFCS 2009, pg 663-674. [arXiv:0911.4724](https://arxiv.org/abs/0911.4724).
- Montanaro A. *The quantum query complexity of learning multilinear polynomials*. Information Processing Letters, 112(11):438-442, 2012. [arXiv:1105.3310](https://arxiv.org/abs/1105.3310).

Скрытый сдвиг

Ускорение: сверхполиномиальное.

Описание: Пусть дан оракул некоторой функции f из \mathbb{Z}_N . Известно, что $f(x) = g(x + s)$, где g — известная функция, а s — неизвестный сдвиг. В задаче скрытого сдвига необходимо

найти s . При помощи сведения к алгоритму Гровера становится понятно, что для отыскания значения скрытого сдвига необходимо осуществить, по крайней мере, \sqrt{N} запросов. Однако в некоторых специальных случаях задачу скрытого сдвига можно решить за $O(1)$ запросов. В частности, В. ван Дам со товарищи показал, что это можно сделать в случае, если f является мультипликативной характеристикой конечного кольца или поля. Ранее обнаруженный алгоритм вычисления сдвинутого символа Лежандра является специальным типом этой задачи, поскольку символ Лежандра $\left(\frac{x}{p}\right)$ является мультипликативной характеристикой \mathbb{F}_p . Для этих задач неизвестно классических алгоритмов, выполняющихся за время $O(\text{polylog}(N))$. Как следствие, нахождение символа Лежандра скомпрометирует некоторые криптографические алгоритмы генерации псевдослучайных чисел, поскольку появится возможность делать квантовые запросы к генератору. М. Рётеллер нашёл алгоритм со сверхполиномиальным ускорением, который позволяет отыскивать скрытый сдвиг для некоторых нелинейных булевых функций. На основе этой работы в дальнейшем было показано, что для произвольной булевой функции $f: \mathbb{Z}_2^n \rightarrow \mathbb{Z}_2$ усреднённая сложность квантового алгоритма равна $O(n)$, в то время как классические алгоритмы показывают в лучшем случае результат $\Omega(2^{n/2})$. Также при помощи задачи скрытой подгруппы для двугранных групп показано, что для инъективных функций в \mathbb{Z}_N квантовая сложность по количеству запросов есть $O(\log N)$, тогда как для классических алгоритмов это значение равно $\Theta(\sqrt{N})$. Однако квантовая сложность по размеру квантовой схемы для инъективных функций в \mathbb{Z}_N есть $O(2^{c\sqrt{\log n}})$, что достигается при помощи решета Куперберга.

- Dam van W., Hallgren S., Ip L. *Quantum algorithms for some hidden shift problems*. SIAM Journal on Computing, 36(3):763-778, 2006. [arXiv:quant-h/0211140](https://arxiv.org/abs/quant-h/0211140).

- Dam van W., Hallgren S. *Efficient quantum algorithms for shifted quadratic character problems*. 2000. [arXiv:quant-ph/0011067](https://arxiv.org/abs/quant-ph/0011067).
- Dam van W. *Quantum algorithms for weighing matrices and quadratic residues*. Algorithmica, 34(4):413-428, 2002. [arXiv:quant-ph/0008059](https://arxiv.org/abs/quant-ph/0008059).
- Rötteler M. *Quantum algorithms for highly non-linear Boolean functions*. Proceedings of SODA 2010 [arXiv:0811.3208](https://arxiv.org/abs/0811.3208).
- Rötteler M. *Quantum algorithms to solve the hidden shift problem for quadratics and for functions of large Gowers norm*. In Proceedings of MFCS 2009, pg 663-674. [arXiv:0911.4724](https://arxiv.org/abs/0911.4724).
- Gavinsky D., Rötteler M., Roland J. *Quantum algorithm for the Boolean hidden shift problem*. In Proceedings of the 17th annual international conference on Computing and combinatorics (COCOON '11), 2011. [arXiv:1103.3017](https://arxiv.org/abs/1103.3017).
- Ettinger M., Høyer P. *On quantum algorithms for noncommutative hidden subgroups*. Advances in Applied Mathematics, Vol. 25, No. 3, pg. 239-251, 2000. [arXiv:quant-ph/9807029](https://arxiv.org/abs/quant-ph/9807029).
- Kuperberg G. *A subexponential-time quantum algorithm for the dihedral hidden subgroup problem*. SIAM Journal on Computing, 35(1):170-188, 2005. [arXiv:quant-ph/0302112](https://arxiv.org/abs/quant-ph/0302112).

Сопоставление с образцом

Ускорение: сверхполиномиальное.

Описание: Пусть есть строка T длины n и строка P длины $m < n$, обе состоят из символов некоторого конечного алфавита. Задачей сопоставления с образцом является поиск проявления строки P в качестве подстроки T , либо сигнал о том, что строка P не является подстрокой T . В более общей задаче объекты T и P могут представлять собой d -мерные массивы, нежели одномерные массивы (строки). В этом случае задачей сопоставления с образцом является нахождение позиции массива P

в виде блока размера $t \times t \times \dots \times t$ в массиве T в виде блока размера $n \times n \times \dots \times n$. Имея в наличии алгоритм неструктурированного квантового поиска, можно сказать, что наихудшей оценкой сложности данного алгоритма будет $\Omega(\sqrt{n} + \sqrt{m})$. Разработан квантовый алгоритм, который имеет такую сложность с точностью до логарифмического множителя. Этот алгоритм основан на алгоритме Гровера, а также использует классическую технику, называемую детерминированной выборкой. Также было показано, что если $t > \log n$, то можно достигнуть сверхполиномиального ускорения. В частности, разработан квантовый алгоритм, который решает усреднённую задачу сопоставления с образцом за время $\tilde{O}((n/m)^{d/2} 2^{O(d^{3/2} \sqrt{\log m})})$. Этот квантовый алгоритм основан на обобщении квантового решета Куперберга для двугранных скрытых подгрупп и задачи скрытого сдвига, так что он может работать на d -мерных пространствах с небольшим уровнем шума.

- Bennett C. H., Bernstein E., Brassard G., Vazirani U. *Strengths and weaknesses of quantum computing*. SIAM J. Comput. 26(5):1524-1540, 1997. [arXiv:quant-ph/9701001](https://arxiv.org/abs/quant-ph/9701001).
- Ramesh H., Vinay V. *String matching in $\tilde{O}(\sqrt{n} + \sqrt{m})$ quantum time*. Journal of Discrete Algorithms 1:103-110, 2003. [arXiv:quant-ph/0011049](https://arxiv.org/abs/quant-ph/0011049).
- Montanaro A. *Quantum pattern matching fast on average*. [arXiv:1408.1816](https://arxiv.org/abs/1408.1816).
- Kuperberg G. *A subexponential-time quantum algorithm for the dihedral hidden subgroup problem*. SIAM Journal on Computing, 35(1):170-188, 2005. [arXiv:quant-ph/0302112](https://arxiv.org/abs/quant-ph/0302112).

Линейные системы

Ускорение: сверхполиномиальное.

Описание: Пусть дан оракул для квадратной матрицы A размера $n \times n$, а также некоторое описание вектора b . Необходимо

найти некоторое свойство значения $f(A)b$ для заданной эффективно вычисляемой функции f . Пусть A является эрмитовой матрицей с $O(\text{polylog}(n))$ ненулевыми элементами в каждой строке и условным числом k . На квантовом компьютере можно с полиномиальной точностью вычислить оценки значений различных операторов по отношению к вектору $f(A)b$ за время $O(k^2 \log n)$ (при условии, если можно эффективно выразить вектор b в виде соответствующего квантового состояния). Для некоторых функций, например для $f = 1/x$, этот алгоритм может быть расширен и для неэрмитовых матриц, и даже для неквадратных матриц. Далее время работы этого алгоритма было уменьшено до $O(k \log^3 k \log n)$. Расширения этого алгоритма применяются для решения линейных систем дифференциальных уравнений, в методе наименьших квадратов при аппроксимации, а также в машинном обучении.

- Harrow A. W., Hassidim A., Lloyd S. *Quantum algorithm for solving linear systems of equations*. Physical Review Letters 15(103):150502, 2009. [arXiv:0811.3171](https://arxiv.org/abs/0811.3171).
- Ambainis A. *Variable time amplitude amplification and a faster quantum algorithm for solving systems of linear equations*. 2010. [arXiv:1010.4458](https://arxiv.org/abs/1010.4458).
- Berry D. *Quantum algorithms for solving linear differential equations*. 2010. [arXiv:1010.2745](https://arxiv.org/abs/1010.2745).
- Wiebe N., Braun D., Lloyd S. *Quantum data-fitting*. Physical Review Letters 109, 050505, 2012. [arXiv:1204.5242](https://arxiv.org/abs/1204.5242).
- Lloyd S., Mohseni M., Robentrost P. *Quantum algorithms for supervised and unsupervised machine learning*. [arXiv:1307.0411](https://arxiv.org/abs/1307.0411).

Структурированный поиск

Ускорение: константное.

Описание: Пусть дан оракул, который представляет список из N чисел, упорядоченных от меньшего к большему. Для заданного числа x необходимо определить его позицию в списке (то место, куда его необходимо поместить, чтобы список остался упорядоченным). Наилучший классический алгоритм требует $\log_2 N$ запросов. Э. Фархи со своими коллегами показал, что на квантовом компьютере требуется всего лишь $0.53 \log_2 N$ запросов. На текущий момент наиболее эффективный квантовый алгоритм решает эту задачу за $0.433 \log_2 N$ запросов. Доказано, что нижней границей эффективности квантового алгоритма является значение $\frac{1}{\pi} \log_2 N$. Также разработан вероятностный квантовый алгоритм, ожидаемая эффективность которого находится на уровне $\frac{1}{3} \log_2 N$ запросов.

- Farhi E., Goldstone J., Gutmann S., Sipser M. *Invariant quantum algorithms for insertion into an ordered list*. 1999. [arXiv:quant-ph/9901059](https://arxiv.org/abs/quant-ph/9901059).
- Childs A. M., Landahl A. J., Parrilo P. A. *Quantum algorithms for the ordered search problem via semidefinite programming*. Physical Review A, 75 032335, 2007. [arXiv:quant-ph/0608161](https://arxiv.org/abs/quant-ph/0608161).
- Childs A., Lee T. *Optimal quantum adversary lower bounds for ordered search*. Proceedings of ICALP, 2008. [arXiv:0708.3396](https://arxiv.org/abs/0708.3396).
- Ben-Or M., Hassidim A. *Quantum search in an ordered list via adaptive learning*. 2007. [arXiv:quant-ph/0703231](https://arxiv.org/abs/quant-ph/0703231).

Свойства графа в модели матрицы смежности

Ускорение: полиномиальное.

Описание: Пусть G есть граф с n вершинами. Дан оракул, который для пары числе из множества $\{1, 2, \dots, n\}$ отвечает на вопрос, соединены ли соответствующие вершины графа ребром. К. Дюрр со товарищи показал, что квантовая сложность

по количеству запросов для поиска минимального остовного дерева для взвешенных графов, а также определение связности для направленных и ненаправленных графов, равна $O(n^{3/2})$, а поиск пути с наименьшим весом имеет квантовую сложность по запросам равную $O(n^{3/2} \log_2 n)$. А. Берзина со своими коллегами показал, что определение того, является ли граф двудольным, можно осуществить за $O(n^{3/2})$ запросов к квантовому оракулу. Все перечисленные задачи, как считается, имеют классическую сложность, равную $\Omega(n^2)$. Свойство графа называется «разреженным», если существует константа c такая, что для любого графа, обладающего рассматриваемым свойством, отношение количества рёбер к количеству вершин не превосходит c . А. Чайлдс и Р. Котари показали, что вычисление любого разреженного свойства графа имеет квантовую сложность $\Theta(n^{2/3})$, если только это свойство не может быть охарактеризовано при помощи списка запрещённых подграфов, либо сложность $o(n^{2/3})$ (маленькое «о») в противном случае. Первый алгоритм основан на алгоритме Гровера, а второй — на квантовом блуждании. По теореме Мадера разреженные свойства графа включают все замкнутые свойства миноров. К примеру таких свойств относятся: планарность, равенство лесу и отсутствие пути заданной длины. Другой интересной вычислительной проблемой является отыскание подграфа H в графе G . Простейшим случаем этой задачи является отыскание треугольника, то есть клики размера 3. Самый быстрый известный квантовый алгоритм решает эту задачу за $O(n^{9/7})$ запросов к оракулу. Классический алгоритм поиска треугольников требует $\Omega(n^2)$ запросов. Более общо, на квантовом компьютере можно найти произвольный подграф из k вершин при помощи $O(n^{2-2/k-t})$ запросов, где $t = (2k - d - 3)/(k(d + 1)(m + 2))$, d — степень вершин подграфа H , $(m + d)$ — количество рёбер подграфа. Если граф G является разреженным, то этот результат ещё может быть

улучшен, и квантовая сложность алгоритма составляет $O(n^{\frac{3}{2} - \frac{1}{\text{vc}(H)+1}})$, где $\text{vc}(H)$ представляет собой размер минимального покрытия подграфа H .

- Dürr C., Høyer P. *A quantum algorithm for finding the minimum*. 1996. [arXiv:quant-ph/9607014](https://arxiv.org/abs/quant-ph/9607014).
- Heiligman M. *Quantum algorithms for lowest weight paths and spanning trees in complete graphs*. 2003. [arXiv:quant-ph/0303131](https://arxiv.org/abs/quant-ph/0303131).
- Dürr C., Mhalla M., Lei Y. *Quantum query complexity of graph connectivity*. 2003. [arXiv:quant-ph/0303169](https://arxiv.org/abs/quant-ph/0303169).
- Dürr C., Heiligman M., Høyer P., Mhalla M. *Quantum query complexity of some graph problems*. SIAM Journal on Computing, 35(6):1310-1328, 2006. [arXiv:quant-ph/0401091](https://arxiv.org/abs/quant-ph/0401091).
- Berzina A., Dubrovsky A., Frivalds R., Lace L., Scegulnaja O. *Quantum query complexity for some graph problems*. In Proceedings of the 30th Conference on Current Trends in Theory and Practice of Computer Science, pages 140-150, 2004.
- Childs A., Kothari R. *Quantum query complexity of minor-closed graph properties*. In Proceedings of the 28th Symposium on Theoretical Aspects of Computer Science (STACS 2011), pg. 661-672. [arXiv:1011.1443](https://arxiv.org/abs/1011.1443).
- Magniez F., Nayak A., Roland J., Santha M. *Search via quantum walk*. In Proceedings STOC 2007, pg. 575-584. [arXiv:quant-ph/0608026](https://arxiv.org/abs/quant-ph/0608026).
- Lee T., Magniez F., Santha M. *Improved quantum query algorithms for triangle finding and associativity testing*. 2012. [arXiv:1210.1014](https://arxiv.org/abs/1210.1014).
- Jeffery S., Kothari R., Magniez F. *Nested quantum walks with quantum data structures*. 2012. [arXiv:1210.1199](https://arxiv.org/abs/1210.1199).

- Magniez F., Santha M., Szegedy M. *Quantum algorithms for the triangle problem*. SIAM Journal on Computing, 37(2):413-424, 2007. [arXiv:quant-ph/0310134](https://arxiv.org/abs/quant-ph/0310134).
- Belovs A. *Span programs for functions with constant-sized 1-certificates*. In Proceedings of STOC 2012, pg. 77-84. [arXiv:1105.4024](https://arxiv.org/abs/1105.4024).
- Burhman H., Dürr C., Heiligman M., Høyer P., Magniez F., Santha M., de Wolf R. *Quantum algorithms for element distinctness*. In Proceedings of the 16th IEEE Annual Conference on Computational Complexity, pages 131-137, 2001. [arXiv:quant-ph/0007016](https://arxiv.org/abs/quant-ph/0007016).
- Lee T., Magniez F., Santha M. *A learning graph based quantum query algorithm for finding constant-size subgraphs*. Chicago Journal of Theoretical Computer Science, Vol. 2012, Article 10, 2012. [arXiv:1109.5135](https://arxiv.org/abs/1109.5135).

Свойства графа в модели списка смежности

Ускорение: полиномиальное.

Описание: Пусть G есть граф с N вершинами, M рёбрами и степенью d . Также имеется оракул, запрос в который метки вершины и числа $j \in \{1, 2, \dots, d\}$ возвращает j -ую соседнюю вершину или пустое значение, если степень заданной вершины меньше d . Пусть точно определено, что граф является либо двудольным, либо далёк от двудольности в том смысле, что для достижения двудольности необходимо удалить константное число рёбер. Показано, что в этом случае квантовая сложность по запросам для определения двудольности равна $\tilde{O}(N^{1/3})$. Также показано, что разделение расширяемых графов от тех, которые не являются расширяемыми, имеет квантовую сложность $\tilde{O}(N^{1/3})$ и $\tilde{\Omega}(N^{1/4})$, в то время как классическая сложность для этой задачи равна $\tilde{\Theta}(\sqrt{N})$. Ключевым инстру-

ментом в этих алгоритмах является алгоритм Амбайниса для различения элементов.

Алгоритм поиска минимального остовного дерева имеет квантовую сложность $\Theta(\sqrt{NM})$. Определение связности графа имеет сложность $\Theta(N)$ для ненаправленных графов, и сложность $\tilde{\Theta}(\sqrt{NM})$ в случае направленных графов. Вычисление пути с наименьшим весом из заданной вершины во все другие вершины во взвешенном графе имеет квантовую сложность по запросам $\tilde{\Theta}(\sqrt{NM})$.

- Ambainis A., Childs A., Liu Y.-K. *Quantum property testing for bounded-degree graphs*. In Proceedings of RANDOM'11: Lecture Notes in Computer Science 6845, pp. 365-376, 2011. [arXiv:1012.3174](https://arxiv.org/abs/1012.3174).
- Dürr C., Heiligman M., Høyer P., Mhalla M. *Quantum query complexity of some graph problems*. SIAM Journal on Computing, 35(6):1310-1328, 2006. [arXiv:quant-ph/0401091](https://arxiv.org/abs/quant-ph/0401091).

Сваренное дерево

Ускорение: сверхполиномиальное.

Описание: Некоторые вычислительные задачи могут быть выражены в терминах сложности по запросам для поиска пути в лабиринте. Это значит, что есть граф G , для которого построен оракул. При запросе оракула с входным значением в виде метки вершины, оракул возвращает список меток смежных вершин. Задачей является поиск пути от заданной вершины к некоторой вершине-назначению. Как показано А. Чайлдсом и соавторами квантовый компьютер может сверхполиномиально превозмочь классический компьютер при решении этой задачи, по крайней мере, для некоторых типов графов. Более конкретно, пусть есть граф, который получен при помощи сваривания двух деревьев глубины n при помощи некоторого слу-

чайного процесса слияния вершин, так что у результирующего графа все вершины, кроме двух корней, имеют степень 3. Стартуя из первой корневой вершины квантовый компьютер может найти вторую корневую вершину за $\text{poly}(n)$ запросов, и это, вероятно, невозможно на классических компьютерах.

- Childs A. M., Cleve R., Deotto E., Farhi E., Gutmann S., Spielman D. A. *Exponential algorithmic speedup by quantum walk*. In Proceedings of the 35th ACM Symposium on Theory of Computing, pages 59-68, 2003. [arXiv:quant-ph/0209131](https://arxiv.org/abs/quant-ph/0209131).

Различение элементов и поиск коллизий

Ускорение: полиномиальное.

Описание: Пусть дан оракул функции, которая на области определения размера N имеет для каждого своего значения ровно два образа. Задача поиска коллизий заключается в нахождении такой пары чисел $x, y \in \{1, 2, \dots, N\}$, для которых $f(x) = f(y)$. Классический вероятностный алгоритм для этой задачи имеет сложность по количеству запросов равную $\Theta(\sqrt{N})$, в то время как квантовый компьютер может достичь того же за $O(N^{1/3})$ запросов. Если отказаться от свойства функции, что она даёт одинаковое значение для двух элементов, то классическая сложность становится равной $\Theta(N)$, а как показано для квантового компьютера, необходимо осуществить $O(N^{2/3})$ запросов, что является оптимальной оценкой. Расширение задачи с 2 до k элементов даёт сложность в $O(N^{3/4 - 1/(4(2^k - 1))})$ запросов. Для $k = 2, 3$ это значение также является оценкой сложности по времени с точностью до логарифма.

Для двух функций f и g с размером областей определения N «клешней» называется пара (x, y) такая, что $f(x) = g(y)$. Алгоритм различения элементов может решить эту задачу

в качестве специального случая за $O(N^{2/3})$ запросов, улучшая предыдущий результат в $O(N^{3/4} \log N)$ запросов.

См. также следующий алгоритм по поиску коллизий в графах.

- Brassard G., Høyer P., Tapp A. *Quantum algorithm for the collision problem*. ACM SIGACT News, 28:14-19, 1997. [arXiv:quant-ph/9705002](https://arxiv.org/abs/quant-ph/9705002).
- Burhrman H., Dürr C., Heiligman M., Høyer P., Magniez F., Santha M., de Wolf R. *Quantum algorithms for element distinctness*. In Proceedings of the 16th IEEE Annual Conference on Computational Complexity, pages 131-137, 2001. [arXiv:quant-ph/0007016](https://arxiv.org/abs/quant-ph/0007016).
- Ambainis A. *Quantum walk algorithm for element distinctness*. SIAM Journal on Computing, 37:210-239, 2007. [arXiv:quant-ph/0311001](https://arxiv.org/abs/quant-ph/0311001).
- Belovs A., Lee T. *Quantum algorithm for k -distinctness with prior knowledge on the input*. 2011. [arXiv:1108.3022](https://arxiv.org/abs/1108.3022).
- Belovs A. *Learning-graph-based quantum algorithm for k -distinctness*. Proceedings of STOC 2012, pg. 77-84. 2012. [arXiv:1205.1534](https://arxiv.org/abs/1205.1534).
- Childs A., Jeffery S., Kothari R., Magniez F. *A time-efficient quantum walk for 3-distinctness using nested updates*. 2013. [arXiv:1302.7316](https://arxiv.org/abs/1302.7316).

Коллизия графов

Ускорение: полиномиальное.

Описание: Пусть дан ненаправленный граф с n вершинами, а также оракул, который для каждой вершины возвращает метку 0 или 1. Задача коллизии графов заключается в определении при помощи запросов к оракулу, существует ли пара вершин, соединённых ребром, метки которых обе 1. Можно, например, воспользоваться алгоритмом неструктурированного поиска

Гровера с построением списка меток вершин, смежных с заданной вершиной с меткой 1. Это значит, что эта задача имеет квантовую сложность $\Omega(\sqrt{n})$, в то время как классическая сложность для неё равна $\Theta(n)$. Показано, что для графов общего вида существует квантовый алгоритм со сложностью $O(n^{2/3})$. Это значение является лучшей верхней оценкой для этой задачи на графах общего вида. Однако более строгие оценки получены для частных типов графов. Например:

1. Для графов G квантовая сложность составляет $\tilde{O}(\sqrt{n} + \sqrt{l})$, где l — количество отсутствующих рёбер в графе.
2. Для графов G квантовая сложность составляет $O(\sqrt{n}\alpha^{1/6})$, где α — размер наибольшего независимого множества.
3. Для графов G квантовая сложность составляет $O(\sqrt{n} + \sqrt{\alpha^*})$, где α^* — максимальное значение общей степени среди всех независимых множеств графа.
4. Для графов G квантовая сложность составляет $O(\sqrt{n}t^{1/6})$, где t — ширина дерева графа.

Более того, с высокой вероятностью квантовая сложность для произвольного случайного графа составляет $\tilde{O}(\sqrt{n})$, в котором наличие или отсутствие ребра между каждой парой вершин выбирается независимо с константной вероятностью (так называемые графы Эрдеша-Реньи). Также в литературе описаны несколько слишком сложных для размещения в этой книге дополнительных моделей графов.

- Magniez F., Santha M., Szegedy M. *Quantum algorithms for the triangle problem*. SIAM Journal on Computing, 37(2):413-424, 2007. [arXiv:quant-ph/0310134](https://arxiv.org/abs/quant-ph/0310134).
- Jeffery S., Kothari R., Magniez F. *Improving quantum query complexity of Boolean matrix multiplication using graph collision*. In Proceedings of ICALP 2012, pg. 522-532. [arXiv:1112.5855](https://arxiv.org/abs/1112.5855).

- Belovs A. *Learning-graph-based quantum algorithm for k -distinctness*. Proceedings of STOC 2012, pg. 77-84. 2012. [arXiv:1205.1534](https://arxiv.org/abs/1205.1534).
- Gavinsky D., Ito T. *A quantum query algorithm for the graph collision problem*. 2012. [arXiv:1204.1527](https://arxiv.org/abs/1204.1527).
- Ambainis A., Balodis K., Iraids J., Ozols R., Smotrovs J. *Parameterized quantum query complexity of graph collision*. 2013. [arXiv:1305.1021](https://arxiv.org/abs/1305.1021).

Коммутативность матриц

Ускорение: полиномиальное.

Описание: Пусть дан оракул, который предоставляет доступ к k матрицам, каждая из которых имеет размер $n \times n$. Для входных целых чисел $i, j \in \{1, 2, \dots, n\}$ и $x \in \{1, 2, \dots, k\}$ оракул возвращает элемент x -ой матрицы, находящийся на позиции (i, j) . Задача заключается в определении того, коммутируют ли эти k матриц. Как показано Ю. Итакурой на квантовом компьютере для этого необходимо выполнить $O(k^{4/5}n^{9/5})$ запросов, в то время как на классическом компьютере требуется выполнить $\Omega(kn^2)$.

- Itakura Y. K. *Quantum algorithm for commutativity testing of a matrix set*. Master's thesis, University of Waterloo, 2005. [arXiv:quant-ph/0509206](https://arxiv.org/abs/quant-ph/0509206).

Коммутативность группы

Ускорение: полиномиальное.

Описание: Пусть даны k генераторов группы G и доступ к оракулу, который осуществляет умножение в этой группе. При помощи запросов оракула необходимо определить, является ли группа коммутативной. Наилучший из известных класси-

ческих алгоритмов требует $O(k)$ запросов. На квантовом компьютере эту задачу можно решить за $\tilde{O}(k^{2/3})$ запросов.

- Magniez F., Nayak A. *Quantum complexity of testing group commutativity*. In Proceedings of 32nd International Colloquium on Automata, Languages and Programming. LNCS 3580, pg. 1312-1324, 2005. [arXiv:quant-ph/0506265](https://arxiv.org/abs/quant-ph/0506265).

Скрытые нелинейные структуры

Ускорение: сверхполиномиальное.

Описание: Произвольная абелева группа G может быть визуализирована в виде решётки. Подгруппа H группы G является подрешёткой, и классы смежности подгруппы H все являются сдвигами в этой подрешётке. Задача поиска абелевой скрытой подгруппы обычно решается при помощи получения суперпозиции над случайным классом смежности скрытой подгруппы, а затем применение преобразования Фурье для получения информации о двойственной решётке. Вместо того, чтобы обобщить эту задачу на поиск скрытой неабелевой подгруппы (см. алгоритм поиска скрытой неабелевой подгруппы), можно обобщить эту задачу до идентификации скрытых подмножеств, не являющихся решётками. Как показано А. Чайлдсом со товарищи эта задача эффективно решается для некоторых типов подмножеств, задаваемых полиномами, например, для сфер. Также в работах Т. Декера показано, как эффективно решать похожие задачи.

- Childs A. M., Schulman L. J., Vazirani U. V. *Quantum algorithms for hidden nonlinear structures*. In Proceedings of the 48th IEEE Symposium on Foundations of Computer Science, pages 395-404, 2007. [arXiv:0705.2784](https://arxiv.org/abs/0705.2784).
- Decker T., Draisma J., Wocjan P. *Quantum algorithm for identifying hidden polynomials*. Quantum Information and Computation, 9(3):215-230, 2009. [arXiv:0706.1219](https://arxiv.org/abs/0706.1219).

- Tulsi T., Grover L., Patel A. *A new algorithm for fixed point quantum search*. Quantum Information and Computation 6(6):483-494, 2005. [arXiv:quant-ph/0505007](https://arxiv.org/abs/quant-ph/0505007).

Центр радиальной функции

Ускорение: полиномиальное.

Описание: Пусть дан оракул функции f из \mathbb{R}^d в некоторое произвольное множество S , где сама функция f является сферически симметричной. Хочется определить центр симметрии с какой-то определённой точностью (для определённости пусть точность зафиксирована). Описан квантовый алгоритм, основанный на курвлетном преобразовании, который решает эту задачу за константное количество запросов к оракулу, независимо от величины d . Это даёт полиномиальное ускорение относительно классического алгоритма, который использует $\Omega(d)$ запросов. Квантовый алгоритм работает в тех случаях, когда функция f флуктуирует на достаточно малых масштабах, то есть когда уровневые множества f представляют собой достаточно тонкие сферические оболочки. Показано, что квантовый алгоритм работает в идеализированной непрерывной модели, и нестрогие аргументы показывают, что эффекты дискретизации должны быть небольшими.

- Liu Y.-K. *Quantum algorithms using the curvelet transform*. Proceedings of STOC 2009, pg. 391-400. [arXiv:0810.4968](https://arxiv.org/abs/0810.4968).

Порядок в группе и членство в группе

Ускорение: сверхполиномиальное.

Описание: Пусть конечная группа G задана в виде оракула следующим образом. Получив упорядоченную пару меток элементов группы, оракул возвращает метку их произведения. Для классической модели вычислений существует несколько сложных задач для такого рода групп. Одной такой задачей яв-

ляется отыскание порядка группы при помощи меток множества её генераторов. Другой задачей является поиск соответствия среди элементов группы заданной строке битов. Конструктивная версия этой задачи членства в случае положительного ответа заключается в нахождении декомпозиции заданного элемента в виде произведения генераторов группы. В классической модели эти задачи не могут быть решены при помощи $\text{polylog}(|G|)$ запросов, даже если группа G является абелевой. Для абелевых групп эта задача может быть решена на квантовом компьютере за $\text{polylog}(|G|)$ запросов при помощи сведения к задаче поиска скрытой абелевой подгруппы. Более того, показано, что эти проблемы могут быть решены на квантовом компьютере за такое же количество запросов для произвольной вычислимой группы. Для групп, которые заданы при помощи матриц над конечными полями, эти задачи могут быть решены на квантовом компьютере за полиномиальное время при помощи сведения их к задачам поиска дискретного логарифма и факторизации соответственно. См. также алгоритм для изоморфизма групп.

- Mosca M. *Quantum Computer Algorithms*. PhD thesis, University of Oxford, 1999.
- Watrous J. *Quantum algorithms for solvable groups*. In Proceedings of the 33rd ACM Symposium on Theory of Computing, pages 60-67, 2001. [arXiv:quant-ph/0011023](https://arxiv.org/abs/quant-ph/0011023).
- Babai L., Beals R., Seress A. *Polynomial-time theory of matrix groups*. In Proceedings of STOC 2009, pg. 55-64.

Изоморфизм групп

Ускорение: сверхполиномиальное.

Описание: Пусть G является конечной группой. Каждому элементу группы сопоставлена метка (стока битов). Для двух меток элементов группы оракул возвращает метку их произве-

дения. Если заданы оракулы двух таких групп G и G' , а также перечень их генераторов, то необходимо определить, являются ли эти две группы изоморфными. Для абелевых групп эта задача может быть решена посредством квантового алгоритма при помощи использования $\text{poly}(\log |G|, \log |G'|)$ запросов к оракулам, и этот алгоритм декомпозирует каждую абелеву группу на каноническое прямое произведение двух циклических групп. Другой квантовый алгоритм решает эту же задачу за $\text{poly}(\log |G|, \log |G'|)$ запросов для определённого класса неабелевых групп. Проще говоря, группа G входит в этот специфический класс, если она имеет нормальную абелеву подгруппу A и элемент u порядка копроизведения $|A|$ такой, что $G = \langle A, u \rangle$. К. Затлукал также приводит описание алгоритма с экспоненциальным квантовым ускорением, который решает сходную задачу, а именно проверяет эквивалентность расширений групп.

- Cheung K. K. H., Mosca M. *Decomposing finite Abelian groups*. Quantum Information and Computation 1(2):26-32, 2001. [arXiv:cs/0101004](https://arxiv.org/abs/cs/0101004).
- Gall F. L. *An efficient quantum algorithm for some instances of the group isomorphism problem*. In Proceedings of STACS 2010. [arXiv:1001.0608](https://arxiv.org/abs/1001.0608).
- Zatloukal K. C. *Classical and quantum algorithms for testing equivalence of group extensions*. 2013. [arXiv:1305.1327](https://arxiv.org/abs/1305.1327).

Статистическое различие

Ускорение: полиномиальное.

Описание: Пусть даны два чёрных ящика (оракула), которые представляют функции, областью определения которых являются целые числа от 1 до T , а областью значений являются целые числа от 1 до N . При помощи равномерно распределённого случайного выбора числа из области определения можно полу-

чить статистическое распределение значений функций. Необходимо с константной точностью определить расстояние L1 между распределениями вероятностей двух функций. В классическом случае количество требуемых запросов линейно увеличивается вместе с N . Показано, что на квантовом компьютере эту задачу можно решить за $O(\sqrt{N})$ запросов. Оценку равномерности и ортогональности распределений вероятностей на квантовом компьютере можно осуществить за $O(N^{1/3})$ запросов. Главным инструментом является алгоритм квантового счёта.

- Bravyi S., Harrow A., Hassidim A. *Quantum algorithms for testing properties of distributions*. IEEE Transactions on Information Theory 57(6):3971-3981, 2011. [arXiv:0907.3920](https://arxiv.org/abs/0907.3920).
- Brassard G., Høyer P., Tapp A. *Quantum counting*. 1998. [arXiv:quant-ph/9805082](https://arxiv.org/abs/quant-ph/9805082).

Конечные кольца и идеалы

Ускорение: сверхполиномиальное.

Описание: Пусть даны оракулы, которые реализуют операции умножения и сложения в конечном кольце R , при этом ещё также даны генераторы этого кольца. По отношению к сложению кольцо R формирует конечную абелеву группу $(R, +)$. При помощи квантового компьютера можно за время $\text{poly}(\log |R|)$ отыскать множество аддитивных генераторов $\{h_1, \dots, h_m\} \subset R$ таких, что $(R, +) \simeq \langle h_1 \rangle \times \dots \times \langle h_m \rangle$, и m полилогарифмично в $|R|$. Это позволяет эффективно вычислять тензорное произведение в R . Таким же образом можно эффективно найти аддитивное генерирующее множество для произвольного идеала кольца R . Это, в свою очередь, позволяет:

1. Найти пересечение двух идеалов;
2. Найти отношение двух идеалов;

3. Доказать принадлежность заданного элемента кольца идеалу;
4. Доказать, является ли заданный элемент единицей и если является, то найти для него обратный элемент;
5. Найти аддитивные и мультипликативные тождественные элементы;
6. Вычислить порядок идеала;
7. Решать линейные уравнения на кольце;
8. Определить, является ли идеал максимальным;
9. Найти аннигиляторы;
10. Проверить инъективность и сюръективность гомоморфизмов кольца.

Также показано, что при помощи квантового компьютера можно определить, является ли заданный полином тождественным нулевому элементу в заданном кольце. Известные классические алгоритмы решают перечисленные задачи за время $\text{poly}(|R|)$.

- Arvind V., Das B., Mukhopadhyay P. *The complexity of black-box ring problems*. In Proceedings of COCCOON 2006, pg 126-145.
- Wocjan P. M., Jordan S. P., Ahmadi H., Brennan J. P. *Efficient quantum processing of ideals in finite rings*. 2009. [arXiv:0908.0022](https://arxiv.org/abs/0908.0022).
- Arvind V., Mukhopadhyay P. *Quantum query complexity of multilinear identity testing*. In Proceedings of STACS 2009, pg. 87-98.

Фальшивые монеты

Ускорение: полиномиальное.

Описание: Пусть даны N монет, из которых k являются фальшивыми. Все настоящие монеты имеют одинаковый вес, а фальшивые монеты тоже все имеют одинаковый вес,

но другой. Есть балансовые весы, на которых можно сравнивать вес двух любых подмножеств монет. В классическом случае требуется $\Omega(k \log \frac{N}{k})$ взвешиваний, чтобы найти все фальшивые монеты. Можно определить оракул, который получает на вход два подмножества монет одинакового размера, а возвращает один бит информации — сбалансированы весы или нет. Показано, что на квантовом компьютере можно определить все фальшивые монеты за $O(k^{1/4})$ запросов. Основными идиомами для решения этой задачи являются увеличение амплитуды и алгоритм Берштейна-Вазирани.

- Terhal B., Smolin J. *Single quantum querying of a database*. Physical Review A 58:1822, 1998. [arXiv:quant-ph/9705041](https://arxiv.org/abs/quant-ph/9705041).
- Iwama K., Nishimura H., Raymond R., Teruyama J. *Quantum Counterfeit Coin Problems*. In Proceedings of 21st International Symposium on Algorithms and Computation (ISAAC2010), LNCS 6506, pp.73-84, 2010. [arXiv:1009.0416](https://arxiv.org/abs/1009.0416).

Ранг матрицы

Ускорение: полиномиальное.

Описание: Пусть есть оракул, представляющий целочисленную матрицу A размера $n \times m$. Необходимо определить ранг матрицы. В классическом варианте для этого необходимо выполнить nm запросов. Если есть гарантия того, что ранг матрицы не менее некоторого числа r , то на квантовом компьютере можно использовать меньшее количество запросов, а именно $O(\sqrt{r(n-r+1)}LT)$, где L — среднее квадратичное обратных чисел k r наибольших элементов матрицы, а T — фактор, зависящий от степени разреженности матрицы. Для общего вида матрицы $T = O(\sqrt{nm})$. Если в матрице имеется по крайней мере k ненулевых элементов в каждой строке и каждом столбце, то $T = O(k \log(n+m))$ (для достижения этой сложности по запросам в k -разреженной матрице, оракул должен прини-

мать на вход номер столбца, а возвращать список ненулевых элементов матрицы из этого столбца). В качестве специального случая этой задачи является определение для квадратной матрицы того, является ли она вырожденной (иногда называется задачей определения детерминанта). Для матриц общего вида задача определения детерминанта имеет не меньшую сложность в квантовой модели вычислений по сравнению с классической. Однако для специальных случаев (например, высокая разреженность матрицы) можно достичь существенно го квантового ускорения.

- Belovs A. *Span-program-based quantum algorithm for the rank problem*. 2011. [arXiv:1103.0842](https://arxiv.org/abs/1103.0842).
- Dörn S., Thierauf T. *The quantum query complexity of the determinant*. Information Processing Letters Vol. 109, No. 6, pg. 305-328, 2009.

Перемножение матриц на полукольцах

Ускорение: полиномиальное.

Описание: Полукольцом называется множество элементов с операциями сложения и умножения, которые подчиняются всем аксиомам кольца, за исключением требования наличия обратных элементов по сложению. Перемножение матриц на полукольцах имеет много приложений в задачах на графах. Задачу перемножения матриц на полукольцах можно ускорить при помощи адаптированного алгоритма Гровера, что приведёт к ускорению операции умножения матриц размера $n \times n$ до времени $\tilde{O}(n^{5/2})$. Для некоторых полуколец этот алгоритм превосходит по эффективности лучшие классические алгоритмы, в то время как для других нет. В частности, особый интерес вызывает булево полукольцо, в котором операция ИЛИ является сложением, а операция И является умножением. Неизвестно квантового алгоритма перемножения матриц в булевом полу-

кольце, который превосходил бы по эффективности классический алгоритм, сложность которого равна $n^{2.373}$. Однако для разреженных случаев эффективные квантовые алгоритмы известны. В частности, пусть A и B являются булевыми матрицами размера $n \times n$. Пусть $C = AB$, и пусть l представляет собой количество элементов матрицы C , равных 1, то есть ИСТИНА. Наиболее эффективный из известных квантовых алгоритмов для задачи в такой постановке имеет сложность $\tilde{O}(n\sqrt{l})$. Если же входные матрицы являются разреженными, то можно также получить квантовое ускорение по сравнению с классическими алгоритмами. В литературе представлено сравнение характеристик классических и квантовых алгоритмов для этой задачи. Например, представлены квантовые алгоритмы, которые выполняют перемножение матриц в полукольце (\max , \min) за время $\tilde{O}(n^{2.473})$, а в полукольце расстояния доминирования за время $\tilde{O}(n^{2.458})$, в то время как наилучший из классических алгоритмов выполняет перемножение в этих двух случаях за время $\tilde{O}(n^{2.687})$.

- Gall F. L., Nishimura H. *Quantum algorithms for matrix products over semirings*. 2013. [arXiv:1310.3898](https://arxiv.org/abs/1310.3898).
- Buhrman H., Špalek R. *Quantum verification of matrix products*. In Proceedings of the 17th ACM-SIAM Symposium on Discrete Algorithms, pages 880-889, 2006. [arXiv:quant-ph/0409035](https://arxiv.org/abs/quant-ph/0409035).
- Gall F. L. *Improved output-sensitive quantum algorithms for Boolean matrix multiplication*. In Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '12), 2012.
- Williams V. V., Williams R. *Subcubic equivalences between path, matrix, and triangle problems*. In 51st IEEE Symposium on Foundations of Computer Science (FOCS '10) pg. 645 - 654, 2010.
- Jeffery S., Kothari R., Magniez F. *Improving quantum query complexity of Boolean matrix multiplication using graph colli-*

tion. In Proceedings of ICALP 2012, pg. 522-532.
[arXiv:1112.5855](#).

Поиск подмножеств

Ускорение: полиномиальное.

Описание: Пусть дан оракул для функции $f: D \rightarrow R$, где D и R являются конечными множествами. Дано некоторое свойство $P \subset (D \times R)^k$, например при помощи явно заданного списка. Задачей является нахождение подмножества D размера k , которое удовлетворяет свойство P , то есть $((x_1, f(x_1)), \dots, (x_k, f(x_k))) \in P$, либо вернуть отказ, если такого подмножества не существует. Как обычно, это необходимо сделать за минимальное количество запросов к оракулу. Показано, что это можно сделать за $O(|D|^{k/(k+1)})$ запросов. В качестве одного из приложений можно указать, что этот алгоритм решает задачу поиска суммы подмножества, то есть находит k чисел из списка, сумма которых равна заданному числу.

- Ambainis A. *Quantum walk algorithm for element distinctness*. SIAM Journal on Computing, 37:210-239, 2007. [arXiv:quant-ph/0311001](#).
- Childs A. M., Eisenberg J. M. *Quantum algorithms for subset finding*. Quantum Information and Computation 5(7):593-604, 2005. [arXiv:quant-ph/0311038](#).
- Belovs A., Špalek R. *Adversary lower bound for the k -sum problem*. In Proceedings of ITCS 2013, pg. 323-328. [arXiv:1206.6528](#).

Поиск с шаблонами

Ускорение: полиномиальное.

Описание: Задача поиска с шаблонами заключается в идентификации скрытой строки x длиной n бит при помощи запросов к оракулу f . Если даны $S \subseteq \{1, 2, \dots, n\}$ и $y \in \{0, 1\}^{|S|}$, оракул f возвращает 1 в случае, если подстрока x , специфицируемая при помощи S , равна y , и возвращает 0 в противном случае. В классическом случае сложность задачи по запросам равна $\Theta(n)$. Показано, что в квантовой модели вычислений сложность по запросам равна $\Theta(\sqrt{n})$. Интерес вызывает то, что это квадратичное ускорение основано не на техниках увеличения амплитуды или квантового блуждания, а на так называемом «достаточно хорошем измерении». К этой задаче также сводится проблема проверки комбинаторной группы. В проблеме проверки комбинаторной группы опять же необходимо идентифицировать скрытую строку x длиной n бит при помощи запросов к оракулу f , однако в этой формулировке оракул принимает на вход подмножество $S \subseteq \{1, 2, \dots, n\}$, а возвращает применение операции ИЛИ ко всем соответствующим битам строки x . Если точно k бит строки x равны 1, то сложность по запросам в классическом случае равна $\Theta(k \log(n/k))$, в то время как квантовая сложность в максимальном случае равна $O(k)$ (то есть не зависит от n), но, по крайней мере, равна $\Omega(\sqrt{k})$.

- Ambainis A., Montanaro A. *Quantum algorithms for search with wildcards and combinatorial group testing*. 2012. [arXiv:1210.1148](https://arxiv.org/abs/1210.1148).

Поток по сети

Ускорение: полиномиальное.

Описание: Сетью называется направленный граф, рёбра которого помечены числами, которые обозначают ёмкость соответствующего ребра, а соединяемые этим ребром вершины называются пунктами отправления и назначения. Поток

по сети называется назначение на рёбра значений потока так, что никакое значение потока не превышает значения ёмкости ребра, и для каждой вершины, не совпадающей с пунктами отправления и назначения, сумма входящих значений потока равна сумме выходящих значений. Задача поиска потока на сети заключается в максимизации совокупного потока, идущего по сети от пункта отправления к пункту назначения. Для сети с n вершинами, m рёбрами и целочисленными ёмкостями с максимумом U квантовый алгоритм находит максимальный поток за время $O(\min\{n^{7/6}\sqrt{m}U^{1/3}, \sqrt{nUm}\} \times \log n)$. Задача потока по сети тесно связана с задачей поиска максимального совпадения на графе, то есть поиска максимального по размеру подграфа, который соединяет все вершины по крайней мере с одной другой вершиной. Эта задача решается за время $O(n\sqrt{n+m} \log n)$ в случае, если граф является двудольным, и за время $O(n^2(\sqrt{m/n} + \log n) \log n)$ в общем случае. В основе этих алгоритмов стоит алгоритм Гровера. Сложность для известных классических случаев описать сложно, поскольку классические алгоритмы для решения задач потока по сети и совпадения на графе используют разнообразные частные случаи. Однако в некоторых частных случаях представленные квантовые алгоритмы эффективнее всех известных классических алгоритмов.

- Ambainis A., Špalek R. *Quantum algorithms for matching and network flows*. Proceedings of STACS 2007, pg. 172-183. [arXiv:quant-ph/0508205](https://arxiv.org/abs/quant-ph/0508205).

Электрическое сопротивление

Ускорение: экспоненциальное.

Описание: Пусть дан оракул, представляющий собой взвешенный граф из n вершин и максимальной степенью d , при этом веса на рёбрах интерпретируются как электрическое

сопротивление. Задачей является вычисление общего сопротивления между двумя выбранными вершинами. Г. Ванг представил алгоритм для решения этой задачи, который работает за время $\text{poly}(\log n, d, 1/\varphi, 1/\varepsilon)$, где φ — расширение графа, а ответ должен быть дан с точностью $(1 + \varepsilon)$. Известные классические алгоритмы для решения этой задачи полиномиальны от n , а не от $\log n$. Квантовый алгоритм основан на новейших разработках в области квантового блуждания.

- Wang G. *Quantum algorithms for approximating the effective resistances of electrical networks*. [arXiv:1311.1851](https://arxiv.org/abs/1311.1851).

Алгоритмы аппроксимации и эмуляции

Квантовая симуляция

Ускорение: сверхполиномиальное.

Описание: На текущий момент учёные верят, что произвольный физически реализуемый гамильтониан H с n степенями свободы и с соответствующим оператором эволюции во времени e^{-iHt} может быть реализован при помощи использования $\text{poly}(n, t)$ гейтов. Если только не выполняется тождество $\text{BPP} = \text{BQP}$, то эта задача не может быть решена на классическом компьютере за полиномиальное время. Для разных способов применения разработано множество различных техник квантовой симуляции. Экспоненциальная сложность симуляции квантовых систем на классических компьютерах привела Р. Фейнмана к первой идее о том, что квантовые компьютеры для некоторых задач могут превзойти классические по эффективности. Хотя задача поиска базовых уровней энергии для локальных гамильтонианов является QMA-полной, а потому, возможно, требует экспоненциального времени работы квантового компьютера в худшем случае, были разработаны

квантовые алгоритмы для аппроксимации базовых и тепловых состояний для некоторых классов гамильтонианов.

- Childs A. M. *Quantum information processing in continuous time*. PhD thesis, MIT, 2004.
- Zalka C. *Efficient simulation of quantum systems by quantum computers*. Proceedings of the Royal Society of London Series A, 454:313, 1996. [arXiv:quant-ph/9603026](https://arxiv.org/abs/quant-ph/9603026).
- Wiesner S. *Simulations of many-body quantum systems by a quantum computer*. 1996. [arXiv:quant-ph/9603028](https://arxiv.org/abs/quant-ph/9603028).
- Aharonov D., Ta-Shma A. *Adiabatic quantum state generation and statistical zero knowledge*. In Proceedings of the 35th ACM Symposium on Theory of Computing, 2003. [arXiv:quant-ph/0301023](https://arxiv.org/abs/quant-ph/0301023).
- Abrams D. S., Lloyd S. *Simulation of many-body Fermi systems on a universal quantum computer*. Physical Review Letters, 79(13):2586-2589, 1997. [arXiv:quant-ph/9703054](https://arxiv.org/abs/quant-ph/9703054).
- Berry D. W., Ahokas G., Cleve R., Sanders B. C. *Efficient quantum algorithms for simulating sparse Hamiltonians*. Communications in Mathematical Physics, 270(2):359-371, 2007. [arXiv:quant-ph/0508139](https://arxiv.org/abs/quant-ph/0508139).
- Kassal I., Jordan S. P., Love P. J., Mohseni M., Aspuru-Guzik A. *Quantum algorithms for the simulation of chemical dynamics*. Proc. Natl. Acad. Sci. Vol. 105, pg. 18681, 2008. [arXiv:0801.2986](https://arxiv.org/abs/0801.2986).
- Lidar D. A., Wang H. *Calculating the thermal rate constant with exponential speedup on a quantum computer*. Physical Review E, 59(2):2429-2438, 1999. [arXiv:quant-ph/9807009](https://arxiv.org/abs/quant-ph/9807009).
- Wu L.-A., Byrd M. S., Lidar D. A. *Polynomial-Time Simulation of Pairing Models on a Quantum Computer*. Physical Review Letters, 89(6):057904, 2002. [arXiv:quant-ph/0108110](https://arxiv.org/abs/quant-ph/0108110).
- Byrnes T., Yamamoto Y. *Simulating lattice gauge theories on a quantum computer*. Physical Review A, 73, 022328, 2006. [arXiv:quant-ph/0510027](https://arxiv.org/abs/quant-ph/0510027).

- Ortiz G., Gubernatis J. E., Knill E., Laflamme R. *Quantum algorithms for Fermionic simulations*. Physical Review A 64: 022319, 2001. [arXiv:cond-mat/0012334](https://arxiv.org/abs/cond-mat/0012334).
- Jordan S., Lee K., Preskill J. *Quantum algorithms for quantum field theories*. Science, Vol. 336, pg. 1130-1133, 2012. [arXiv:1111.3633](https://arxiv.org/abs/1111.3633).
- Childs A., Wiebe N. *Hamiltonian simulation using linear combinations of unitary operations*. Quantum Information and Computation 12, 901-924, 2012. [arXiv:1202.5822](https://arxiv.org/abs/1202.5822).
- Berry D. W., Cleve R., Somma R. D. *Exponential improvement in precision for Hamiltonian-evolution simulation*. 2013. [arXiv:1308.5424](https://arxiv.org/abs/1308.5424).
- Berry D. W., Childs A. M., Cleve R., Kothari R., Somma R. D. *Exponential improvement in precision for simulating sparse Hamiltonians*. [arXiv:1312.1414](https://arxiv.org/abs/1312.1414).
- Feynman R. P. *Simulating physics with computers*. International Journal of Theoretical Physics, 21(6/7):467-488, 1982.
- Aspuru-Guzik A., Dutoi A. D., Love P. J., Head-Gordon M. *Simulated quantum computation of molecular energies*. Science, 309(5741):1704-1707, 2005. [arXiv:quant-ph/0604193](https://arxiv.org/abs/quant-ph/0604193).
- Temme K., Osborne T. J., Vollbrecht K. G., Poulin D., Verstraete F. *Quantum Metropolis Sampling*. Nature, Vol. 471, pg. 87-90, 2011. [arXiv:0911.3635](https://arxiv.org/abs/0911.3635).
- Poulin D., Wocjan P. *Sampling from the thermal quantum Gibbs state and evaluating partition functions with a quantum computer*. Physical Review Letters 103:220502, 2009. [arXiv:0905.2199](https://arxiv.org/abs/0905.2199).

Инварианты узлов

Ускорение: сверхполиномиальное.

Описание: Как показано М. Фридманом и соавторами нахождение определённой аддитивной аппроксимации поли-

нома Джонса к плоскому замыканию косы на $e^{i2\pi/5}$ является BQP-полной задачей. В дальнейшем этот результат был переформулирован и расширен для $e^{i2\pi/k}$ при произвольном k . Далее был обнаружен квантовый алгоритм для оценки полиномов HOMFLY, частным случаем которых является полином Джонса. Ещё более сильное утверждение было выведено в дальнейшем — на квантовых компьютерах можно за полиномиальное время оценить некоторые аддитивные аппроксимации более общих полиномов Тутте для планарных графов. Однако не совсем ясно, для каких значений параметров такая аппроксимация является BQP-сложной (см. также алгоритмы для статистической суммы). Были также обнаружены полиномиальные по времени квантовые алгоритмы для аддитивно аппроксимирующих инвариантов, которые возникают при квантовом дублировании конечных групп (хотя эта задача не является BQP-сложной). Также показано, что задача нахождения определённой аддитивной аппроксимации полинома Джонса для замыкания следа косы на $e^{i2\pi/5}$ является DQC1-полной.

- Freedman M., Larsen M., Wang Z. *A modular functor which is universal for quantum computation*. Comm. Math. Phys. 227(3):605-622, 2002. [arXiv:quant-ph/0001108](https://arxiv.org/abs/quant-ph/0001108).
- Freedman M., Kitaev A., Wang Z. *Simulation of topological field theories by quantum computers*. Communications in Mathematical Physics, 227:587-603, 2002.
- Aharonov D., Jones V., Landau Z. *A polynomial quantum algorithm for approximating the Jones polynomial*. In Proceedings of the 38th ACM Symposium on Theory of Computing, 2006. [arXiv:quant-ph/0511096](https://arxiv.org/abs/quant-ph/0511096).
- Aharonov D., Arad I. *The BQP-hardness of approximating the Jones polynomial*. New Journal of Physics 13:035019, 2011. [arXiv:quant-ph/0605181](https://arxiv.org/abs/quant-ph/0605181).

- Wocjan P., Yard J. *The Jones polynomial: quantum algorithms and applications in quantum complexity theory*. Quantum Information and Computation 8(1/2):147-180, 2008. [arXiv:quant-ph/0603069](https://arxiv.org/abs/quant-ph/0603069).
- Aharonov D., Arad I., Eban E., Landau Z. *Polynomial quantum algorithms for additive approximations of the Potts model and other points of the Tutte plane*. 2007. [arXiv:quant-ph/0702008](https://arxiv.org/abs/quant-ph/0702008).
- Krovi H., Russell A. *Quantum Fourier transforms and the complexity of link invariants for quantum doubles of finite groups*. 2012. [arXiv:1210.1550](https://arxiv.org/abs/1210.1550).
- Shor P. W., Jordan S. P. *Estimating Jones polynomials is a complete problem for one clean qubit*. Quantum Information and Computation, 8(8/9):681-714, 2008. [arXiv:0707.2831](https://arxiv.org/abs/0707.2831).

Инварианты трёхмерных многообразий

Ускорение: свёрхполиномиальное.

Описание: Инвариантом Тураева-Виро называется функция, которая получает на вход трёхмерное многообразие, а возвращает действительное число. Гомеоморфные многообразия на входе дают одно и то же число на выходе. Если на вход квантового компьютера подать трёхмерное многообразие в виде сечения Хеегаарда, то он может эффективно найти определённые аддитивные аппроксимации его инварианта Тураева-Виро, и такая аппроксимация является BQP-полной. Ранее был обнаружен квантовый алгоритм, который за полиномиальное время находит аддитивную аппроксимацию инварианта Виттена-Решитихина-Тураева (ВРТ) для многообразия, представленного сечениями. Возведение в квадрат инварианта ВРТ даёт инвариант Тураева-Виро. Однако на текущий момент неизвестно, является ли задача ВРТ BQP-полной.

- Alagic G., Jordan S., Koenig R., Reichardt B. *Approximating Turaev-Viro 3-manifold invariants is universal for quantum computation*. Physical Review A 82, 040302(R), 2010. [arXiv:1003.0923](https://arxiv.org/abs/1003.0923).
- Garnerone S., Marzulli A., Rasetti M. *Efficient quantum processing of 3-manifold topological invariants*. Advances in Theoretical and Mathematical Physics, 13(6):1601-1652, 2009. [arXiv:quant-ph/0703037](https://arxiv.org/abs/quant-ph/0703037).
- Kauffman L. H., Lomonaco S. J. Jr. *q-deformed spin networks, knot polynomials and anyonic topological quantum computation*. Journal of Knot Theory, Vol. 16, No. 3, pg. 267-332, 2007. [arXiv:quant-ph/0606114](https://arxiv.org/abs/quant-ph/0606114).

Статистическая сумма

Ускорение: свёрхполиномиальное.

Описание: Для классической системы с конечным числом состояний S статистической суммой называется функция $Z = \sum_{s \in S} e^{-E(s)/kT}$, где T — температура, а k — постоянная Больцмана. Существенно то, что любое термодинамическое значение может быть вычислено при помощи получения подходящей частной производной статистической суммы. Статистическая сумма модели Поттса является частным случаем полинома Тутте. Описан квантовый алгоритм для аппроксимации полинома Тутте. После этого даны описания квантовых алгоритмов для вычисления оценок статистических сумм. Эта задача является BQP-полной (энергетические уровни при этом могут быть комплекснозначными). Аппроксимация значения статистической суммы также может быть произведена при помощи симуляции процесса термализации.

- Aharonov D., Arad I., Eban E., Landau Z. *Polynomial quantum algorithms for additive approximations of the Potts model*

- and other points of the Tutte plane.* 2007. [arXiv:quant-ph/0702008](#).
- Lidar D. A. *On the quantum computational complexity of the Ising spin glass partition function and of knot invariants.* New Journal of Physics Vol. 6, pg. 167, 2004. [arXiv:quant-ph/0309064](#).
 - Arad I., Landau Z. *Quantum computation and the evaluation of tensor networks.* SIAM Journal on Computing, 39(7):3089-3121, 2010. [arXiv:0805.0040](#).
 - Van den Nest M., Dür W., Raussendorf R., Briegel H. J. *Quantum algorithms for spin models and simulable gate sets for quantum computation.* Physical Review A, 80:052334, 2009. [arXiv:0805.1214](#).
 - Geraci J. *A new connection between quantum circuits, graphs and the Ising partition function.* Quantum Information Processing, 7(5):227-242, 2008. [arXiv:0801.4833](#).
 - Geraci J., Lidar D. A. *On the exact evaluation of certain instances of the Potts partition function by quantum computers.* Comm. Math. Phys. Vol. 279, pg. 735, 2008. [arXiv:quant-ph/0703023](#).
 - Poulin D., Wocjan P. *Sampling from the thermal quantum Gibbs state and evaluating partition functions with a quantum computer.* Physical Review Letters 103:220502, 2009. [arXiv:0905.2199](#).
 - Wocjan P., Chiang C.-F., Abeyesinghe A., Nagaj D. *Quantum speed-up for approximating partition functions.* Physical Review A 80:022340, 2009. [arXiv:0811.0596](#).

Адиабатические алгоритмы

Ускорение: неизвестно.

Описание: В модели адиабатических квантовых вычислений всё начинается с начального гамильтониана, базовое состояние которого легко подготовить, а потом система медленно эволю-

ционирует к гамильтониану, базовое состояние которого соответствует решению некоторой вычислительной задачи. Исходя из адиабатической теоремы система будет отслеживать мгновенное базовое состояние в том случае, если изменение гамильтониана производится достаточно медленно. Время работы адиабатического алгоритма в худшем случае пропорционально $1/\gamma^3$, где γ — минимальная разница в собственных значениях между базовым состоянием и первым возбуждённым состоянием. Впервые адиабатические квантовые вычисления были предложены Э. Фархи со товарищи в качестве метода для решения NP-полных задач комбинаторной оптимизации. Адиабатические квантовые алгоритмы обычно используют так называемые «стохастические» гамильтонианы, в которых не наблюдается проблем со знаком. Такие алгоритмы иногда называются алгоритмами «квантового отжига». Адиабатические квантовые вычисления с не-стохастическими гамильтонианами являются по мощности такими же, как и обычные квантовые схемы. Адиабатические алгоритмы, использующие стохастические гамильтонианы, являются менее мощными, но в то же время полагаются мощнее классических вычислений. Асимптотическое приближение времени работы адиабатических алгоритмов, тем не менее, очень сложно проанализировать. Также в ранней литературе описывались алгоритмы отжига в классическом смысле, которые симулировались на квантовых компьютерах (отжиг является классическим алгоритмом оптимизации, который симулирует термические процессы). Адиабатические квантовые компьютеры могут выполнять процесс, с какой-то стороны похожий на алгоритм Гровера для неструктурированного поиска. Также показано, что при помощи адиабатических квантовых алгоритмов можно добиться квадратичного ускорения и для более широкого класса задач. В качестве прикладных задач адиабатические квантовые алгоритмы хорошо решают такие задачи, как ранжирование текстов (PageRank), машинное обучение и задачи на графах.

Некоторые квантовые алгоритмы для аппроксимации также используют адиабатическую подготовку состояния.

- Jansen S., Ruskai M.-B., Seiler R. *Bounds for the adiabatic approximation with applications to quantum computation*. Journal of Mathematical Physics, 48:102111, 2007. [arXiv:quant-ph/0603175](https://arxiv.org/abs/quant-ph/0603175).
- Farhi E., Goldstone J., Gutmann S., Sipser M. *Quantum computation by adiabatic evolution*. 2000. [arXiv:quant-ph/0001106](https://arxiv.org/abs/quant-ph/0001106).
- Farhi E., Goldstone J., Gutmann S., Lapan J., Lundgren A., Preda D. *A Quantum Adiabatic Evolution Algorithm Applied to Random Instances of an NP-Complete Problem*. Science, 292(5516):472-475, 2001. [arXiv:quant-ph/0104129](https://arxiv.org/abs/quant-ph/0104129).
- Aharonov D., van Dam W., Kempe J., Landau Z., Lloyd S., Regev O. *Adiabatic Quantum Computation is Equivalent to Standard Quantum Computation*. SIAM Journal on Computing, 37(1):166-194, 2007. [arXiv:quant-ph/0405098](https://arxiv.org/abs/quant-ph/0405098).
- Bravyi S., DiVincenzo D. P., Oliveira R. I., Terhal B. M. *The Complexity of Stoquastic Local Hamiltonian Problems*. Quantum Information and Computation, 8(5):361-385, 2008. [arXiv:quant-ph/0606140](https://arxiv.org/abs/quant-ph/0606140).
- Altshuler B., Krovi H., Roland J. *Anderson localization casts clouds over adiabatic quantum optimization*. Proceedings of the National Academy of Sciences 107(28):12446-12450, 2010. [arXiv:0912.0746](https://arxiv.org/abs/0912.0746).
- Reichardt B. *The quantum adiabatic optimization algorithm and local minima*. In Proceedings of STOC 2004, pg. 502-510.
- Farhi E., Goldstone J., Gutmann S. *Quantum adiabatic evolution algorithms versus simulated annealing*. 2002. [arXiv:quant-ph/0201031](https://arxiv.org/abs/quant-ph/0201031).
- Farhi E., Goldstone J., Gosset D., Gutmann S., Meyer H. B., Shor P. *Quantum adiabatic algorithms, small gaps, and different paths*. Quantum Information and Computation, 11(3/4):181-214, 2011. [arXiv:0909.4766](https://arxiv.org/abs/0909.4766).

- Farhi E., Goldstone J., Gutmann S., Nagaj D. *How to make the quantum adiabatic algorithm fail*. International Journal of Quantum Information, 6(3):503-516, 2008. [arXiv:quant-ph/0512159](https://arxiv.org/abs/quant-ph/0512159).
- Farhi E., Goldstone J., Gutmann S., Nagaj D. *Unstructured randomness, small gaps, and localization*. Quantum Information and Computation, 11(9/10):840-854, 2011. [arXiv:1010.0009](https://arxiv.org/abs/1010.0009).
- Farhi E., Goldstone J., Gutmann S. *Quantum adiabatic evolution algorithms with different paths*. 2002. [arXiv:quant-ph/0208135](https://arxiv.org/abs/quant-ph/0208135).
- van Dam W., Mosca M., Vazirani U. *How powerful is adiabatic quantum computation?* In Proceedings of FOCS 2001, pg. 279-287. [arXiv:quant-ph/0206003](https://arxiv.org/abs/quant-ph/0206003).
- Farhi E., Gosset D., Hen I., Sandvik A. W., Shor P., Young A. P., Zamponi F. *The performance of the quantum adiabatic algorithm on random instances of two optimization problems on regular hypergraphs*. Physical Review A, 86:052334, 2012. [arXiv:1208.3757](https://arxiv.org/abs/1208.3757).
- Finnila A. B., Gomez M. A., Sebenik C., Stenson C., Doll J. D. *Quantum annealing: a new method for minimizing multidimensional functions*. Chemical Physics Letters, 219:343-348, 1994.
- Morita S., Nishimori H. *Mathematical foundation of quantum annealing*. Journal of Mathematical Physics, 49(12):125210, 2008.
- Roland J., Cerf N. J. *Quantum search by local adiabatic evolution*. Physical Review A, 65(4):042308, 2002. [arXiv:quant-ph/0107015](https://arxiv.org/abs/quant-ph/0107015).
- Somma R. D., Boixo S. *Spectral gap amplification*. SIAM Journal on Computing, 42:593-610, 2013. [arXiv:1110.2494](https://arxiv.org/abs/1110.2494).
- Szegedy M. *Quantum speed-up of Markov chain based algorithms*. In Proceedings of the 45th IEEE Symposium on Foundations of Computer Science, pg. 32, 2004.

- Garnerone S., Zanardi P., Lidar D. A. *Adiabatic quantum algorithm for search engine ranking*. Physical Review Letters 108:230506, 2012.
- Pudenz K. L., Lidar D. A. *Quantum adiabatic machine learning*. Quantum Information Processing, 12:2027, 2013. [arXiv:1109.0325](https://arxiv.org/abs/1109.0325).
- Neven H., Denchev V. S., Rose G., Mcready W. G. *Training a binary classifier with the quantum adiabatic algorithm*. 2008. [arXiv:0811.0416](https://arxiv.org/abs/0811.0416).
- Gaitan F., Clark L. *Ramsey numbers and adiabatic quantum computing*. Physical Review Letters, 108:010501, 2012. [arXiv:1103.1345](https://arxiv.org/abs/1103.1345).
- Gaitan F., Clark L. *Graph isomorphism and adiabatic quantum computing*. 2013. [arXiv:1304.5773](https://arxiv.org/abs/1304.5773).

Z-функции

Ускорение: сверхполиномиальное.

Описание: Пусть $f(x, y)$ является полиномом степени d над конечным полем \mathbb{F}_p . Пусть N_r является числом проективных решений уравнения $f(x, y) = 0$ на расширенном поле \mathbb{F}_{p^r} . Тогда Z-функция для f определяется как $Z_f(T) = \exp(\sum_{r=1}^{\infty} \frac{N_r}{r} T^r)$. Примечательно то, что $Z_f(T)$ всегда имеет форму $Z_f(T) = \frac{Q_f(T)}{(1-pT)(1-T)}$, где $Q_f(T)$ представляет собой полином степени $2g$, где $g = \frac{1}{2}(d-1)(d-2)$, и этот полином называется родом полинома f . Если задана Z-функция, то при помощи неё можно легко вычислить количество нулей полинома f на произвольном расширенном поле \mathbb{F}_{p^r} . Также можно определить подобную Z-функцию и в том случае, если первоначальное поле, над которым построен полином f , не имеет первичного порядка. Показано, что квантовый компьютер может определить Z-функцию рода g над конечным полем \mathbb{F}_{p^r}

за время $\text{poly}(\log p, r, g)$. Все самые эффективные классические алгоритмы являются экспоненциальными либо по $\log(p)$, либо по g . Также показано, что из-за связи между нулями римановой Z -функции и собственными значениями некоторых квантовых операторов, квантовые компьютеры могут эффективно аппроксимировать количество решений уравнений в конечных полях.

- Kedlaya K. S. *Quantum computation of zeta functions of curves*. Computational Complexity, 15:1-19, 2006. [arXiv:math/0411623](https://arxiv.org/abs/math/0411623).
- van Dam W. *Quantum computing and zeros of zeta functions*. 2004. [arXiv:quant-ph/0405081](https://arxiv.org/abs/quant-ph/0405081).

Весовые перечислители

Ускорение: сверхполиномиальное.

Описание: Пусть C является кодом из n бит, то есть представляет собой подмножество \mathbb{Z}_2^n . Весовым перечислителем для C называется значение $S_C(x, y) = \sum_{c \in C} x^{|c|} y^{n-|c|}$, где $|c|$ обозначает вес Хэмминга для c . Весовые перечислители имеют множество способов применения в теории кодирования информации. Если код C является линейным, то он может быть определён как $C = \{c: Ac = 0\}$, где A является матрицей над \mathbb{Z}_2 . В этом случае $S_C(x, y) = \sum_{c: Ac=0} x^{|c|} y^{n-|c|}$. Квадратичные перечислители веса являются обобщением обычных, которое вводится следующим образом:

$S(A, B, x, y) = \sum_{c: Ac=0} (-1)^{c^x Bc} x^{|c|} y^{n-|c|}$. Теперь необходимо рассмотреть следующий специальный случай. Пусть A является квадратной матрицей размера $n \times n$ над \mathbb{Z}_2 такой, что $\text{diag}(A) = I$. Пусть $\text{lwtr}(A)$ обозначает нижнюю треугольную матрицу, полученную из матрицы A сбросом всех её элементов, расположенных выше главной диагонали, в значение 0. Пусть l и k являются положительными целыми числами. Если есть уверенность, что $|S(A, \text{lwtr}(A), k, l)| \geq \frac{1}{2}(k^2 + l^2)^{n/2}$, то задача

определения знака выражения $S(A, \text{lwtr}(A), k, l)$ является BQP-полной. Оценка значения квадратичных весовых перечислителей близка к вычислению статистических сумм по моделям Поттса и Изинга.

- Knill E., Laflamme R. *Quantum computation and quadratically signed weight enumerators*. Information Processing Letters, 79(4):173-179, 2001. [arXiv:quant-ph/9909094](https://arxiv.org/abs/quant-ph/9909094).
- Lidar D. A. *On the quantum computational complexity of the Ising spin glass partition function and of knot invariants*. New Journal of Physics Vol. 6, pg. 167, 2004. [arXiv:quant-ph/0309064](https://arxiv.org/abs/quant-ph/0309064).
- Geraci J. *A new connection between quantum circuits, graphs and the Ising partition function*. Quantum Information Processing, 7(5):227-242, 2008. [arXiv:0801.4833](https://arxiv.org/abs/0801.4833).
- Geraci J., Bussell F. V. *A theorem on the quantum evaluation of weight enumerators for a certain class of cyclic Codes with a note on cyclotomic cosets*. 2007. [arXiv:cs/0703129](https://arxiv.org/abs/cs/0703129).

Симуляция отжига

Ускорение: полиномиальное.

Описание: В задаче симуляции отжига имеется ряд цепей Маркова, которые заданы стохастическими матрицами M_1, M_2, \dots, M_n . Эти матрицы отличаются друг от друга в серии очень медленно, так что их ограниченные распределения $\pi_1, \pi_2, \dots, \pi_n$ удовлетворяют ограничению $|\pi_{t+1} - \pi_t| < \varepsilon$ для произвольного малого значения ε . Эти распределения могут быть симитированы температурными распределениями при успешно низких температурах. Если состояние π_1 легко подготовить, то можно при помощи применения этой цепи Маркова можно прийти к состоянию π_n . Обычно есть желание того, чтобы π_n было решением какой-либо задачи оптимизации. Пусть δ_i является разницей между наибольшим и вторым

наибольшим собственными значениями матрицы M_i . Пусть $\delta = \min_i \delta_i$. Время работы классического алгоритма для этой задачи пропорционально значению $1/\delta$. Показано, что на квантовом компьютере эту задачу можно решить за время, пропорциональное значению $1/\sqrt{\delta}$.

- Szegedy M. *Spectra of Quantized Walks and a $\sqrt{\delta\epsilon}$ rule*. 2004. [arXiv:quant-ph/0401053](https://arxiv.org/abs/quant-ph/0401053).
- Szegedy M. *Quantum speed-up of Markov chain based algorithms*. In Proceedings of the 45th IEEE Symposium on Foundations of Computer Science, pg. 32, 2004.
- Somma R. D., Boixo S., Barnum H. *Quantum simulated annealing*. 2007. [arXiv:0712.1008](https://arxiv.org/abs/0712.1008).
- Somma R. D., Boixo S., Barnum H., Knill E. *Quantum simulations of classical annealing*. Physical Review Letters 101:130504, 2008. [arXiv:0804.1571](https://arxiv.org/abs/0804.1571).

Перезапись строк

Ускорение: сверхполиномиальное.

Описание: Перезапись строк является довольно общим способом осуществления вычислений. Системы перезаписи строк (иногда называемые «грамматиками») определяются при помощи набора правил, которые позволяют заменять одни подстроки другими. Например, контекстно-свободные грамматики эквивалентны автоматам с магазинной памятью. Показано, что некоторые задачи по перезаписи строк обещают быть BQP-полными. Это значит, что на квантовых компьютерах их можно решить за полиномиальное время, а классические компьютеры, вероятно, за такое время не справятся. Пусть даны три строки s , t и t' , а также набор перезаписывающих правил, для которого обещано, что он удовлетворяет некоторым требованиям, задачей является найти определённую аппроксимацию разницы между количеством способов получения строки t

из строки s и количеством способов получения строки t' из строки s . Похожим образом задачи аппроксимации разницы между количеством путей от одной до другой вершины графа и разницы между вероятностями перехода из одного состояния в другое в случайном блуждании также являются BQP-полными.

- Janzing D., Wocjan P. *A promiseBQP-complete string rewriting problem*. Quantum Information and Computation, 10(3/4):234-257, 2010. [arXiv:0705.1180](https://arxiv.org/abs/0705.1180).
- Janzing D., Wocjan P. *BQP-complete problems concerning mixing properties of classical random walks on sparse graphs*. 2006. [arXiv:quant-ph/0610235](https://arxiv.org/abs/quant-ph/0610235).

Степени матриц

Ускорение: сверхполиномиальное.

Описание: У квантовых компьютеров есть экспоненциальное преимущество в аппроксимации элементов матриц, являющихся степенями экспоненциально больших разреженных матриц. Пусть есть симметричная матрица размера $N \times N$, в которой в каждой строке имеется $\text{polylog}(N)$ ненулевых элементов. Для заданного номера строки эффективно можно получить список ненулевых элементов. Задачей является аппроксимировать i -ый диагональный элемент $(A^m)_{ii}$ матрицы A^m для произвольного $1 < i < N$ и m , которое зависит от N полилогарифмично. Аппроксимация аддитивна по значению $b^m \varepsilon$, где b — заданная верхняя оценка значения $|A|$, а ε имеет порядок $1/\text{polylog}(N)$. Показано, что эта задача является BQP-полной, поэтому квантовый компьютер может решить её за полиномиальное время, в то время как классические алгоритмы, возможно, нет.

- Janzing D., Wocjan P. *A simple promiseBQP-complete matrix problem*. Theory of Computing, 3:61-79, 2007. [arXiv:quant-ph/0606229](https://arxiv.org/abs/quant-ph/0606229).

Краткие выводы

Приведённое описание квантовых алгоритмов, упоминания и демонстрация которых имеется на сегодняшний день в научной литературе, показывает, что данная отрасль знания не стоит на месте и не пребывает в стагнации, как это может показаться некоторым недоброжелателям. Даже несмотря на то, что большинство из представленных алгоритмов имеют глубоко научный характер и относятся к очень абстрактным областям знаний, уже сегодня имеются и другие алгоритмы, которые решают очень прикладные задачи. В частности, уже неоднократно упомянутые алгоритмы Шора позволяют скомпрометировать несколько современных криптографических систем, и сегодняшний день считающихся «хорошими». В частности это криптография RSA и протокол обмена ключами Диффи-Хеллмана.

Да и другие алгоритмы позволяют решать многочисленные прикладные задачи. Неструктурированный поиск, квантовое блуждание, нахождение глобального минимума и т. д. — все эти задачи уже являются более прикладными, нежели теоретическими. Но даже если рассмотреть и такие алгоритмы, как работа с идеалами в рамках теории групп, то их фундаментальность и глубина даёт основание предполагать, что они лягут в основу многих алгоритмов для решения прикладных задач. Именно через это прошла традиционная информатика, когда сначала учёные разрабатывали непонятные алгоритмы с совершенно неясными структурами данных, а уже потом прикладные программисты реализовывали программное обеспече-

На представленной диаграмме прямоугольником обозначается квантовый алгоритм или задача, которая может быть решена в рамках модели квантовых вычислений. Жирная граница использована для тех алгоритмов, которые описаны в данной книге.

Для каждого алгоритма и задачи приводятся две характеристики — ускорение и типы задач, решаемых алгоритмом (или тип самой задачи). Ускорение алгоритма может быть:

- Константное (с).
- Полиномиальное (р).
- Сверхполиномиальное (sp).
- Экспоненциальное (е).
- Неизвестное (?) — для адиабатических квантовых алгоритмов.

Все алгоритмы и задачи классифицированы по следующим типам задач (при этом у алгоритма вполне может быть несколько типов):

- Теория чисел.
- Теория множеств.
- Теория групп.
- Алгоритмы на графах.
- Алгоритмы на матрицах.
- Алгоритмы поиска.
- Алгоритмы оптимизации.
- Алгоритмы о свойствах функций.
- Криптографические алгоритмы и задачи.
- Алгоритмы квантовой симуляции.

Наконец, между алгоритмами и (или) задачами рассматриваются четыре типа отношений, а именно:

- Если из алгоритма А идёт обычная стрелка в алгоритм В, то это значит, что алгоритм В основан на алгоритме А.
- Если же стрелка пунктирная, то алгоритм В сводится к алгоритму А, то есть это в большей мере отношение подобия.
- Волнистая стрелка обозначает, что алгоритм А решает прикладную задачу В.
- Наконец, двойная линия между прямоугольниками без стрелок обозначает, что два алгоритма сходны между собой.

Как видно, на сегодняшний день квантовая модель вычислений уже является довольно развитой областью знаний. Более того, в научные исследования по данному вопросу вовлечены многие лучшие умы в области как физики, так и информатики (компьютерных наук). Количество публикаций растёт день ото дня. Ищутся новые алгоритмы и способы их применения к решению прикладных задач.

Так что остаётся только отметить, что всем тем, кто хочет всерьёз заниматься квантовыми вычислениями, уже сегодня надо полноценно погружаться в эту область и изучать фундаментальные основы. В следующем приложении приводится обзор литературы, и каждый сможет для себя решить и выбрать те книги и другие источники знаний по теме, с которыми необходимо ознакомиться в ближайшее время.

Ну и хотел бы отметить, что нарисовать приведённую на рис. 31 диаграмму отношений квантовых алгоритмов мне помог добрый коллега С. Деревягин.

Обзор литературы о квантовых вычислениях

Для того чтобы двигаться дальше, у читателя должна появиться «дорожная карта». Ведь на текущий момент даже на русском языке уже существует несколько десятков источников, посвящённых модели квантовых вычислений, и в них можно запутаться.

Во-первых, разные источники посвящены разным аспектам этой новой области знаний. Где-то рассказывается про математические основы модели. В каких-то книгах упор делается на перспективные физические разработки и поиск методов реализации математической модели в «железе». Ну а есть вообще такие источники, в которых собрана «сборная солянка» из всего, что удалось найти автору, всё это бессистемно перемешано и сдобрено к тому же какими-то комментариями, которые часто оказываются ещё и неграмотны.

Также есть источники, в которых вся тема разжёвывается с самых основ (к примеру, приводятся элементы линейной алгебры), а в других читатель с первых страниц погружается

в пучину глубокого и тяжёлого «матана», понимание которого он должен был бы получить с молоком матери.

В этом приложении я приведу описания всех источников информации на русском языке, посвящённых модели квантовых вычислений, которые были опубликованы до момента написания этой книги. Такой список описаний поможет читателю сформировать план дальнейшего изучения предмета.

Далее все книги описываются по унифицированной форме, представляющей собой набор реквизитов. Реквизиты «Оценка» и «Полезность» являются числами от 1 (худшая) до 5 (лучшая), при этом значение реквизита «Полезность» состоит из трёх чисел, которые оценивают полезность информационную, идейную и прикладную. Информационная полезность говорит о том, сколько новой информации можно почерпнуть в источнике. Идейная полезность, как полагается, оценивает количество новых идей, заложенных в книге или статье. А полезность прикладная обозначает то, насколько книга применима на практике.

Итак, вот список источников...

Классические и квантовые ветвящиеся программы

Авторы: Аблаев Ф. М., Васильев А. В.

Рецензия: Немного странная статья, в которой рассматривается один класс программ в схемном представлении, а также показывается, как это схемное представление может быть расширено до квантовой вычислительной модели. Прикладного аспекта в этой работе немного, равно как и теоретические основы не рассматриваются вовсе. Похоже, что авторы просто продолжают описание каких-то своих изысканий, которые

имеют мало прикладного значения, но интересуют авторов вполне.

Краткое содержание: Статья представляет описание модели квантовых вычисления для одного класса программ. Содержание: вычисления в модели ветвящихся программ, методы построения эффективных квантовых алгоритмов и нижние оценки сложности их представления, математическое описание класса задач, эффективно решаемых квантовыми ветвящимися программами.

Оценка: 3.

Полезность: 2 2 3.

Количество страниц: 37.

Категории: Квантовые вычисления, Программирование, Разработка.

Ссылка на скачивание: <http://www.twirpx.com/file/1378540/>

Ссылка на цитаты из источника: <http://goo.gl/HkNmQ>

Физика квантовой информации. Квантовая криптография. Квантовая телепортация. Квантовые вычисления

Авторы: Бауместер Д., Экерт А., Цайлингер А.

Рецензия: Это сборник статей от примерно 40 авторов, которые работают в различных научных лабораториях по всему миру в рамках одной программы по развитию квантовых вычислений (всё больше на физическом уровне). Как и в любом сборнике статей в данном издании намешано в кучу всё, хотя редакторы, конечно же, постарались сделать более или менее связное изложение. В итоге мы имеем довольно сильно наполненную матаном книгу, в которой вводными являются первые три статьи, а остальные рассматривают физические аспекты

и различные эксперименты, которые были проведены. К слову, в книге много интересных иллюстраций.

Краткое содержание: В издании представлены следующие статьи:

1. Физика квантовой информации: основные понятия
2. Квантовая криптография
3. Квантовая плотная кодировка и квантовая телепортация
4. Концепция квантовых вычислений
5. На подступах к квантовым вычислениям
6. Квантовые сети и многочастичное перепутывание
7. Декогерентность и квантовое исправление ошибок
8. Очищение перепутывания

Оценка: 3.

Полезность: 4 3 2.

Количество страниц: 376.

Категории: Квантовая механика, Квантовые вычисления, Сборник статей, Физика.

Ссылка на скачивание: <http://www.twirpx.com/file/352844/>

Ссылка на цитаты из источника: <http://goo.gl/XUaZ9G>

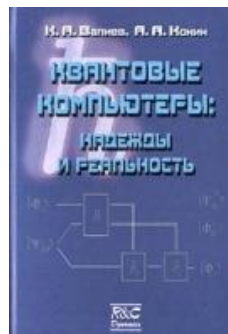
Квантовые компьютеры. Надежды и реальность

Авторы: Валиев К. А., Кокин А. А.

Рецензия: В очередной раз убеждаюсь в том, что подобные книги пишутся людьми, которые совершенно оторваны от реальности (сидят в «башне из слоновой кости»). Парадокс очевиден — чтобы прочитать эту книгу необходимо быть знатоком квантовых вычислений. Но если ты знаток квантовых вычислений, то тебе уже бессмысленно читать эту книгу. Данный образец чуть более чем полностью наполнен громоздкими

математическими формулами без каких-либо объяснений, причём эти формулы охватывают всё — от теории функций комплексного переменного до ядерной физики. Читать очень сложно.

Краткое содержание: В книге семь глав, из которых только две посвящены собственно модели квантовых вычислений — первая и вторая. В первой рассматриваются самые небольшие основы теории вычислений, во второй — пара алгоритмов (поиск и уже замученный алгоритм факторизации Шора). Затем пять глав посвящены обсуждению различных физических способов реализации квантовых компьютеров: ионы в ловушках, Жидкостные ядерные магнитно-резонансные, твердотельные ЯМР, твердотельные на квантовых точках и на сверхпроводниковых элементах.



Оценка: 2.

Полезность: 1 0 1.

Количество страниц: 320.

Категории: Квантовые вычисления, Теория вычислений, Теория информации.

Ссылка на скачивание: <http://www.twirpx.com/file/166848/>

Ссылка на цитаты из источника: <http://goo.gl/12A4NJ>

Квантовые вычисления для программистов

Автор: Васильев А. В.

Рецензия: Продолжаю знакомиться с моделью квантовых вычислений и литературой на русском языке, которая к настоящему моменту есть на эту тему. Перво-наперво

в очередной десятке решил ознакомиться со статьёй, которая заинтриговала меня своим названием. Что ж, статья весьма достойна — в ней кратко и ёмко отражены все особенности модели квантовых вычислений, которые требуются разработчику программного обеспечения. Хотя ни на один вопрос ответа не дано, поскольку статья является больше обзорной, ознакомиться с ней полезно.

Краткое содержание: Статья состоит из следующих разделов:

- Введение
- Математический аппарат квантовых вычислений. Линейная алгебра
- Квантовые модели вычислений
- Квантовые вычисления для программистов
- Список литературы

Оценка: 4.

Полезность: 5 3 3.

Количество страниц: 17.

Категории: Квантовые вычисления, Программирование, Разработка.

Ссылка на скачивание: <http://www.twirpx.com/file/1378539/>

Ссылка на цитаты из источника: <http://goo.gl/vj9Edt>

Квантовый вызов. Современные исследования оснований квантовой механики

Авторы: Гринштейн Дж., Зайонц А.

Рецензия: Довольно занятая книга по основам квантовой механики, в которой лишь последняя глава затрагивает тему квантовых вычислений и квантового компьютера (и то очень

поверхностно, рассматривается только простейший вариант алгоритма Дойча). В ней авторы попробовали самыми простыми словами описать то, что сегодня собой представляет формализм квантовой механики, какие нерешённые проблемы существуют, куда стоит двигаться. Книга живая, в ней описано множество интереснейших экспериментов. Однако опять необходимо отметить, что книга написана физиками для физиков. Для того чтобы её читать, необходимо владеть базовыми знаниями в области квантовой механики. Кроме того, в русском переводе две последние главы написаны редакторами перевода, и они больше о «советских физиках Ландау и Лифшице», чем о квантовой механике).



Краткое содержание: Основные достижения физики XX века связаны с квантовой механикой. Однако выдающийся физик Ричард Фейнман утверждал, что никто не понимает квантовую механику. Попытки вдуматься в это утверждение и работы Дж. Белла привели к появлению квантовых вычислений и квантовой криптографии. Авторы раскрывают смысл высказывания Р. Фейнмана наиболее подходящим для этого способом. На примере экспериментов, сделанных в последние годы, они показывают потрясающую парадоксальность квантового мира. Новейшие экспериментальные результаты и фундаментальные проблемы квантовой физики изложены в предельно доходчивой форме. Дополнение редакторов перевода развивает основную линию книги, проясняя логическую структуру споров на переднем крае науки. Оно также знакомит с результатами, опубликованными уже после выхода в США второго издания.

Оценка: 4.

Полезность: 5 4 3.

Количество страниц: 432.

Категории: Квантовая механика, Квантовые вычисления.

Ссылка на скачивание: <http://www.twirpx.com/file/960776/>

Ссылка на цитаты из источника: <http://goo.gl/94CJEY>

Основы квантовой кибернетики

Автор: Гуц А. К.

Рецензия: Занятная книга, в которой в одну кучу смешаны вопросы теории вычислимости, философии сознания, кибернетики, квантовой механики, биографии некоторых выдающихся личностей и некоторые другие темы. Формул мало, тема объясняется довольно просто. Впрочем, глубокого погружения в вопрос нет, из квантовых алгоритмов только кратко описываются алгоритм Шора для факторизации и алгоритм поиска. Если кто-то хочет понять, что такое квантовые алгоритмы, и при этом познакомиться с оригинальными воззрениями автора на природу сознания, то книга рекомендуется к чтению.

Краткое содержание: В книге излагаются некоторые разделы квантовой кибернетики. Это теория квантового компьютера и квантовых вычислений, элементы квантовой теории информации. Предлагается квантовая модель осознания. Описывается попытка С. Хамероффа и Р. Пенроуза представить сущность сознания с точки зрения квантовой механики. Рассказывается о проблемах квантовой психопатологии и приводятся эскизы квантовой теории времени.

Оценка: 4.

Полезность: 3 3 0.

Количество страниц: 204.

Категории: Квантовые вычисления, Кибернетика, Сознание.

Ссылка на скачивание: <http://www.twirpx.com/file/1109456/>

Ссылка на цитаты из источника: <http://goo.gl/I8pIH3>

Алгоритмы

Авторы: Дасгупта С., Пападимитриу Х., Вазирани У.

Рецензия: Зачётная книга трёх довольно известных в узких кругах авторов, которые рассматривают различные и разнообразные алгоритмы, начиная с самых азов и заканчивая алгоритмом факторизации Шора. Книга интересна своим не очень стандартным подходом. Она учит фундаментальным вещам, при этом нет никаких факториалов и быстрых сортировок. Очень доходчиво объясняются основы теории сложности вычислений. Объясняется сама суть вычислений, почему они выполняются так, а не иначе. Очень интересны главы про алгоритмы на графах, а также главы про динамическое и линейное программирование. В последней главе приводится описание алгоритма факторизации Шора (похоже, что это единственная глава, которую писал У. Вазирани; и она полностью повторяет конспекты его лекций по квантовым вычислениям). Алгоритм описан так, что в принципе можно понять, как он работает, хотя реализовать его в коде вряд ли получится без дополнительного чтения. В общем, очень рекомендую эту книгу.

Краткое содержание: В этой книге, предназначенной для студентов математических и программистских специальностей (начиная с младших курсов), подробно разбираются основные методы построения и анализа эффективных алгоритмов. Она основана на лекциях авторов в университетах Сан-Диего и Беркли. Выбор материала не вполне стандартный (скажем, о сортировке и структурах данных, связанных

с хранением упорядоченных множеств в сбалансированных деревьях, не говорится, зато обсуждаются линейное программирование и даже квантовые вычисления). Авторы старались выделить основные идеи и излагать доказательства наглядно, не злоупотребляя формализмом, но и не жертвуя математической строгостью; оригинальный подход авторов делает книгу интересной не только студентам, но и опытным преподавателям. Каждый раздел снабжён упражнениями.

Оценка: 5.

Полезность: 5 2 4.

Количество страниц: 319.

Категории: Алгоритмы, Квантовые вычисления.

Ссылка на скачивание: <http://www.twirpx.com/file/1416654/>

Ссылка на цитаты из источника: <http://goo.gl/dYsTmk>

Сознание и квантовые компьютеры

Автор: Иванов Е. М.

Рецензия: Когда философы начинают рассуждать на темы, которые очень далеко находятся от «чистой философии», получаются вот такие статьи. Очень прискорбно читать подобное, особенно когда автором затрагивается очень важная и насущная тема, но при этом сам автор откровенно «плавает» в технических вопросах, из-за чего получаются страшнейшие ляпы в тексте, которые не должны быть в источниках подобного рода. В итоге я насчитал на 17 страницах статьи не менее 5 фактических ошибок, а сам текст настолько затуманен и запутан, что даже коллеги автора по философскому цеху вряд ли смогут понять всё то, что хотел выразить автор, не ознакомившись сперва с массой источников по квантовой механике и квантовым вычислениям.

Краткое содержание: В своей статье автор рассматривает так называемый «двухаспектный подход» к пониманию человеческого сознания, после чего проводит прямые аналогии с квантовым вычислительным устройством.

Оценка: 2.

Полезность: 3 3 0.

Количество страниц: 17.

Категории: Сознание, Философия.

Ссылка на скачивание: <http://www.twirpx.com/file/1007805/>

Ссылка на цитаты из источника: <http://goo.gl/5k3Oy1>

Введение в квантовые вычисления

Авторы: Каие Ф., Лафлам Р., Моска М.

Рецензия: Очень сильный учебник, но опять страдающий той же самой болезнью, что и многие иные материалы по квантовым вычислениям, которые к этому времени я уже прочитал. Всё дело в том, что эту книгу опять же надо читать после того, как уже читатель изучил квантовые вычисления. По крайней мере, жёстким матаном она наполнена практически полностью. Формулы на каждой странице, причём многоэтажные. Однако надо отметить, что в этой книге даётся не только описание всех базовых тем, но и предлагаются к вниманию читателя квантовые алгоритмы в числе больше трёх. Начиная с алгоритмов Шора, Гровера и Дойча, далее производится переход к алгоритму Саймона, а потом рассматриваются совсем уж заковыристые задачи, ос-



нованные на задаче о скрытой подгруппе. Этим учебник очень выгодно отличается от многих других источников.

Краткое содержание: Книга последовательно раскрывает модель квантовых вычислений, обращаясь, при необходимости, к смежным вопросам. В том числе:

1. Введение в основные понятия.
2. Линейная алгебра и дираковская система обозначений.
3. Кубиты и концепции квантовой механики.
4. Квантовая модель вычислений.
5. Сверхплотное кодирование и квантовая телепортация.
6. Введение в квантовые алгоритмы.
7. Алгоритмы со сверхполиномиальным ускорением.
8. Алгоритмы, основанные на усилении амплитуды.
9. Квантовая теория вычислительной сложности.
10. Исправление квантовых ошибок.

Оценка: 3.

Полезность: 4 2 2 .

Количество страниц: 338.

Категории: Квантовые вычисления, Линейная алгебра, Учебник.

Ссылка на скачивание: <http://www.twirpx.com/file/942457/>

Ссылка на цитаты из источника: <http://goo.gl/0BmzR9>

Квантовая информация

Автор: Килин С. Я.

Рецензия: Небольшая статья про квантовые вычисления и квантовую теорию информацию. Статья интересна тем, что написана крайне простым языком, для её чтения в принципе не надо знать квантовую механику. Однако

в статье, само собой разумеется, нет никаких определения (так что кое-что придётся, всё-таки, знать). Ещё хорошо тем, что в статье много поясняющих картинок и диаграмм, что очень способствует пониманию темы. Однако неприятно поразила статья тем, что раздел про квантовые вычисления слово в слово переписан из другого источника («Квантовые компьютеры: надежды и реальность»).

Краткое содержание: В статье рассматриваются такие темы: основополагающая работа Э. Шрёдингера по квантовой механике, квантовая телепортация, квантовая криптография, квантовые вычисления и компьютеры, проблема декогеренции.

Оценка: 4.

Полезность: 3 4 3.

Количество страниц: 21.

Категории: Квантовые вычисления, Теория вычислений.

Ссылка на скачивание: <http://www.twirpx.com/file/665639/>

Ссылка на цитаты из источника: <http://goo.gl/5FzwEH>

Классические и квантовые вычисления

Авторы: Китаев А., Шень А., Вялый М.

Рецензия: Что можно сказать по этой книге? Ничего хорошего. Чтобы её прочитывать, необходимо знать суть модели квантовых вычислений, при этом она позиционируется как вводная в тему. Это странная позиция многих авторов от науки — чтобы прочитывать базовую, вводную книгу, необходимо глубоко знать предмет. Но если ты глубоко знаешь предмет, то читать вводную книгу тебе не интересно. В итоге для меня книга оказалась в целом бесполезная. Разбираться в дебрях формул мне не очень хотелось, а то, что написано без формул, я и так знал. Ну и авторы, к слову, в некоторых вопросах сами путались,

а многие темы вообще описывали так, как будто бы брали материалы из зарубежных курсов (да хоть с той же Coursera) по криптографии и теории автоматов. Вычислительная теория сложности чуть ли не полностью взята у Джеффри Улльмана. Поэтому сложно дать высокую оценку книге — в целом она бесполезна, лучше пользоваться оригиналами. Ну а ежели собираешься переработать хорошие, годные источники для получения популярной книги, то и, стало быть, писать надо популярно. Учитывая то, что книга написана по гранту РФФИ, в очередной раз наступает печаль за нашу фундаментальную науку.

Краткое содержание: Вначале приводится краткое введение в классическую теорию сложных вычислений. Затем подробно излагаются основы теории квантовых вычислений, включая описания известных квантовых алгоритмов.

Оценка: 2.

Полезность: 3 2 1.

Количество страниц: 192.

Категории: Квантовые вычисления, Теория сложности.

Ссылка на скачивание: <http://www.twirpx.com/file/344199/>

Ссылка на цитаты из источника: <http://goo.gl/asB5yK>

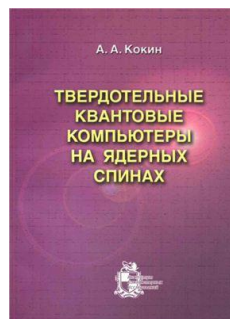
Твердотельные квантовые компьютеры на ядерных спинах

Автор: Кокин А. А.

Рецензия: Для того чтобы быть в курсе темы физической реализации квантовых компьютеров, можно ознакомиться с этой книгой. В принципе, если кто-то из читателей интересуется вопросами физической реализации модели квантовых вычислений, то эту книгу можно посоветовать в качестве основного ис-

точника информации. В ней кратко рассказывается про физические основы квантовых вычислений, а потом очень много информации даётся про технологию ядерно-магнитного резонанса, которая видится очень перспективной. Эта технология рассматривается в разных вариантах, приводятся плюсы и минусы, делаются заключения по применимости. В общем, книга советуется к ознакомлению, но не всем.

Краткое содержание: Детально рассматриваются физические основы ЯМР квантового компьютера, общие требования, предъявляемые к полномасштабному квантовому компьютеру, к многоспиновой системе, представляющей элементную базу квантового компьютера. Описываются принципы построения ЯМР квантового компьютера и способы организации квантовых логических операций. Даётся обзор предложенных твердотельных вариантов ЯМР квантовых компьютеров, анализируются преимущества и недостатки отдельных вариантов и оценивается их перспективность. Основное внимание уделяется полупроводниковому ансамблевому варианту с полосковыми затворами на ядерных спинах донорных атомов фосфора и варианту ансамблевого ЯМР квантового компьютера на основе антиферромагнитной структуры с использованием принципов клеточного автомата.



Оценка: 3.

Полезность: 4 1 2.

Количество страниц: 204.

Категории: Квантовая механика, Квантовые вычисления, Физика.

Ссылка на скачивание: <http://www.twirpx.com/file/587850/>

Ссылка на цитаты из источника: <http://goo.gl/54FQGq>

Введение в теорию квантовых вычислений (методы квантовой механики в кибернетике). Книга 1

Авторы: Кулик С. Д., Берков А. В., Яковлев В. П.

Рецензия: Методическое пособие (или, всё-таки, книга?), выпущенное в МИФИ, чем и привлекло моё внимание. Похоже, что эту книгу написали трое аспирантов (если это не так, прошу прощения), которые посадили за компьютеры своих «падаванов» из числа студентов-третьекурсников, которые насобирали из различных источников кучу материала, кое-как структурировали и снабдили огромным количеством ссылок на первоисточники. В итоге получилась жуткая смесь ликбеза с совершенно непонятными без обращения к дополнительной литературе пассажами по квантовой механике и квантовым вычислениям. И особенно доставило постоянное использование оборотов: «Специалисты доказали», «Специалисты обнаружили» и т. д. Что ж, посмотрим, что будет во второй книге.

Краткое содержание: В книге даётся довольно подробное описание следующих тем и вопросов:

1. Теория информации.
2. Алгебра Буля и системотехника.
3. Аналоговые вычисления (есть очень интересные примеры).
4. Теория вероятности и диаграммная техника.
5. Классические и квантовые алгоритмы.
6. Описание квантовых компьютеров.

Оценка: 3.

Полезность: 3 3 2.

Количество страниц: 212.

Категории: Квантовые вычисления, Логика, Системотехника, Теория алгоритмов, Теория вероятности, Теория вычислений, Теория информации.

Ссылка на скачивание: <http://www.twirpx.com/file/806435/>

Ссылка на цитаты из источника: <http://goo.gl/bZgRqC>

Введение в теорию квантовых вычислений (методы квантовой механики в кибернетике). Книга 2

Авторы: Кулик С. Д., Берков А. В., Яковлев В. П.

Рецензия: Это очень странно, но книга поразительно отличается от предыдущей. Прочитав две трети оной, я подумал, что, возможно, это лучший учебник по теме, написанный отечественными авторами — в некоторых местах было невозможно оторваться, а тема изложена крайне доступно (хотя, возможно, у меня уже количество прочитанной литературы на тему перешло в качество). Однако четвёртый раздел книги, посвящённый схемотехнике квантовых алгоритмов, больше похож на методическое пособие, и, похоже, его писал тот же человек, что и полностью предыдущую книгу. Опять такие выражения как «специалисты установили», «отметим», «важно» и т. д., а сама часть полностью состоит из решения задач. А тему четвёртой части, если откровенно, можно было раскрыть на пяти страницах. В общем, смазанное впечатление.

Краткое содержание: В книге очень подробно рассматриваются следующие темы:

1. Квантовая механика (математические основы, формализм)
2. Квантовая теория информации
3. Квантовые алгоритмы
4. Схемотехника

Оценка: 4.

Полезность: 5 3 3.

Количество страниц: 532.

Категории: Квантовая механика, Квантовые вычисления, Схемотехника.

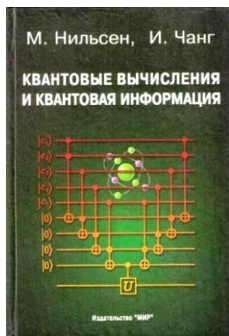
Ссылка на скачивание: <http://www.twirpx.com/file/806433/>

Ссылка на цитаты из источника: <http://goo.gl/bZgRqC>

Квантовые вычисления и квантовая информация

Авторы: Нильсен М., Чанг И.

Рецензия: Хорошая книга по теме квантовых вычислений. В принципе, вдумчиво прочитав её, можно получить всю необходимую информацию по этому направлению исследований, и она в то же время совершенно легка для понимания, несмотря на обилие формул. Книга позиционируется в качестве вводного учебника, поэтому все вопросы в ней рассматриваются так, чтобы у читающего не было дискомфорта от непонимания темы. Все необходимые определения есть тут, все нотации и понятия описываются в начале.



В целом, можно прочитать первую главу, чтобы осуществить первый вход в тему. Все остальные главы — это уже более детальное и более математичное описание положений, данных в первой главе. Ну и вообще, книга вполне может использоваться в качестве учебника для двухсеместрового курса по квантовым вычислениям и (или) квантовой теории информации.

Конечно, книгу я всячески рекомендую для прочтения тем, кто интересуется вопросом квантовых вычислений.

Краткое содержание: Книга даёт подробное и всестороннее введение в новую область исследований изучения роли физических законов (и, особенно, законов квантовой механики) при решении задач информатики. Охвачены такие темы, как квантовые алгоритмы (факторизация, дискретный логарифм), квантовая телепортация, сверхплотное кодирование, устойчивые к ошибкам вычисления, квантовая криптография:

1. Фундаментальные принципы
 - Введение и общий обзор
 - Введение в квантовую механику
 - Введение в информатику
2. Квантовые вычисления
 - Квантовые схемы
 - Квантовое преобразование Фурье и его приложения
 - Квантовые алгоритмы поиска
 - Квантовые компьютеры: физическая реализация
3. Квантовая информация
 - Квантовый шум и квантовые преобразования
 - Меры различия квантовой информации
 - Исправление квантовых ошибок
 - Энтропия и информация
 - Квантовая теория информации

Оценка: 5.

Полезность: 5 4 3.

Количество страниц: 824.

Категории: Квантовые вычисления, Теория вычислений, Учебник.

Ссылка на скачивание: <http://www.twirpx.com/file/976415/>

Ссылка на цитаты из источника: <http://goo.gl/fFm16G>

Квантовые вычисления

Автор: Ожигов Ю. С.

Рецензия: Небольшое методическое пособие, которое даёт введение в тему, но при этом перенасыщено формулами. К тому же, в нём автор даёт довольно-таки интересный, но своеобразный собственный взгляд на природу квантовых вычислений. Почитать интересно, но сквозь многочисленные формулы продираться довольно-таки сложно. Тем не менее, книга может быть охарактеризована как второй шаг после начального для понимания квантовых вычислений.

Краткое содержание: Методическое пособие начинается с краткого введения в теорию квантовой механики, затем кратко (совершенно кратко) описываются положения теории вычислений, затем немного говорится про реализацию квантовых компьютеров «в железе». Две последних главы книги посвящены двум квантовым алгоритмам — Гровера и Шора (опять они, такое ощущение, что других нет), которые рассматриваются столь подробно, что ни в какой другой книги такого нет.

Оценка: 3.

Полезность: 4 2 2.

Количество страниц: 104.

Категории: Квантовая механика, Квантовые вычисления, Теория вычислений.

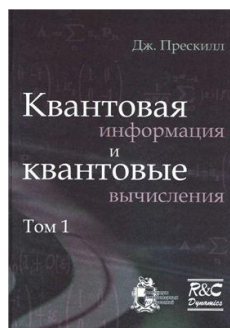
Ссылка на скачивание: <http://www.twirpx.com/file/312946/>

Ссылка на цитаты из источника: <http://goo.gl/SKkGeg>

Квантовая информация и квантовые вычисления. Том 1

Авторы: Прескилл Дж.

Рецензия: Очень хорошая книга от одного из ведущих физиков-теоретиков современности, который в доступных выражениях рассказывает, что такое квантовая информация и квантовые вычисления. Это учебник, написанный на базе лекций, читаемых автором в Калифорнийском технологическом университете, а, следовательно, он наполнен матаном чуть более чем полностью. Однако автору удаётся объяснить сложные вещи простыми словами, а набившие оскомину Алиса и Боб в этой книге становятся занятными персонажами, выполняющими кучу интересных экспериментов и в художественном виде описывающих их. В общем, после книги Нильсена и Чанга — это вторая книга для чтения теми, кто хотел бы глубоко постичь модель квантовых вычислений.



Краткое содержание: Книга написана на базе одноимённого курса, читаемого автором в Калтехе, и представляет собой подробное и всестороннее введение в эту новую, быстро развивающуюся область науки. Ясный физический характер изложения делает книгу доступной для новичка в этой области, находящейся на стыке физики, математики, информатики и технологии. Она содержит необходимые для понимания основного материала сведения из квантовой механики, классической теории информации и основные понятия из классических теорий вычислений и сложности. Каждая глава завершается небольшой подборкой задач разного уровня, способствующих более глубокому усвоению материала.

Оценка: 4.

Полезность: 4 2 3.

Количество страниц: 463.

Категории: Квантовая механика, Квантовые вычисления, Учебник.

Ссылка на скачивание: <http://www.twirpx.com/file/801359/>

Ссылка на цитаты из источника: <http://goo.gl/5Upgl5>

Квантовая информация и квантовые вычисления. Том 2

Авторы: Прескилл Дж.

Рецензия: Во втором томе автор делает упор на квантовую коррекцию ошибок и описывает многочисленные методы и квантовые коды, устойчивые к различным видам ошибок (которых больше видов, чем в классическом случае). Книга будет интересна тем читателям, кто уже продвинулся на стезе изучения модели квантовых вычислений до понимания более сложных тем, нежели просто описание этой модели. Однако тема, раскрываемая в этой книге, больше предназначена для тех, кто будет конструировать аппаратное обеспечение будущего квантового вычислительного устройства.

Оценка: 4.

Полезность: 4 2 3.

Количество страниц: 313.

Категории: Квантовая механика, Квантовые вычисления, Учебник.

Ссылка на скачивание: <http://www.twirpx.com/file/801360/>

Ссылка на цитаты из источника: <http://goo.gl/5Upgl5>

Квантовые вычисления. За и против

Автор: Садовничий В. А.

Рецензия: Ректор МГУ Виктор Садовничий, видимо, тоже озаботился темой квантовых вычислений и реализации квантовых компьютеров, поэтому начал выпускать сборник переводов статей на эту волнующую тему. В итоге, вышло всего два сборника (этот — первый), что очень жаль. Этот сборник содержит 15 статей самых различных авторов, в том числе и таких мастодонтов, как Гровер Л. К. и Дойч Д. Сборник начинается с вводных статей по модели квантовых вычислений, потом приводится ряд статей о физической реализации и с описанием физических экспериментов, а заканчивается интереснейшими приложениями модели — квантовой теорией игр, квантовой робототехникой и квантовой теорией хаоса. В общем, книга будет небезынтересной программистам, математикам и физикам.



Краткое содержание: В сборнике приводятся переводы следующих статей:

1. Браунштейн С. Л. Квантовые вычисления: учебное руководство
2. ДиВинченцо Д. П. Квантовые вычисления
3. Баренко А. и др. Условная квантовая динамика и логические гейты
4. Прескилл Дж. Квантовые вычисления: за и против
5. Гровер Л. К. Квантовая механика помогает найти иголку в стоге сена
6. Гровер Л. К. Польза суперпозиции
7. Боувмеестр Д. и др. Экспериментальная квантовая телепортация
8. Чуанг А. Л. и др. Экспериментальная реализация квантового алгоритма
9. Гершенфелд Н. и др. Квантовые вычисления с молекулами

10. Джоунс А. Быстрый поиск с ядерно-магнитным резонансным компьютером
11. Айсерт Й. и др. Квантовые игры и квантовые стратегии
12. Бенёв П. Квантовые роботы и окружающая среда
13. Абрамс С. и др. Квантовый алгоритм вычисления собственных значений и собственных векторов, обеспечивающий экспоненциальное увеличение скорости
14. Шак Р. Исследование квантового хаоса с помощью квантового компьютера
15. Черны В. Квантовые компьютеры и труднорешаемые (NP-полные) задачи

Оценка: 3.

Полезность: 4 4 2.

Количество страниц: 212.

Категории: Квантовая механика, Квантовые вычисления, Сборник статей, Теория вычислений, Теория игр, Теория хаоса.

Ссылка на скачивание: <http://www.twirpx.com/file/343966/>

Ссылка на цитаты из источника: <http://goo.gl/EQZIIw>

Квантовые компьютеры и квантовые вычисления 2000, № 01

Автор: Садовничий В. А.

Рецензия: Это здорово, что Виктор Садовничий начал выпускать журнал по квантовым вычислениям. Плохо то, что он это дело забросил, ибо тема очень интересная, на Западе ежегодно появляется огромное количество материалов как по физике квантовых вычислений, так и по математике. Ежегодно появляются новые алгоритмы. А у нас эта тема не исследуется вовсе. Первый выпуск журнала, по всей видимости, собирался из статей по принципу клича на весь мир. Из-

за этого он не особо интересный. Но для анналов темы сохранить можно.

Краткое содержание: Первый номер международного периодического издания по квантовым вычислениям, которое начал выпускать В. А. Садовничий, так это дело и не запустилось. В этом номере приведены некоторые статьи, которые, видимо, набрались к выпуску номера:

1. Rieffel E., Polak W. Основы квантовых вычислений
2. Fedichkin L., Yanchenko M., Valiev K. A. Novel coherent quantum bit using spatial quantization levels in semiconductor quantum dot
3. V'yurkov V. V., Gorelik L. Y. Charge based quantum computer without charge transfer
4. Fedichkin L. Polynomial scheme of avoiding multiqubit errors arising due qubit-qubit interaction
5. Laiho R., Молотков С. Н., Назин С. С. О телепортации полностью неизвестного однофотонного состояния
6. Dugic M. Decoherence-induced suppression of decoherence in quantum computation

МЕЖДУНАРОДНЫЙ НАУЧНЫЙ ЖУРНАЛ
Том 1. №1 2000

КВАНТОВЫЕ КОМПЬЮТЕРЫ



КВАНТОВЫЕ ВЫЧИСЛЕНИЯ

Quantum
Computers & Computing

Школа по квантовым компьютерам

Оценка: 3.

Полезность: 1 2 1.

Количество страниц: 117.

Категории: Журналы, Квантовые вычисления, Сборник статей.

Ссылка на скачивание: <http://www.twirpx.com/file/344004/>

Квантовые компьютеры и квантовые вычисления. Выпуск 2

Автор: Садовничий В. А.

Рецензия: Второй выпуск журнала, который начал выпускать ректор МГУ и который посвящён квантовым вычислениям, полностью состоит из очень давних, но тем не менее ценных статей. Воистину, это жемчужины модели квантовых вычислений, и очень здорово, что есть сборник, в котором все они представлены. Особенный интерес представляют статьи П. Шора и Д. Дойча, в которых описываются алгоритмы, теперь носящие имена этих замечательных учёных. А лекции Р. Фейнмана просто необыкновенны (крайне рекомендуются для ознакомления — этот сборник можно найти и открыть только ради этих двух статей Р. Фейнмана). Ну и остальные статьи будут небезынтересны тем, кто интересуется темой.

Краткое содержание: А это второй номер журнала, который посвящён переводам статей основоположников теории квантовых вычислений. Здесь собраны все базисные материалы, а именно:

1. Ландауэр Р. Необратимость и выделение тепла в процессе вычислений
2. Беннетт Ч. Логическая обратимость вычислений
3. Бенёв П. Квантовомеханические гамильтоновы модели машин Тьюринга
4. Фейнман Р. Моделирование физики на компьютерах
5. Фейнман Р. Квантовомеханические компьютеры
6. Дойч Д. Квантовая теория, принцип Чёрча — Тьюринга и универсальный квантовый компьютер
7. Дойч Д., Джозса Р. Быстрое решение задач с помощью квантовых вычислений

8. Шор П. Полиномиальные по времени алгоритмы разложения числа на простые множители и нахождения дискретного логарифма для квантового компьютера
9. Манин Ю. И. Классическое вычисление, квантовое вычисление и факторизация Шора

Оценка: 4.

Полезность: 3 3 1.

Количество страниц: 288.

Категории: Журналы, Квантовые вычисления, Сборник статей.

Ссылка на скачивание: <http://www.twirpx.com/file/1312632/>

Ссылка на цитаты из источника: <http://goo.gl/b0Y6TN>

Квантовые вычисления

Автор: Стин Э.

Рецензия: Довольно пустая книга, да ещё и наполненная ошибками (скорее всего, ошибками перевода). Неправильно переведены термины, некорректно указаны некоторые фамилии (многие, кстати). Ну и по факту квантовые алгоритмы описаны так, что их невозможно реализовать, если не знать сути алгоритмов. То есть, похоже, что автор не вдумывался в алгоритмы, а просто переписал математическое определение оных. Ну да так все делают. Ничего интересного.

Краткое содержание: В книге рассматриваются такие темы:

- Классическая теория информации
- Классическая теория вычислений
- Квантовая физика против физики классической
- Квантовая информация

- Универсальный квантовый компьютер
- Экспериментальные процессоры, оперирующие квантовой информацией
- Исправление квантовых ошибок

Оценка: 3.

Полезность: 2 1 1.

Количество страниц: 111.

Категории: Квантовые вычисления.

Ссылка на скачивание: <http://www.twirpx.com/file/170554/>

Ссылка на цитаты из источника: <http://goo.gl/m1ZAeq>

Великая квантовая революция

Автор: Фейгин О. О.

Рецензия: Как говорится, автор начал за здоровье, а закончил за упокой. Вроде как книга обозначена как популярное изложение идей квантовой механики, включая квантовые вычисления, однако собственно квантовой механике посвящена только первая глава, которая написана более-менее нормальным языком. Далее понеслось какое-то откровенное мракобесие, причём за мыслью автора уследить было никак нельзя. К третьей главе автор так разошёлся, что зачастую уже было не понятно, как в одном предложении могут соединяться собранные в нём слова. Полная бессмыслица. Например, абзац начинался про Юпитер, а заканчивался про чёрные дыры, при этом связи не было никакой. Ну и особенно порадовал пассаж про то, как белая дыра располагается СВЕРХУ чёрной дыры, и из белой на чёрную



вываливается материя, причём перешедшая назад во времени на 10 минут. В общем, разочарован, читать не советую.

Краткое содержание: В книге три главы, и автор начинает рассказывать про основы квантовой механики, включая краткое описание модели квантовых вычислений, квантовой информации и квантовой криптографии. В этой главе удивляет то, что картинки и часто даже сам текст взят из других источников без указания на них. Во второй главе автор переключается на интерпретации квантовой механики и много времени посвящает многомировой интерпретации Эверетта. Также один раздел описывает странные воззрения Д. Бома на квантовую механику. Далее в третьей главе описываются основы космологии, причём автора начинает заносить так, что читать становится практически невозможно. Рассмотрено немного теории относительности, далее теория струн (совсем немного), а потом понеслись какие-то безумные философствования. Вся книга перемежена странными изображениями, как говорится, ни к селу, ни к городу.

Оценка: 2.

Полезность: 3 2 1.

Количество страниц: 272.

Категории: Квантовая механика, Квантовые вычисления, Космология, Популярная наука, Теория относительности, Теория струн.

Ссылка на скачивание: <http://www.twirpx.com/file/672806/>

Ссылка на цитаты из источника: <http://goo.gl/2Hk62P>

Введение в квантовую теорию информации

Автор: Холево А. С.

Рецензия: Ещё одна наполненная тяжёлыми матанами чуть более чем полностью книга, которая основана на курсе лекций автора. Чтобы прочитать и понять её, необходимо уже быть специалистом по квантовой механике и квантовым вычислениям. Алгоритмы в книге практически не рассматриваются, только упоминаются. Остальные темы, связанные с квантовой теорией информации, рассматриваются верхами. Впрочем, у книги есть достоинство — она концентрирует информацию, являясь эдаким индексом. То есть её можно использовать для того, чтобы быстро получить необходимые сведения, а потом обратиться к более существенным источникам.

Краткое содержание: В книге рассматриваются следующие темы:

- Основные понятия классической теории информации
- Состояния и наблюдаемые
- Применения сцепленных состояний
- Оптимальное различение квантовых состояний
- Классическая пропускная способность квантового канала связи
- Квантовые каналы
- Энтропийные характеристики квантовых систем
- Передача классической информации при помощи сцепленного состояния
- Квантовая пропускная способность и когерентная информация
- Квантовые коды, исправляющие ошибки

Оценка: 2.

Полезность: 2 1 1.

Количество страниц: 127.

Категории: Квантовые вычисления.

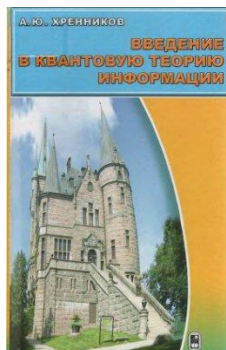
Ссылка на скачивание: <http://www.twirpx.com/file/115955/>

Ссылка на цитаты из источника: <http://goo.gl/Xqps7u>

Введение в квантовую теорию информации

Автор: Хренников А. Ю.

Рецензия: Очень занятная книга нашего автора, который переступил каноны и описал квантовую механику и модель квантовых вычислений со своей, можно сказать аутентичной точки зрения. Нет, математический аппарат описан строго, здесь отступлений быть не может. А вот интерпретации свои, и этим книга интересна. Для будущих программистов на квантовых языках программирования книга будет очень интересна тем, что в ней квантовые алгоритмы (Гровера, Дойча, Саймона и два алгоритма Шора) описаны очень просто и именно как алгоритмы, а не как набор математических формул и их нагромождений.



Краткое содержание: Предлагаемая книга представляет собой введение в квантовую теорию информации и основания квантовой механики. При этом квантовая информация понимается в существенно более широком смысле, чем в стандартных учебниках, и не сводится лишь к математическому описанию передачи информации, криптографии и вычислениям с помощью квантовых носителей информации, например фотонов. Поэтому книга содержит приложения математических методов квантовой теории информации за пределами физики: к психологии, когнитивным наукам, экономике и финансам. Предварительных знаний о квантовой и (или) классической механике не требуется. В краткой форме представлен математический аппарат квантовой механики (теория операторов

в гильбертовом пространстве). Сложные математические рассуждения возникают лишь в последних главах книги, посвящённых попыткам выйти за пределы стандартного квантового формализма.

Оценка: 4.

Полезность: 4 4 2.

Количество страниц: 286.

Категории: Квантовые вычисления, Квантовая механика, Философия.

Ссылка на скачивание: <http://www.twirpx.com/file/479316/>

Ссылка на цитаты из источника: <http://goo.gl/hDDqGP>

Квантовые компьютеры

Авторы: Цыганков В. С., Сементин С. А., Кучеренко А. О.

Рецензия: Ещё одно методическое пособие по квантовым вычислениям, которое было написано в каком-то региональном ВУЗе. И, видимо, это реально является фактором, который влияет на ценность книги и стиль её изложения. Для чтения этого методического пособия достаточно общего образования и понимания технических вопросов, и оно даёт первоначальное понимание того, что такое модель квантовых вычислений. Реально, для чтения не требуется специальных знаний по квантовой механике, и после прочтения можно сделать самое первое понимание этого. Так что порекомендую это небольшое пособие всем интересующимся темой.

Краткое содержание: Пособие начинается с краткого введения в тему, рассмотрения исторических предпосылок появления модели квантовых вычислений. Далее тщательнейшее внимание уделено так называемым обратимым вычислениям, и сначала делается акцент на логически обратимых вычислениях.

ях, а потом рассмотрение переходит к унитарным преобразованиям. Затем изложение переходит к рассмотрению возможных физических реализаций — большая часть изложения посвящена ЯМР-реализации. В конце рассматриваются три алгоритма — Шора, Дойча и Гровера.

Оценка: 5.

Полезность: 4 3 3.

Количество страниц: 53.

Категории: Квантовые вычисления, Теория вычислений.

Ссылка на скачивание: <http://www.twirpx.com/file/326479/>

Ссылка на цитаты из источника: <http://goo.gl/gTaQ0y>

Квантовая информатика

Автор: Чивилихин С. А.

Рецензия: Методическое пособие, которое довольно сильно отличается от всех предыдущих книг и других материалов на тему квантовых вычислений в лучшую сторону. В нём нет перегруженных формул, но есть последовательное изложение описания модели квантовых вычислений с применением аппарата линейной алгебры. Не очень приятным аспектом издания является большое количество повторов, но сам текст изложен очень доступно. Если знать основы квантовых вычислений, то книга читается очень легко.

Краткое содержание: В книге последовательно рассматривается пять тем. Первая — основные принципы квантовой информатики. Затем автор рассматривает однокубитовые и двухкубитовые квантовые системы, описывая разнообразные гейты (X, Y, Z, T, Ф, CNOT). Далее в четвёртой главе кратко рассматриваются прикладные аспекты квантовых вычислений, приводится в пример алгоритм Дойча (распознавание функ-

ции). Наконец, в пятой главе даётся понятие информационной энтропии с квантовой точки зрения.

Оценка: 4.

Полезность: 4 1 1.

Количество страниц: 80.

Категории: Квантовые вычисления, Теория вычислений, Теория информации.

Ссылка на скачивание: <http://www.twirpx.com/file/367740/>

Ссылка на цитаты из источника: <http://goo.gl/9p522O>



В процессе подготовки и написания этой книги мною так же были переведены некоторые материалы по квантовым вычислениям, и многим читателям будет небезынтересно ознакомиться и с ними. На текущий момент есть следующие переводы:

- Грин А. С., Лумсдаине П. Л., Росс Н. Дж., Зелингер П., Валирон Б. *Введение в квантовое программирование на языке Qipper*. Ссылка: <http://goo.gl/A9XXUg>.
- Грин А. С., Зелингер П., Лумсдаине П. Л., Валирон Б., Росс Н. Дж. *Масштабируемый язык квантовых вычислений*. Ссылка: <http://goo.gl/Xz14oj>.
- Хаес Б. *Программируем свой квантовый компьютер*. Ссылка: <http://goo.gl/bkyPrW>.
- Унру Д. *Квантовые языки программирования*. Ссылка: <http://goo.gl/ew1IKM>

Конечно, в процессе дальнейшей работы этот список будет увеличиваться, поэтому все заинтересованные читатели приглашаются в каталог [«Переводы»](#), где находятся все самые последние материалы.

Чтобы читателю было легче ориентироваться в этом перечне, ниже приводится блок-схема, которая позволяет выбрать следующую книгу для чтения исходя из целей и потребностей читателя.

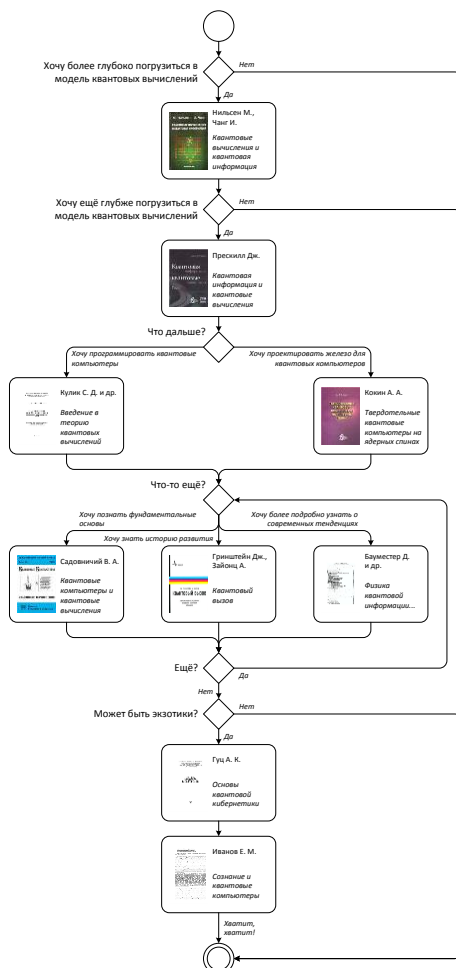


Рис. 32. Дорожная карта чтения литературы по квантовым вычислениям

Обзор видеокурсов по квантовым вычислениям и смежным темам

Поскольку в последние времена стало очень популярным создавать онлайн-курсы по различным предметам, процесс этот не обошёл и тему квантовых вычислений. Мною было прослушано несколько таких курсов (но не все имеющиеся в сети Интернет), поэтому в этом приложении будут приведены описания таких курсов.

Надо отметить, что курсов, в которых рассматривается именно модель квантовых вычислений, на текущий момент нигде не представлено. На площадке Coursera был такой курс (см. описание далее), однако затем оттуда исчез даже его архив. Другие описанные здесь курсы относятся к модели квантовых вычислений каким-либо аспектом. Ну а те из читателей, кто не смог просмотреть курс по квантовым вычислениям, но всё же хотел бы это сделать, могут написать мне запрос по электронной почте (roman.dushkin@gmail.com), я что-нибудь придумаю.

Далее приводится стандартизированное описание нескольких курсов. Большинство реквизитов понятны сами по себе, лишь два требуют дополнительного пояснения. Реквизит «Сложность» обозначает требование специальных знаний до начала прослушивания курса. Если этот реквизит равен 5, то курс достаточно сложен, в нём много предварительных требований. Соответственно, значение 1 обозначает, что курс может быть понят школьником. Реквизит «Язык» обозначает простоту использования английского языка лектором. Большинство лекторов в МООС используют очень простой язык, понимая, что их курсы прослушиваются разными людьми по всему миру. Однако иногда встречаются и такие, кого очень сложно понять. Чем ниже значение этого реквизита, тем труднее понять лектора.

Quantum Mechanics and Quantum Computation **(Квантовая механика и квантовые вычисления)**

Лектор: Вазирани У.

Университет: Университет Калифорнии в Беркли.

Старт обучения: Июль 2012.

Количество недель обучения: 10.

Научные направления: CS: Теория, Физика и науки о Земле, CS: Системы, безопасность, сети.

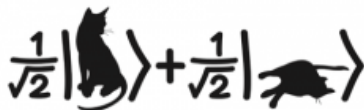
Объём лекций в неделю (примерно): 2 блока лекций (2 темы) по 1.5 — 2 часа.

Сложность: 5.

Язык: 4.

Программирование: Нет.

Описание курса: Крайне интересный курс от одного из апологетов модели квантовых вычислений. Курс более чем полностью наполнен таким мозгодробительным



матаном, что без специальной подготовки к нему лучше не подступаться. В первую очередь, необходимы суровые знания в линейной алгебре, желательно на уровне понимания тензорных операций. Физику тоже желательно знать, равно как и теорию сложности алгоритмов. Без этого очень многое в курсе будет просто непонятно. Первая половина курса даёт краткое введение в квантовую механику, а во второй лектор кратко рассказывает, что такое квантовый алгоритм, после чего даёт описание нескольких таких алгоритмов. Наиболее известный из них — алгоритм факторизации Шора.

Описание процедуры сертификации: В процессе слушания курса каждую неделю необходимо сдавать довольно мощный тест (всего 7 штук, стоимость — 70 % от итоговой оценки), причём сдавать можно 10 раз, но на каждую следующую попытку накладывается штраф, на 10 % превышающий предыдущий (то есть первая попытка без штрафа, вторая — 10 %, третья — 20 % и т. д.). По результатам таких попыток берётся максимальная, но в любом случае лучше всего сдавать на максимум с первого раза. Экзамен стоит 30 % итоговой оценки, можно попытаться сдать только один раз. Проходной балл — 80 %, что делает курс очень суровым.

Ссылка на интеллект-карту: <http://goo.gl/Edyxh4>

Дополнительные материалы:

- [Символьные вычисления на примере решения одной не-сложной задачи по квантовой механике](#)

Exploring Quantum Physics

(Введение в квантовую механику)

Лекторы: Галицкий В., Кларк Ч., Аппельбаум Я.

Университет: Университет Мериленда.

Старт обучения: Март 2013.

Количество недель обучения: 8.

Научные направления: Физика.

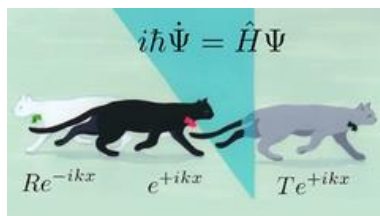
Объём лекций в неделю (примерно): 2 блока по 1 часу.

Сложность: 5.

Язык: 5.

Программирование: Нет.

Описание курса: Курс типа «галопом по европам» квантовой механике. Меня заинтересовал тем, что мне хотелось более глубоко погрузиться в эту интереснейшую область



знаний после прохождения курса по квантовым вычислениям. Однако мои ожидания практически не оправдались. Три лектора курса (один был приглашён в процессе) давали крайне сложные объяснения различных, часто не связанных друг с другом тем. Иной раз мне казалось, что для того, чтобы понять курс, надо было до того проучиться пару семестров на кафедре квантовой механики. И снова я убеждался, что решение задач в таких вещах может быть проведено чисто синтаксически, без понимания сути вычислений. Впрочем, многие специалисты по квантовой механике с этим соглашаются. И, кстати, они так и не объяснили толком, что такое «спин», поэтому он так

и остался для меня «некоторым специальным параметром, введённым для решения некоторых задач».

Описание процедуры сертификации: Для получения сертификата об успешном окончании необходимо было выполнить 7 домашних заданий (70 % выходной оценки, по 10 % каждое) и сдать выпускной экзамен (30 % выходной оценки). Домашние задания можно было сдавать 3 — 4 раза, выпускной экзамен только один раз, причём время сдачи было ограничено 6-ью часами. Большинство задач было на выбор из 5 — 6 альтернатив, но были и довольно занимательные задачи со вводом расчётных значений. В целом, сложность домашних заданий и выпускного экзамена оценивается мной как умеренная — многие задачи я решал именно синтаксически, просто манипулируя символами по известным математическим правилам. Проходной балл — 60 %.

Ссылка на интеллект-карту: <http://goo.gl/KtDWSN>

Дополнительные материалы: Нет.

A Look at Nuclear Science and Technology **(Введение в атомную электроэнергетику)**

Лектор: Фоулк Л.

Университет: Питтсбургский Университет.

Старт обучения: Июнь 2013.

Количество недель обучения: 8.

Научные направления: Инженерные науки, Физика, Химия.

Объём лекций в неделю (примерно): 1 блок из 5 лекций общим объёмом 1.5 часа.

Сложность: 4.

Язык: 4.

Программирование: Нет.

Описание курса: Вполне зачётный курс, позволяющий постичь суть атомной электроэнергетики. Автор курса начал с самых основ — рассказал, что такое ядерный распад, какие типы



ядерных распадов бывают, свойства альфа-, бета- и гамма-распадов, а также указал на особую роль нейтронов. Если у слушателя физическое образование и сносные знания в области квантовой механики, то вводная часть курса будет мало интересна. Примерно с четвёртой недели начинается введение в технологию производства электроэнергии на АЭС. Рассматриваются такие темы, как устройство некоторых типов ядерных реакторов, государственное регулирование в области атомной энергетики, безопасность и её обеспечение, контроль рисков и т. д. Всё это довольно интересно.

Описание процедуры сертификации: Еженедельно для первых семи недель публиковался экзамен, состоящий из десяти вопросов, каждый из которых давал один балл. Для сдачи давалось три попытки и две недели времени. На восьмой неделе был опубликован дополнительный «финальный экзамен» для тех, кто пропустил или плохо сдал какой-либо из недельных экзаменов. Все вопросы были крайне просты, более того, они дублировались из тестов (внутри видео и по итогам каждого видео). Большую часть экзаменов можно было без труда сдать на 10 баллов с первой попытки. Проходной балл — 70 %, для получения сертификата с отличием — 90 %.

Ссылка на интеллект-карту: <http://goo.gl/gWNUK3>

Дополнительные материалы: Нет.

Energy 101

(Энергия)

Лектор: Шэлтон С.

Университет: Технологический институт Джорджии.

Старт обучения: Январь 2013.

Количество недель обучения: 9.

Научные направления: Физика, Науки о Земле.

Объём лекций в неделю (примерно): 1 блок на 1 час.

Сложность: 3.

Язык: 4.

Программирование: Нет.

Описание курса: Довольно поверхностный курс про то, что такое энергия, откуда она берётся, как преобразуется из одной формы в другую и т. д., который, тем не менее, я буду реко-



мендовать прослушать каждому человеку с гуманитарным складом мышления. Ибо иной раз читаешь такую ахинею на тему энергосбережения, добычи полезных ископаемых и т. д., что рука сама тянется к лицу. В целом, курс очень заточен на США, все примеры относятся только к этому государству, есть даже раздел про энергетическую и национальную безопасность США. Всё это можно спокойно пропускать без ущерба для общего понимания курса и тем, в нём рассматриваемых. Про Россию, кстати, в курсе тоже говорится в связи с описанием углеводородного энергетического сырья, его до-

бычи и т. д. Очень интересны темы про так называемую «альтернативную» энергетику, особенно выводы.

Описание процедуры сертификации: Курс настолько простой, что и процедура сертификации для него особого смысла не имеет. Тем не менее, к каждой лекции курса есть небольшой опросник, состоящий от трёх до шести вопросов, для ответа на каждый из которых необходимо выбрать одну из четырёх (обычно) альтернатив. Вопросы никогда не меняются, альтернативы тоже (только их порядок), а для ответа на каждый опросник дано 10 попыток без штрафов. Поэтому получить 100 % можно даже просто подбирая ответы. Тем не менее, вопросы настолько просты, что в некоторых опросниках я получал по 100 %, даже не слушая лекций (потом слушал). А на другие вопросы можно ответить легко, остановив лекцию как раз там, о чём вопрос.

Ссылка на интеллект-карту: <http://goo.gl/Ipih3K>

Дополнительные материалы: Нет.



В сети Интернет как на самом мощном МООС-ресурсе Coursera, так и в других местах ещё есть несколько курсов, которые либо напрямую, либо опосредовано связаны с темой квантовых вычислений. Поскольку мне не довелось прослушать эти курсы, то далее они просто будут перечислены:

- Курс «Introduction to Physical Chemistry» (Введение в физическую химию) на Coursera. Очевидно, что должен затрагивать аспекты квантовой механики. Однако вряд ли в нём упоминаются квантовые вычисления в принципе.
- Курс «Cryptography II» (Криптография 2) на Coursera. Темой одной из недель обучения являются квантовые протоколы обмена информацией и их практическое применение к криптографическим задачам.

- Курс «Energy 101» (Введение в энергетику) на edX. Скорее всего, повторяет одноимённый курс на Coursera, хотя читается иным лектором и идёт от иного университета.
- Курс «Mastering Quantum Mechanics» (Инструментарий квантовой механики) на edX. Небольшой курс, в котором рассматриваются отдельные вопросы квантовой механики, а упор сделан на математическом инструментарии.
- Курс «Quantum Mechanics for Scientists and Engineers» (Квантовая механика для учёных и инженеров) на Stanford OpenEdx. Ещё один курс с рассмотрением основ квантовой механики, который позиционируется как курс для более продвинутых учеников, у которых ещё нет, но которые хотят получить первоначальные знания в области квантовой механики.
- Курс «Квантовая механика» на Лектории. Представляет собой просто многочасовые видео лекций МФТИ по квантовой механике. Может быть интересен тем, что основан на отечественной школе, а также ведётся на русском языке.

Если же кто из читателей знает видео-курс по квантовым вычислениям или квантовой механике, равно как и по смежным темам, который здесь не указан, то просьба сообщить мне о нём.

Заключение

И вот книга подходит к своему завершению. С одной стороны я, как автор, рад тому, что могу донести до своего читателя те все новые знания, которые получил сам в процессе изучения модели квантовых вычислений. Да чего скрывать, даже в процессе написания этой книги мне пришлось изучить много всего такого, чего я не знал раньше.

Но с другой стороны мне немного грустно прощаться, поскольку тема квантовых вычислений безгранична, а сегодня вообще человек стоит только лишь на пороге понимания этой замечательной модели. И то, что в этой книге мне удалось рассмотреть только 4 квантовых алгоритма, несколько меня обескураживает.

Дальше можно было бы развить начатый фреймворк для квантовых вычислений на языке Haskell, либо реализовывать всё новые квантовые алгоритмы на языке программирования Quipper. Надеюсь, что восторженный читатель для своей тренировки и углубления полученных знаний сделает что-либо подобное.

В общем, охватывая взглядом весь написанный текст, я понимаю, что написал лишь самую малость из того, что можно было бы написать. И при этом книга даёт лишь напутствие тем читателям, которые захотят двигаться дальше. Теперь, прочитав эту книгу, вы знаете достаточно для того, чтобы идти вперёд в изучении модели квантовых вычислений, а приведённая схема дальнейшего чтения поможет в этом.

Вот перечень дополнительных и более глубоких тем в рамках изучения модели квантовых вычислений, с которыми заинтересованный читатель может начать знакомиться сразу после прочтения этой книги:

- Расширенное введение в математику квантовой механики.
- Обратимые вычисления и модель «бильярдных шаров».
- Квантовая телепортация.
- Парадокс ЭПР и эксперимент Белла.
- Описание многочисленных вариантов гейтов (унитарных операторов) и их действий над кубитами и их системами.
- Удаление промежуточных результатов вычислений («мусорных» битов) без потери обратимости.
- Квантовая машина Тьюринга и квантовое лямбда-исчисление.
- Теория квантовой информации, включая теоремы Шеннона для каналов без шума и с шумом в условиях передачи квантовой информации.
- Квантовая коррекция ошибок и построение помехоустойчивых квантовых кодов.
- Расширение теории сложности для учёта модели квантовых вычислений.
- Перспективные исследования по физической реализации квантовых вычислений.

Большинство этих тем рассматривается с той или иной степенью глубины в описанных ранее книгах по квантовой меха-

нике и квантовым вычислениям. Я надеюсь, что читатель после прочтения этой книги и ознакомления с моим мнением относительно имеющейся на текущий момент времени литературы по вопросу, сможет самостоятельно составить план своего дальнейшего обучения этой непростой, но очень интересной вычислительной модели.

Ну и остаётся написать, что я буду всегда рад получить отзывы, замечания, комментарии и предложения. Пишите на электронную почту. Ни одно из конструктивных писем не останется без ответа.

Низкий поклон спонсорам

Проект данной книги был профинансирован при помощи модной сегодня технологии краудфандинга. В народном сборе средств приняло участие 159 спонсоров, которые общим итогом превысили запрошенную сумму на 70 %.

Вот славные имена тех спонсоров, кто хотел указания их имени на специальной странице книги (приводятся в алфавитном порядке):

- Айткулов Павел
- Антонов Юрий
- Астанин Сергей
- Астапов Дмитрий
- Базанов Иван
- Белокрыс Глеб
- Буцкий Станислав
- Вознюк Алексей
- Воронина Марина
- Гадельшин Артур
- Галкин Василий
- Гинзбург Даниэль
- Гневушев Евгений
- Гурьянов Антон
- Дранников Сергей
- Евсеев Антон
- Егоров Иван
- Жмурин Андрей
- Иванова Татьяна
- Казменко Иван
- Катунин Павел
- Кийко Владимир
- Кириллов Александр
- Киричков Александр
- Клим Тимоти
- Корнеев Александр
- Корягин Константин
- Коцеруба Виктор
- Крентовский Максим
- Кудрявцев Илья
- Левченко Владимир
- Логунцов Сергей
- Маврин Артём
- Майоров Андрей
- Макаров Евгений
- Невидимкин Дмитрий
- Орищенко Олег
- Отт Алекс
- Пахомов Всеволод
- Пачин Юрий
- Пехтерев Виталий
- Пилипенко Геннадий
- Попов Евгений
- Руд Дмитрий
- Савиных Андрей
- Сергеев Дмитрий
- Сеницын Андрей
- Сурис Дженни
- Файзуллин Рустам
- Федюшин Андрей
- Фионов Сергей
- Фоллэ Виктор
- Фролкин Антон
- Шведунوف Александр
- Шурпин Антон
- Яковлев Пётр

Принимаются благодарности

Коллеги и симпатизанты! При подготовке этой книги было затрачено очень много времени и других ресурсов, материальных и нематериальных. Я стоял и крепко стою на позиции о том, что информация и знания должны быть свободны. Однако вкладываться на общественных началах в распространение знаний и повышение общего уровня квалификации наших соотечественников становится всё сложнее и сложнее.

Эта книга всё так же остаётся бесплатной для скачивания и распространения. Но здесь хотелось бы настоять на **обязательности** перечисления благодарности за этот материал, уникальный во всех отношениях для российского книжного рынка и электронного рынка информации. Ваша благодарность в обязательном порядке ожидается по следующим реквизитам:

- Яндекс.Деньги: **4100137733052**
- WebMoney: **R211895623295**
- PayPal: **darkus.14@gmail.com**

Следует напомнить, что на счета в этих платёжных системах благодарность можно перечислить и при помощи терминалов мгновенной оплаты.

Также лиц, заинтересованных в сотрудничестве по вопросам издания, распространения, написания новых книг и т. д., прошу обращаться по адресу электронной почты **roman.dushkin@gmail.com**.

Душкин Р. В.

**Квантовые вычисления и функциональное про-
граммирование**

Вёрстка *Душкин Р. В.*